Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Matemática

Anti-unification in Absorptive Theories (Anti-unificação em Teorias Absortivas)

Andrés Felipe González Barragán

Tese apresentada como requisito parcial para
conclusão do Doutorado em Matemática

Orientador
Prof. Dr. Mauricio Ayala-Rincón

Coorientador
Prof. Dr. Temur Kutsia

Brasília
2025

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Matemática

Anti-unification in Absorptive Theories (Anti-unificação em Teorias Absortivas)

Andrés Felipe González Barragán

Tese apresentada como requisito parcial para
conclusão do Doutorado em Matemática

Prof. Dr. Mauricio Ayala-Rincón (Orientador)
Universidade de Brasília, Brasil
Prof. Dr. Temur Kutsia (Coorientador)
RISC/Johannes Kepler Universität - Linz

Prof. Dr. Santiago Escobar
Universitat Politècnica de València, Espanha

Prof. Dr. Thaynara Arielly de Lima
Universidade Federal de Goiás, Brasil

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

Prof. Dr. David Michael Cerna
Czech Academy of Sciences, República Tcheca

Prof. Dr. Giovany de Jesus Malcher Figueiredo
Coordenador do Programa de Pós-graduação em Matemática

Brasília, 02 de Outubro de 2025

Resumo

A anti-unificação consiste em determinar uma expressão que represente a estrutura comum a duas expressões dadas, denominada *generalização* dessas expressões. O desenvolvimento de procedimentos de anti-unificação para o cálculo de generalizações tem adquirido importância crescente em áreas como análise de programas, detecção de clones e raciocínio simbólico. Enquanto os trabalhos iniciais se concentraram na anti-unificação *sintática*, aplicações modernas frequentemente exigem raciocínio módulo teorias equacionais. Esta tese dedica-se à anti-unificação em teorias absortivas, nas quais determinados símbolos de operação "absortivos" satisfazem os axiomas $f(\varepsilon_f, x) \approx \varepsilon_f$ and $f(x, \varepsilon_f) \approx \varepsilon_f$ para uma constante "absorvente" relativa ε_f; exemplos incluem multiplicação e zero, conjunção e falso, interseção e conjunto vazio.

Neste trabalho são apresentadas ferramentas conceituais e um algoritmo correto para anti-unificação modulo teorias absortivas, formalizado por meio de um conjunto de regras de inferência que transformam configurações, estruturas que codificam problemas de anti-unificação e soluções parciais. São demonstradas propriedades de preservação das configurações sob a aplicação das regras de inferência, garantindo a correção do algoritmo. Uma contribuição central deste trabalho é o refinamento da análise de completude de um algoritmo apresentado na *12ª International Joint Conference on Automated Reasoning* (2024). São identificadas generalizações adicionais não consideradas anteriormente, as regras de inferência são melhoradas, e demonstra-se como as soluções correspondentes podem ser calculadas a partir das configurações finais.

Esta tese também aborda teorias que podem incluir símbolos funcionais absortivos, comutativos e absortivo-comutativos, que, para abreviar, chamamos de teorias absortivo-comutativas. Propõe-se um novo conjunto de regras de inferência, acompanhado da prova de correção do algoritmo resultante, estendendo a aplicabilidade da abordagem a estruturas de termos mais expressivas encontradas em cenários práticos. Além disso, é tratado o caso restrito de anti-unificação para generalizações lineares, nas quais cada variável ocorre no máximo uma vez nas soluções. O algoritmo é adaptado para o caso linear absortivo e comutativo, provando-se sua correção e completude. No caso linear puramente absortivo, o algoritmo computa um conjunto mínimo e completo de generalizações.

Palavras-chave: Anti-unificação, Generalização, Teoria Equacional, Teoria Absortiva.

Resumo Estendido

Motivação A anti-unificação (AU), também chamada de generalização, é um problema fundamental no raciocínio indutivo que consiste em encontrar uma expressão simbólica que capture a estrutura comum entre duas expressões dadas. Esse problema pode ser visto informalmente como o dual da unificação: enquanto a unificação determina como identificar duas expressões da forma mais geral possível, a anti-unificação busca a expressão mais específica que contenha as semelhanças compartilhadas entre ambas. Na literatura, essas expressões mais específicas são conhecidas como *generalizadores menos gerais* (em inglês, *least general generalizations*, ou lggs).

Por exemplo, no contexto sintático, em que nenhuma propriedade algébrica, como a comutatividade ou a associatividade, é considerada, sejam f um símbolo de função binária, a, b, c constantes e x, y variáveis. Então x e $f(w, y)$ são generalizações das expressões $f(f(a, a), c)$ e $f(f(b, b), c)$. Além disso, a expressão $f(f(z, z), c)$ é a generalização mais específica possível, contendo a estrutura comum às expressões acima.

O interesse pela anti-unificação tem crescido substancialmente devido à sua ampla gama de aplicações acadêmicas e industriais em ciência da computação, tais como Aprendizado de Máquina (*Machine Learning*), Raciocínio Automatizado e Lógica Formal. Essas aplicações incluem: detecção de código duplicado e análise semântica de fragmentos de código [20]; paralelização de estruturas de código por meio de esquemas recursivos [14], visando melhorar a manutenibilidade e eficiência computacional; detecção de erros e reparo automático de programas, reduzindo custos de manutenção através de ferramentas como Staticfixer [32], Fixie [45], REVISAR [44], REX [37] e Getafix do Facebook [13]; compressão por aprendizado de bibliotecas (*Library Learning*), em que a AU permite abstrair automaticamente padrões recorrentes de código em funções reutilizáveis via Babble [22]; Programação por Exemplos (PBE) [30], que usa AU sintática para generalizar especificações de programas a partir de exemplos de entrada; e Programação Lógica Indutiva (ILP) [29], que gera programas por meio da generalização de exemplos de treinamento e de conhecimento prévio.

Embora a anti-unificação sintática seja suficiente para aplicações básicas, cenários industriais avançados — como análise de código, reparo automático e aprendizado de

bibliotecas — exigem raciocínio módulo teorias equacionais para capturar equivalências semânticas. Nesses contextos, a AU pode admitir múltiplas generalizações menos gerais, o que leva ao cálculo de um conjunto mínimo completo de generalizações menos gerais (o conjunto de todos os lggs). A existência e o tamanho desse conjunto permitem classificar as teorias equacionais como *unitárias* (uma única solução para cada par de termos), *finitárias* (um número finito de soluções para cada par, havendo pelo menos um par com mais de uma solução), *infinitárias* (algum par admite infinitas soluções) ou *nulárias* (para algum par, nenhum conjunto mínimo completo de lggs pode ser computado). Essas classificações motivaram pesquisas extensas em AU sobre uma ampla variedade de estruturas matemáticas e computacionais.

Essas investigações abrangem diversos domínios, incluindo termos apresentados como grafos [19], termos de ordem superior [15,18,24,33], linguagens *variádicas* (i.e., linguagens que admitem funções de aridade variável) [16,34], termos nominais [17,42,43], aproximações modulares [10,35,36,38] e teorias equacionais de primeira ordem [1,2,21,23,25,26, 39,41].

Avanços significativos foram alcançados na AU módulo teorias equacionais. Inicialmente formulada para linguagens sintáticas de primeira ordem por Plotkin e Reynolds [39,41], a AU foi posteriormente estendida para teorias com axiomas que descrevem propriedades algébricas e suas combinações. Por exemplo, Burghardt [21] tratou a AU módulo uma teoria equacional arbitrária usando gramáticas. Maria Alpuente et al. [1,2] analisaram a AU em teorias com operadores comutativos (C) e associativos (A) e suas combinações, demonstrando que essas teorias são finitárias. Por outro lado, Kutsia e Cerna [26] determinaram que teorias contendo dois ou mais operadores unitários são de tipo nulário. Eles também provaram a nularidade para as seguintes combinações: $(\mathrm{AU})^{>1}$ $(\mathrm{CU})^{>1}$, $(\mathrm{ACU})^{1}$ e (AU)(CU). Em [25], apresentaram um algoritmo que computa um conjunto infinito de soluções para as teorias (I), (AI) e (CI). Além disso, Cerna [23] mostrou a nularidade para semianéis e para as combinações $(\mathrm{UI})^{>1}$, $(\mathrm{AUI})^{>1}$, $(\mathrm{CUI})^{>1}$,$(\mathrm{ACUI})^{>1}$. Aqui, a notação $(\cdot)^{1}$ denota teorias com exatamente um operador do tipo considerado, enquanto $(\cdot)^{>1}$ denota teorias com mais de um operador desse tipo.

Esta tese trata da anti-unificação em **teorias absortivas**, nas quais certos símbolos de operação satisfazem os axiomas $f(\varepsilon_f, x) \approx \varepsilon_f$ and $f(x, \varepsilon_f) \approx \varepsilon_f$, para uma constante absorvente relativa ε_f. A absorção é uma das propriedades fundamentais presentes em diversas estruturas algébricas, incluindo semigrupos, monoides, semianéis e álgebras booleanas. Exemplos concretos incluem: os inteiros com o máximo divisor comum (mdc) e o número 1 como constante absorvente; o conjunto das partes de um dado conjunto com a interseção ($\cap$) e o conjunto vazio; os inteiros com a multiplicação e o zero; e os conjuntos de predicados com a conjunção e o valor falso.

As propriedades absortivas induzem comportamentos semânticos complexos, pois um único termo pode possuir infinitas representações equivalentes. Isso torna o cálculo de conjuntos mínimos completos de generalizações menos gerais uma tarefa altamente não trivial, exigindo o controle rigoroso de espaços de soluções potencialmente infinitos. Determinar o tipo de um problema de AU — unitário, finitário, infinitário ou nulário— requer raciocínio e técnicas de prova rigorosas. Consequentemente, a anti-unificação em teorias absortivas apresenta desafios teóricos profundos e relevância prática.

Nesta tese, desenvolvemos os fundamentos formais da AU em teorias absortivas e propomos quatro algoritmos corretos que computam generalizações para: (i) teorias com apenas símbolos absortivos; (ii) teorias que combinam símbolos absortivos e comutativos, e suas variantes lineares, nas quais o objetivo é encontrar generalizações em que cada variável ocorre no máximo uma vez. Em particular, os algoritmos para as variantes lineares são demonstrados completos, enquanto, no caso puramente absortivo, o algoritmo é capaz de produzir conjuntos mínimos completos de generalizações.

Contribuições As contribuições deste trabalho, que serão apresentadas em detalhe na descrição dos capítulos, são listadas a seguir.

- Desenvolvemos um **algoritmo correto para o problema de anti-unificação absortiva**, introduzindo um procedimento baseado em regras, $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}$, definido sobre *configurações*, uma estrutura para codificar problemas de anti-unificação. O algoritmo opera sobre pares de termos *ground* (termos que não contêm variáveis), utilizando uma bijeção entre variáveis e constantes que permite computar generalizações e mapeá-las de volta sem alterar as soluções. Para fortalecer o arcabouço, também definimos *conjuntos de abstração* e *substituições de abstração*, que permitem o cálculo de generalizações mais específicas do que as obtidas apenas por $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}$. A abordagem é sustentada por provas de terminação e por um lema de preservação para configurações, garantindo a correção do processo. Em conjunto, esses resultados estabelecem uma base rigorosa para a anti-unificação absortiva.
- Realizamos uma análise crítica da **completude na anti-unificação absortiva** apresentada em [9], identificando falhas e limitações em sua formulação. Mostramos que a combinação de $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}$ com substituições de abstração é insuficiente, pois generalizações essenciais podem ser omitidas. Para resolver isso, propusemos um método de refinamento que acreditamos poder completar o algoritmo. Embora tenhamos formulado a declaração de completude que desejamos alcançar e apresentado um esboço da prova, alguns pontos desse esboço ainda precisam ser finalizados. Esse refinamento computa generalizações adicionais que são essenciais para atingir a completude.

- Introduzimos um algoritmo para o **problema de anti-unificação absortiva linear**, no qual cada variável em uma generalização ocorre, no máximo, uma vez. Propõe-se o algoritmo $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}{}^{\ell}}$, que é correto. Além disso, utilizamos a *substituição de abstração linear*, que gera generalizações lineares mais específicas do que as produzidas apenas por $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}{}^{\ell}}$. Juntos, $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}{}^{\ell}}$ e a substituição de abstração linear atingem a completude, permitindo o cálculo do conjunto mínimo completo de generalizações absortivas lineares para qualquer par de termos. Esse resultado representa o primeiro tratamento sistemático da variante absortiva linear e demonstra que a restrição de linearidade permite obter conjuntos de generalizações minimais de forma eficiente.
- Estendemos o estudo para a **anti-unificação absortiva comutativa**, abrangendo teorias que incluem símbolos absortivos, comutativos e absortivo-comutativos. Nesse contexto, propomos dois algoritmos: $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$, que computa generalizações na teoria combinada, e $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}{}^{\ell}}$, que computa generalizações lineares. Provamos a terminação e a correção de $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$e estabelecemos a correção e a completude de $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}{}^{\ell}}$. Essa é a primeira formulação sistemática da anti-unificação absortiva comutativa, ampliando significativamente a aplicabilidade do arcabouço a teorias equacionais mais expressivas.
- Caracterizamos o **tipo do problema de anti-unificação** em teorias absortivas. No caso geral (não linear), o problema é pelo menos *infinitário*, devido ao número potencialmente infinito de generalizações incomparáveis. Em contraste, para as variantes lineares, tanto no caso puramente absortivo quanto no absortivo-comutativo, o problema é *finitário*, o que significa que existe um conjunto finito completo de generalizações.

Organização Este trabalho está estruturado em seis capítulos. **O Capítulo 1** introduz os principais conceitos, notações e trabalhos relacionados à anti-unificação em teorias de primeira ordem. **O Capítulo 2** apresenta um algoritmo correto para a anti-unificação absortiva, $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}$, incluindo as noções de configurações e substituições de abstração, com provas de terminação e de correção. **O Capítulo 3** aborda os desafios da completude, identifica limitações em trabalhos anteriores e propõe refinamentos rumo a uma formulação corrigida de completude. **O Capítulo 4** trata da variante linear, introduzindo $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}{}^{\ell}}$ e a substituição de abstração linear. Juntos, eles alcançam a completude e computam o conjunto mínimo completo de generalizações lineares absortivas. **O Capítulo 5** estende o arcabouço para a teoria absortiva comutativa, introduzindo, respectivamente, $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$ e $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}{}^{\ell}}$, que fornecem algoritmos para generalizações e generalizações lineares, com garantias de correção e completude lineares. Finalmente, **o Capítulo 6** resume

os resultados e discute direções futuras, incluindo a finalização das provas de completude e a exploração de combinações com operadores associativos.

Palavras-chave: Anti-unificação, Generalização, Teoria Equacional, Teoria Absortiva.

Abstract

Anti-unification is the problem of finding an expression that captures the common structure of two given expressions. The resulting expression is called a *generalization* of them. The development of anti-unification procedures, applied for computing generalizations of expressions, has become increasingly important in applications such as program analysis, clone detection, and symbolic reasoning. While early work focused on *syntactic* anti-unification, modern applications often require reasoning modulo equational theories. This thesis focuses on anti-unification in absorptive theories, where certain "absorptive" operation symbols satisfy the axioms $f(\varepsilon_f, x) \approx \varepsilon_f$ and $f(x, \varepsilon_f) \approx \varepsilon_f$ for a relative "absorbing" constant ε_f; e.g., multiplication and zero, conjunction and false, intersection and empty set.

This thesis presents foundational tools and a sound algorithm for absorptive theories, defined through a set of inference rules that transform configurations that are structures encoding anti-unification problems and partial solutions. Preservation properties of the configurations transformed by the inference rules are established, ensuring the soundness of the algorithm. A key contribution of this work is the refinement of the completeness analysis of an algorithm presented at the *12th International Joint Conference on Automated Reasoning* (2024). Additional generalizations not previously considered are identified, the inference rules are enhanced, and it is shown how the corresponding solutions can be computed from the final configurations. This thesis also addresses theories that may include absorptive, commutative, and absorptive-commutative function symbols, for short, absorptive commutative theories. A new set of inference rules is presented, and the resulting algorithm is proved sound. This extends the applicability of the approach to more expressive term structures encountered in practice. Additionally, the restricted variant of anti-unification to linear generalizations, where each variable appears at most once in the target solutions, is addressed. The algorithm is adapted for the linear absorptive commutative invariant, showing its soundness and completeness. In the linear pure absorptive variant, the proposed algorithm computes a minimal and complete set of generalizations.

Keywords: Anti-unification, Generalization, Equational Theory, Absorptive Theory.

Contents

List of Tables

Introduction

Anti-unification (AU) or Generalization is a fundamental problem in inductive reasoning that involves finding a symbolic expression capturing the common structure between two given symbolic expressions. This problem can be informally seen as dual to unification; while unification determines how to identify two expressions in the most general form, AU seeks the most specific expression that contains the shared similarities between two expressions; in literature, these most specific expressions are known as *least general generalizations* (`lgg`s).

For instance, in the *syntactic* setting—where no algebraic properties such as commutativity or associativity are considered—let f be a binary function symbol, a, b, c be constants, and x, y be variables. Then x and $f(w, y)$ are generalizations of the expressions $f(f(a, a), c)$ and $f(f(b, b), c)$. Moreover, the expression $f(f(z, z), c)$ is the most specific generalization possible, containing the most common structure of the expressions above.

Interest in AU has grown substantially due to its diverse industry and academic applications across computer science, such as Machine Learning, Automated Reasoning, and Formal Logic. These applications include: detecting duplicate code and performing semantic code analysis for code fragments [20]; parallelizing code structures through recursion schemes [14] to enhance software maintainability and computational efficiency; bug detection and automated program repair, which reduces maintenance costs through tools like *Staticfixer* [32], *Fixie* [45], *REVISAR* [44], *REX* [37], and Facebook's *Getafix* [13]; Library Learning Compression, where AU enables automated abstraction of recurrent code patterns into reusable library functions via *Babble* [22]; Programming By Examples (PBE) [30], which uses syntactic AU to generalize program specifications from input examples; and Inductive Logic Programming (ILP) [29], which generates programs through generalization of training examples and background knowledge.

While syntactic anti-unification is sufficient for basic applications, advanced industrial scenarios—such as code analysis, program repair, and library learning—require reasoning modulo equational theories to capture semantic equivalences. In such contexts, AU admits multiple minimal generalizations, which leads to the computation of a *minimal complete set of generalizations* (the set of all `lgg`s). The existence and size of this set allows

us to classify equational theories as *unitary* (a single solution for every pair of terms), *finitary* (a finite set of solutions for every pair of terms, with at least one pair admitting more than one solution), *infinitary* (some pair of terms admits infinitely many solutions), or *nullary* (for some pair of terms, no minimal complete set can be computed). These classifications motivated extensive research into AU across a wide range of mathematical and computational structures.

Such investigations span diverse domains including first-order theories [1, 2, 21, 23, 25, 26, 39, 41], terms-graphs [19], higher-order terms [15, 18, 24, 33], unranked (variadic) languages [16, 34], nominal terms [17, 42, 43], and modulo approximations [10, 35, 36, 38].

Significant developments have been made in AU modulo equational theories. Initially formulated for first-order syntactic languages by Plotkin and Reynolds [39, 41], AU has been extended to theories with axioms describing algebraic properties and their combinations. For instance, Burghardt [21] considered AU modulo an arbitrary equational theory using grammars. Alpuente et al. [1,2] analyzed AU in theories with commutative (C) and associative (A) operators and their combinations, determining that these theories were finitary. On the other hand, Kutsia and Cerna [26] determined that the type for theories including two or more unital operators is nullary. They also prove the nullarity for the next combinations of theories: $(\text{AU})^{>1}$ $(\text{CU})^{>1}$, $(\text{ACU})^{1}$, and (AU)(CU) and in [25] they provides an algorithm that computes an infinite set of solutions for theories (I), (AI), and (CI). Additionally, Cerna [23] showed the nullarity for semirings and the combinations $(\text{UI})^{>1}$, $(\text{AUI})^{>1}$, $(\text{CUI})^{>1}$,$(\text{ACUI})^{>1}$. Here, the notation $(\cdot)^1$ denotes theories with exactly one operator of the given type, and $(\cdot)^{>1}$ denotes theories with more than one operator of that type.

This thesis focuses on anti-unification in absorptive theories, where certain absorptive operation symbols satisfy the axioms $f(\varepsilon_f, x) \approx \varepsilon_f$ and $f(x, \varepsilon_f) \approx \varepsilon_f$ for a relative absorbing constant ε_f. Absorptive is one of the fundamental properties used in various algebraic structures, including semi-groups, monoids, semi-rings, and Boolean algebras. Concrete examples include integers with the greatest common divisor (*gcd*) and *one* as the absorbing constant; the powerset of a given set with intersection ($\cap$) and $\emptyset$; integers with multiplication and zero; and sets of predicates with conjunction and *false*.

The absorptive properties induce complex semantic behaviors, as a single term may have infinitely many equivalent representations. This makes computing complete sets of least general generalizations highly non-trivial, requiring careful management of potentially infinite solution spaces. Determining the type of an AU problem—whether unitary, finitary, infinitary, or nullary—demands rigorous reasoning and proof techniques. Consequently, anti-unification in absorptive theories presents both rich theoretical challenges and practical relevance. In this thesis, we develop the formal foundations of AU in ab-

sorptive theories and propose four sound algorithms that compute generalizations for theories with only absorptive symbols, for theories combining absorptive and commutative symbols, and for their respective linear variants, in which the focus is on finding generalizations where each variable occurs at most once. In particular, the algorithms for the linear variants are shown to be complete, while in the pure absorptive setting, the algorithm is capable of producing minimal complete sets of generalizations.

Contributions

The contributions of this work, which will be introduced in detail in the description of the chapters, are listed below.

- We develop a **sound algorithm for absorptive anti-unification problem** by introducing a rule-based procedure, $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$, defined over configurations, a structure for encoding anti-unification problems. The algorithm operates on pairs of ground terms, using a bijection between variables and constants that allows computing generalizations and mapping them back without altering the solutions. To strengthen the framework, we also define abstraction sets and abstraction substitutions, which enable the computation of more specific generalizations than those obtained by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ alone. The approach is supported by proofs of termination and a preservation lemma for configurations, ensuring the soundness of the process. Altogether, this establishes a rigorous foundation for absorptive anti-unification.
- We critically analyzed the **completeness in absorptive anti-unification** approach presented in [9], identifying flaws and limitations in its formulation. We show that the combination $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ with abstraction substitutions is insufficient, as essential generalizations may be missed. To address this, we proposed a refinement method that is believed to complete the algorithm. Although we formulate the statement of completeness that we want to achieve and provide a sketch of the proof, some points in the sketch remain to be finalized. This refinement computes additional generalizations that are essential to achieving completeness.
- We introduce an algorithm for **linear absorptive anti-unification problem**, in which each variable in a generalization occurs at most once. An algorithm, $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$, is proposed, which is sound. In addition, we make use of the linear abstraction substitution, which generates more specific linear generalizations than those generated by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$. Together, $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$ and the linear abstraction substitution achieve completeness, allowing the computation of the minimal complete set of linear absorptive generalizations for any pair of terms. This marks the first systematic treat-

ment of the linear absorptive variant and demonstrates that the linear restriction enables the derivation of efficient and minimal sets of generalizations.

- We extend the study to **absorptive commutative anti-unification**, covering theories that include absorptive, commutative, and absorptive-commutative symbols. Within this setting, we propose two algorithms: $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$, which computes generalizations in the combined theory, and $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}^\ell}}$, which computes linear generalizations. We prove termination and soundness for $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$, and establish soundness and completeness for $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}^\ell}}$. This provides the first systematic account of absorptive commutative anti-unification, significantly broadening the applicability of the framework to more expressive equational theories.
- We characterize the **type of the anti-unification problem** in absorptive theories. In the general (non-linear) case, the problem is at least *infinitary*, due to the potentially infinite number of incomparable generalizations. In contrast, for the linear variants, both purely absorptive and absorptive commutative, the problem is *finitary*, meaning that a finite complete set of generalizations exists.

Organization

This work is structured into six chapters. **Chapter 1** introduces the main concepts, notation, and related work on anti-unification for first-order theories. **Chapter 2** presents a sound algorithm for absorptive anti-unification, $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}$, including configurations and abstraction substitutions, with proofs of termination and soundness. **Chapter 3** addresses the challenges of completeness, identifies limitations in prior work, and proposes refinements toward a corrected completeness formulation. **Chapter 4** treats the linear variant, introducing $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^\ell}}$ and the linear abstraction substitution. Together, they achieve completeness and compute the minimal complete set of linear absorptive generalizations. **Chapter 5** extends the framework to absorptive commutative theory, and introduces, respectively, $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$ and $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}^\ell}}$, providing algorithms for generalizations and linear generalizations with soundness and linear completeness guarantees. Finally, **Chapter 6** summarizes the results and discusses future directions, including completing the completeness proofs and exploring combinations with associative operators.

Chapter 1

Background

This chapter introduces the main concepts, notation, and formal definitions used throughout the thesis. It provides a comprehensive overview of anti-unification in first-order theories, presenting different types of anti-unification problems, their classifications (unitary, finitary, infinitary, nullary), and foundational results in the literature. In addition, this chapter surveys existing algorithms and approaches for computing least general generalizations in syntactic and equational settings, highlighting their strengths and limitations.

1.1 First-Order Syntax

The table above includes the conventions used for the main objects.

Letters	Use
a, b, c, d	constant
e, f, g, h	function
i, j, k, l	index
m, n	natural number
o, p, q	position
r, s, t, u, v	term
w, x, y, z	variable
$\gamma, \eta, \theta, \mu, \rho, \sigma, \tau$	substitution

Table 1.1: Conventions

At first, we introduce the language and the main notions of universal algebra in a syntactic level and some of the semantic notions that are the base of the whole document, many of them are taken from Baader and Nipkow [12].

Definition 1.1.1 (Signature)**.** A *signature* $\mathcal{F}$ is a set of *function symbols*, where each $f \in \mathcal{F}$ is associated with a non-negative integer n, the *arity* of f. The function symbols with arity 0 are called *constant symbols*.

Definition 1.1.2 (Term)**.** Let $\mathcal{F}$ be a signature and $\mathcal{V}$ be a countable set of variables such that $\mathcal{F} \cap \mathcal{V} = \emptyset$. The set $\mathcal{T}(\mathcal{F}, \mathcal{V})$ of all the $\mathcal{F}$-terms over $\mathcal{V}$ is inductively defined as

- $\mathcal{V} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$ (i.e., every variable is a term),
- for all $n \geq 0$, all f with arity n, and all $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, we have that $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$.

Definition 1.1.3 (Head, Depth)**.** The next operators are defined over $\mathcal{T}(\mathcal{F}, \mathcal{V})$:

1. The *head* of a term t is defined inductively as

$$\begin{aligned} \mathit{head}(x) &:= x, \\ \mathit{head}(f(t_1, \ldots, t_n)) &:= f, \end{aligned}$$

   for all $x \in \mathcal{V}$ and $n \geq 0$.

2. The *depth* of a term t is defined inductively as

$$\begin{aligned} \mathit{depth}(x) &:= 0, \\ \mathit{depth}(a) &:= 0, \\ \mathit{depth}(f(t_1, \ldots, t_n)) &:= 1 + \max\{\mathit{depth}(t_1), \ldots, \mathit{depth}(t_n)\}, \end{aligned}$$

   for all $x \in \mathcal{V}$, a constant symbol and $n \geq 1$.

Definition 1.1.4 (Positions)**.** Let s and t be terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$.

1. The *set of positions* of the term t is a set $\mathit{pos}(t)$ of strings over the alphabet of positive integers, which is inductively defined as follows:
   - If $t = x \in \mathcal{V}$, then $\mathit{pos}(t) := \{\epsilon\}$, where ϵ denotes the empty string.
   - If $t = f(t_1, \ldots, t_n)$, for $n \geq 0$, then

$$\mathit{pos}(t) = \{\epsilon\} \cup \bigcup_{i=1}^{n} \{i.p \mid p \in \mathit{pos}(t_i)\}$$

2. For $p \in \mathit{pos}(t)$, the *subterm* of t at position p, denoted by $t|_p$, is defined by induction on the length of p:

$$\begin{aligned} t|_\epsilon &:= t, \\ f(t_1, \ldots, t_n)|_{i.q} &= t_i|_q \end{aligned}$$

   The *set of subterms of* t is defined as $\mathit{subt}(t) = \{t|_p \mid p \in \mathit{pos}(t)\}$ and the set of *proper subterms of a term* t is defined as $\mathit{subtp}(t) = \mathit{subt}(t) - \{t\}$.

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

$$\begin{aligned} t[s]_\epsilon &:= s, \\ f(t_1, \ldots, t_n)[s]_{i.q} &:= f(t_1, \ldots, t_i[s]_q, \ldots, t_n). \end{aligned}$$

4. The size $size(s)$ of a term s is the cardinality of $pos(s)$.

5. By $var(s)$ we denote the set of variables occurring in s, i.e.,

$$var(s) := \{x \in \mathcal{V} \mid \text{there exists } p \in pos(s) \text{ such that } s|_p = x\}.$$

Note that, for $p = i.q$, $p \in pos(t)$ implies that t is of the form $t = f(t_1, \ldots, t_n)$ with $i \leq n$, and $q \in pos(t_i)$.

Definition 1.1.5 (Position Ordering)**.** The *position ordering* is the prefix ordering over sequences of positions; i.e., for a given term t, and $p, q \in pos(t)$, it is defined as $p \sqsubset q$ if there exists a non-empty p' such that $p.p' = q$. It is a partial order on $pos(t)$. We say that the positions p, q are *parallel* $p||q$ if p and q are incomparable with respect to $\sqsubseteq$.

Let t be a term and $p, q \in pos(t)$. If $p \sqsubseteq q$ then $t|_q$ is a subterm of $t|_p$ (i.e., $t|_q \in subt(t|_p)$); if $p \sqsubset q$, it is a proper subterm of $t|_p$ (i.e., $t|_q \in subtp(t|_p)$).

Definition 1.1.6 (Ground Terms)**.** A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ is called *ground* if $var(t) = \emptyset$. The set of all ground terms is denoted as $\mathcal{T}(\mathcal{F}, \emptyset)$ or just $\mathcal{T}(\mathcal{F})$.

Definition 1.1.7 (Linear Term)**.** A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ is called *linear* if each $x \in var(t)$ occurs at most once in t.

Definition 1.1.8 (Binding)**.** A *binding* is a pair (x, t), where $x \in \mathcal{V}$ and $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and denoted as $\{x \mapsto t\}$.

Definition 1.1.9 (Substitution)**.** A *substitution* is a function $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $\sigma(x) \neq x$ for only finitely many xs. The following definitions are related to substitutions:

1. The (finite) set of variables that σ does not map to themselves is called the *domain* of σ

$$dom(\sigma) := \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$$

if $dom(\sigma) = \{x_1, \ldots, x_n\}$, then σ can be described as a set of *bindings* as

$$\sigma = \{x_1 \mapsto \sigma(x_1), \ldots, x_n \mapsto \sigma(x_n)\}$$

2. The *range* of σ is the set $ran(\sigma) := \{\sigma(x) \,|\, x \in dom(\sigma)\}$ and the *variables in the range* of σ consist of the variables occurring in $ran(\sigma)$

$$\mathcal{V}ran(\sigma) := \bigcup_{x \in dom(x)} var(\sigma(x)).$$

Every substitution σ can be *extended* to a mapping $\hat{\sigma} : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ as follows

$$\begin{aligned} \hat{\sigma}(x) &:= \sigma(x), \text{ if } x \in \mathcal{V}, \\ \hat{\sigma}(f(t_1, \ldots, t_n)) &:= f(\hat{\sigma}(t_1), \ldots, \hat{\sigma}(t_n)), \end{aligned}$$

for $n \geq 0$. To simplify the notation, σ will be used to denote its extension $\hat{\sigma}$. In addition, we write $t\sigma$ instead of $\sigma(t)$. The *identity substitution*, the substitution that maps every variable to itself, is denoted by *id*. A substitution is called *ground* if all terms in its range are ground.

Definition 1.1.10 (Renaming)**.** A *renaming* is a substitution σ such that, restricted to its domain, is bijective and $ran(\sigma) \subseteq \mathcal{V}$. A renaming σ is called *disjoint* if $ran(\sigma) \cap dom(\sigma) = \emptyset$.

Example 1.1.11. The substitutions $\sigma = \{x \mapsto w, y \mapsto z\}$ and $\rho = \{x \mapsto y, y \mapsto x\}$ are examples of renaming; the first substitution is disjoint, but the second one is not. $\diamond$

Definition 1.1.12 (Substitution Composition)**.** The *composition* $\sigma\tau$ of two substitutions σ and τ, denoted by juxtaposition, is defined as $x\sigma\tau := (x\sigma)\tau$. Notice that $\sigma\tau$ is a mapping of $\mathcal{V}$ into $\mathcal{T}(\mathcal{F}, \mathcal{V})$, and is a substitution since $x\sigma\tau = x$ holds for $x \in \mathcal{V} - (dom(\sigma) \cup dom(\tau))$.

Notice that the set representation in bindings of the composition between the substitutions $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ and $\tau = \{y_1 \mapsto s_1, \ldots, y_m \mapsto s_m\}$ can be obtained from $\{x_1 \mapsto t_1\tau, \ldots, x_n \mapsto t_n\tau, y_1 \mapsto s_1, \ldots, y_m \mapsto s_m\}$ by removing all the bindings $\{y_i \mapsto s_i\}$'s with $y_i \in \{x_1, \ldots, x_n\}$ and $\{x_i \mapsto t_i\tau\}$'s with $x_i = t_i\tau$.

Definition 1.1.13 (Restriction of a Substitution)**.** The *restriction of a substitution* σ respect to a finite set of variables V, denoted by $\sigma|_V$ is a substitution defined as $\sigma|_V(x) := x\sigma$ for $x \in V$ and $\sigma|_V(x) = x$ otherwise.

Definition 1.1.14 (Linear Substitution)**.** A substitution σ is called *linear* if each term t in its range, $t \in ran(\sigma)$, is a linear term.

1.2 Algebra, Congruence and Equational Theories

Definition 1.2.1 ($\mathcal{F}$-algebra)**.** Let $\mathcal{F}$ be a signature. An *$\mathcal{F}$-algebra* $\mathcal{A}$ consist of:

- a carrier set (domain) A, and
- a mapping that associates with each function symbol f of arity n a function $f^{\mathcal{A}} : A^n \to A$ (for all $n \geq 0$).

Definition 1.2.2 ($\mathcal{F}$-homomorphism)**.** Let $\mathcal{F}$ be a signature, and let $\mathcal{A}$ and $\mathcal{B}$ be $\mathcal{F}$-algebras with carriers A and B, and mappings $f^{\mathcal{A}}$ and $f^{\mathcal{B}}$, respectively. A *$\mathcal{F}$-homomorphism* from $\mathcal{A}$ to $\mathcal{B}$ is a mapping $\phi : A \to B$ such that for all $n \geq 0$, f of arity n and $a_1, \ldots, a_n \in A$ we have $\phi(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(\phi(a_1), \ldots, \phi(a_n))$.

Definition 1.2.3 (Congruence)**.** Let $\mathcal{A}$ be a $\mathcal{F}$-algebra. An equivalence relation $\equiv$ on its carrier A is called *congruence* on $\mathcal{A}$ if it is compatible with the interpretation of all symbols of $\mathcal{F}$, i.e., for all $n \geq 0$, all f of arity n and all $a_1 \equiv b_1, \ldots, a_n \equiv b_n$ in A we have

$$f^{\mathcal{A}}(a_1, \ldots, a_n) \equiv f^{\mathcal{A}}(b_1, \ldots, b_n).$$

The *quotient algebra* $\mathcal{A}/\equiv$ has as carrier the set of equivalent classes $[a]_{\equiv} := \{b \in A \mid a \equiv b\}$ and interprets $f^{\mathcal{A}/\equiv}([a_1]_{\equiv}, \ldots, [a_n]_{\equiv}) := [f^{\mathcal{A}}(a_1, \ldots, a_n)]_{\equiv}$.

Definition 1.2.4 (Equation)**.** Let $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, an *equation* is an unordered pair $\{s, t\}$ denoted as $s \approx t$.

Definition 1.2.5. An equation $s \approx t$ *holds* in the $\mathcal{F}$-algebra $\mathcal{A}$ ($\mathcal{A} \models s \approx t$) if for all $\mathcal{F}$-homomorphism ϕ from $\mathcal{T}(\mathcal{F}, \mathcal{V})$ to $\mathcal{A}$, we have $\phi(s) = \phi(t)$.

Definition 1.2.6. Let $\mathcal{F}$ be a signature and E be a set of equations.

1. The $\mathcal{F}$-algebra $\mathcal{A}$ is a *model* of E ($\mathcal{A} \models E$) if every equation of E holds in $\mathcal{A}$.
2. The class of all models of E is called $\mathcal{F}$-variety defined by E. it is denoted $\mathfrak{V}(E)$.

Definition 1.2.7 (Semantic Consequence)**.** Let E be a set of equations, the equation $s \approx t$ is a *semantic consequence* of E ($E \models s \approx t$) if it holds in all models of E, i.e., for all $\mathcal{A} \in \mathfrak{V}(E)$ we have that $s \approx t$ holds in $\mathcal{A}$.

Definition 1.2.8 (Equational Theory)**.** Let E be a set of equations, the relation

$$\approx_E := \{(s, t) \in \mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathcal{T}(\mathcal{F}, \mathcal{V}) \mid E \models s \approx t\}$$

is called the *equational theory* induced by E or E-theory. If $(s, t) \in \approx_E$ we denoted as $s \approx_E t$ and we say s and t are E-equivalent. The equations in E are called *axioms* of the E-theory. A symbol f is called an E-symbol if it occurs in the axioms of the E-theory.

Obviously $s \approx t \in E$ implies that $s \approx_E t$. Additionally, the relation $\approx_E$ is a congruence relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$. Thus, we can build the quotient algebra $\mathcal{T}(\mathcal{F}, \mathcal{V})/\approx_E$.

Example 1.2.9. Let f be a binary function symbol and a, b be constants. The set of axioms $\{f(x,y) \approx f(y,x)\}$ induces the *commutative theory* C. Notice that f is a C-symbol, the terms $f(a, f(b,a))$ and $f(f(b,a),a)$ are C-equivalent, and the equivalent class of $f(a, f(b,a))$ modulo C is the set

$$[f(a,f(b,a))]_{\approx_C} = \{f(a,f(b,a)), f(a,f(a,b)), f(f(b,a),a), f(f(a,b),a)\}.$$

$\diamond$

Definition 1.2.10 (Absorptive Theories)**.** Consider the theory with a nonempty finite set of associated pairs of a binary function f_k and a constant ε_{f_k} satisfying the axioms

$$\{f_k(\varepsilon_{f_k}, x) \approx \varepsilon_{f_k}, f_k(x, \varepsilon_{f_k}) \approx \varepsilon_{f_k}\}, \text{ for } 1 \leq k \leq n.$$

Such a theory is called an *absorptive theory*, for brevity, $\mathfrak{a}$-*theory*. f_k and ε_{f_k} are called *absorptive symbol* and *absorbing constant*, respectively. Pairs (f_k, ε_{f_k}) are called *related absorptive symbols*, for each $1 \leq k \leq n$. We denote by $\mathtt{Abs}$ the set of *absorptive symbols* $\{f_1, \ldots, f_n\}$.

Definition 1.2.11 (C, A, I, U, Ui, $\mathfrak{a}$, D Axioms and Theories)**.** The theories considered in this work are theories specified by signatures that, in addition to non-interpreted symbols, include commutative (C), associative (A), idempotent (I), unital (U), unital with inverses (Ui), and distributive (D) symbols:

- (C) A *commutative theory* is a theory in which there is a function symbol f satisfying the commutative axiom $\{f(x,y) \approx f(y,x)\}$.
- (A) An *associative theory* is a theory in which there is a function symbol f satisfying the associative axiom $\{f(f(x,y),z)) \approx f(x,f(y,z))\}$.
- (I) An *idempotent theory* is a theory in which there is a function symbol f satisfying the idempotent axiom $\{f(x,x) \approx x\}$.
- (U) A *unital theory* is a theory in which there is a function symbol f and its related constant e_f satisfying the unital axioms $\{f(x,e_f) \approx x, f(e_f,x) \approx x\}$.
- (Ui) A *unital with inverses theory* is a theory in which there is a function symbol f, its related constant e_f and the related unary function $\mathtt{i}$ satisfying the axioms $\{f(x,e_f) \approx e_f, f(e_f,x) \approx e_f, f(i_f(x),x) \approx e_f, f(x,i_f(x)) \approx e_f\}$.
- (D) A *distributive theory* is a theory in which there are two function symbols f and g satisfying the distributive of symbol f over the symbol g axioms below.

$$\{f(x,g(y,z)) \approx g(f(x,y),f(x,z)), f(g(x,y),z) \approx g(f(x,z),f(y,z))\}.$$

Notation. To simplify the notation for theories that satisfy multiple algebraic properties, such as those in Definition 1.2.11, this work adopts the notational convention used in [27]. When we want to emphasize the symbols of the E-theory, we write $E(S)$, where S is a set of tuples of E-symbols. These tuples of E-symbols are sorted, the first element of each tuple is the head symbol of the related axioms and is called the main symbol, while the other elements of the tuple are the related symbols or special constants of the respective axioms. If we work with combinations of theories $E_1, \ldots, E_n$ with their respective sets of tuples of E_i-symbols $(1 \leq i \leq n)$ $S_1, \ldots, S_n$, such that S_i and S_j (for $i \neq j$, $1 \leq i, j \leq n$) do not share a tuple with the same main symbol, we use the convention $(E_1(S_1)) \cdots (E_n(S_n))$. Moreover, if $S_1, \ldots, S_n$ share all the same main symbols in their tuples, we write $E_1 \cdots E_n(S)$, where each tuple in S consists of the main symbol and its respective related symbols in the respective order. Some of the analyses of theories and their properties rely on the number of symbols that fulfill the corresponding equational axioms. To express this, we use a specific notation: for a given theory E, we write E^1 to denote $E(S)$ when S includes a single element. Similarly, $E^{>1}$ represents $E(S)$ when S includes a finite set with at least two tuples. When we refer to E without additional notation, we mean the equational theory $E(S)$, where S is a finite set that may have one or more elements.

Example 1.2.12. We illustrate this notation with the theories below.

- $\mathrm{AC}(\{(f),(g)\})$ denotes a theory with two AC-symbols f and g.
- $(\mathrm{AC}(\{(f)\}))(\mathrm{C}(\{(g)\}))$ denotes a theory with an AC-symbol f, and a C-symbol g.
- $\mathrm{AU}(\{(f, e_f)\})$ describe the monoids.
- $\mathrm{AU}\mathfrak{a}(\{(f, e_f, \varepsilon_f)\})$ describes the binoids.
- $\mathrm{ACU}\mathfrak{i}(\{(f, e_f, i_f)\})$ describes the Abelian groups.
- $(\mathrm{A}\mathfrak{a}\mathrm{UD}(\{(f, \varepsilon_f, e_f, g)\}))(\mathrm{ACU}\{(g, \varepsilon_f)\})$ describes the semiring theory (usually written as $(\mathrm{A}\mathfrak{a}\mathrm{UD}(\{(\times, 0, 1, +)\}))(\mathrm{ACU}\{(+, 0)\}))$. For short, we denote this theory by $\mathrm{SRNG}(\{f, e_f, g, \varepsilon_f\})$ and the commutative semirings (adding the commutative axiom for the symbol f) by $\mathrm{CSRNG}(\{f, e_f, g, \varepsilon_f\})$.
- U^1 describes a theory with only one U-symbol and its respective unit and $\mathrm{U}^{>1}$ or simply U describe a theory with at least two U-symbols and their respective units.
- $(\mathfrak{a})(\mathfrak{a}\mathrm{C})(\mathrm{C})$ abbreviates a theory with at least one $\mathfrak{a}$ tuple of symbols, at least one $\mathfrak{a}\mathrm{C}$ tuple of symbols, and at least one C-symbol. We call this theory the absorptive commutative theory.

The focus of this work is the study of anti-unification in the theories $\mathfrak{a}$ and $(\mathfrak{a})(\mathfrak{a}\mathrm{C})(\mathrm{C})$. $\diamond$

Definition 1.2.13 (Reduction Relation $\rightarrow^E$)**.** Let $E \subseteq \mathcal{T}(\mathcal{F},\mathcal{V}) \times \mathcal{T}(\mathcal{F},\mathcal{V})$ be a finite set of equations. The *reduction relation* $\rightarrow^E \subseteq \mathcal{T}(\mathcal{F},\mathcal{V}) \times \mathcal{T}(\mathcal{F},\mathcal{V})$ is defined as

$s \rightarrow^E t$ iff $\exists\{s',t'\} \in E, p \in pos(s), \sigma$ substitution such that $s\,|_p = \sigma(s')$ and $t = s[\sigma(t')]_p$.

Definition 1.2.14 (E-normal form)**.** A term t is an *E-normal form* if there does not exist $t' \in \mathcal{T}(\mathcal{F},\mathcal{V})$ such that $t \rightarrow^E t'$. The set of terms in E-normal form is denoted as $\mathcal{T}_E(\mathcal{F},\mathcal{V})$ and the set of ground terms in in E-normal form as $\mathcal{T}_E(\mathcal{F})$.

Birkhoff's Theorem in [12] states that the reflexive, symmetric, and transitive closure of the relation $\rightarrow^E$ ($\overset{*}{\longleftrightarrow}_E$) coincides with the semantic consequence relation $\approx_E$.

Definition 1.2.15 (Type Equation)**.** An equation $s \approx t$ is called: *regular* if it holds that $var(s) = var(t)$; *collapse* if $s \in \mathcal{T}(\mathcal{F},\mathcal{V})$, $t \in \mathcal{V}$ and $t \in var(s)$; *subterm collapsing* if $s \in \mathcal{T}(\mathcal{F},\mathcal{V})$ and $t \in subt(s)$.

Example 1.2.16. The equation of the axiom of the C-theory in Definition 1.2.11 is regular since both sides contain the same variables. The equation of the axiom of the I-theory in Definition 1.2.11 is a collapse equation since the right-hand side of the equation is a variable that belongs to the left-hand side. All equations of the $\mathfrak{a}$-theory in Definition 1.2.10 are subterm collapsing equations since one of the sides of all the axioms is ε_{f_k}, which is a subterm of the other side in each equation. $\diamond$

1.3 Anti-Unification

Definition 1.3.1 (E-Generalization, $\preceq_E$)**.** The *generalization relation* of the E-theory holds for terms $r, s \in \mathcal{T}(\mathcal{F},\mathcal{V})$, written $r \preceq_E s$, if there exists a substitution σ such that $r\sigma \approx_E s$. Whenever $r \preceq_E s$, we say that r is *more general* than s modulo E or r is an *E-generalization* of s. The set of all E-generalizations of s and t is denoted as $\mathcal{G}_E(s,t)$. By $\prec_E$ and $\simeq_E$, we denote the strict and equivalence relations induced by $\preceq_E$.

Remark. For any pair of terms, a variable is always a generalization: the *trivial generalization.*

Definition 1.3.2 (Associated Substitutions)**.** Let r be an E-generalization of s and t. The substitutions σ and ρ, such that $r\sigma \approx_E s$ and $r\rho \approx_E t$ are called as *associated substitutions* of r.

Example 1.3.3. Observe that the term $f(f(b,x),y)$ is an $\mathfrak{a}$-generalization of the terms ε_f and $f(f(b,c),a)$, for f an $\mathfrak{a}$-symbol. Indeed, $\sigma = \{x \mapsto \varepsilon_f, y \mapsto a\}$ and $\rho = \{x \mapsto c, y \mapsto a\}$ satisfy $f(f(b,x),y)\sigma = f(f(b,\varepsilon_f),a) \approx_{\mathfrak{a}} \varepsilon_f$ and $f(f(b,x),y)\rho = f(f(b,c),a)$. In this case, σ and ρ are the associated substitutions of $f(f(b,x),y)$. $\diamond$

Definition 1.3.4 (Least General Generalization, `lgg`)**.** Let $s,t \in \mathcal{T}(\mathcal{F},\mathcal{V})$. A term $r \in \mathcal{G}_E(s,t)$ is a *least general generalization*, in short `lgg`, of s and t if there does not exist $r' \in \mathcal{G}_E(s,t)$ such that $r \prec_{\mathfrak{a}} r'$.

Example 1.3.5. Following Example 1.3.3, the term $f(f(b,x),a)$ is an `lgg` of ε_f and $f(f(b,c),a)$, since any instantiation of $f(f(b,x),a)$ leads to terms that are not generalizations, that is, there does not exist $r \in \mathcal{G}_{\mathfrak{a}}(\varepsilon_f, f(f(b,c),a))$ such that $f(f(b,x),a) \prec r$. Notice that the generalization $f(f(b,x),y)$ given in Example 1.3.3 is not an `lgg`.

Definition 1.3.6 (Complete and Minimal Complete Set of E-generalizations)**.** Let s and t be terms and $G \subseteq \mathcal{G}_E(s,t)$, G is a *complete set of E-generalizations* of the terms s and t if for each $r \in \mathcal{G}_E(s,t)$ there exists $r' \in G$ such that $r \preceq_E r'$. If in addition, for $r, r' \in G$, $r \preceq_E r'$ implies $r = r'$, then we say that G is *minimal* and G is called the *minimal complete set of E-generalizations* and is denoted by $mcsg_E(s,t)$.

Example 1.3.7. From Example 1.3.3, the minimal complete set of $\mathfrak{a}$-generalizations of the terms ε_f and $f(f(b,c),a)$ is

$$mcsg_{\mathfrak{a}}(\varepsilon_f, f(f(b,c),a)) = \{f(f(x,c),a), f(f(b,x),a), f(f(b,c),x)\}.$$

$\diamond$

Definition 1.3.8 (Anti-Unification Type)**.** The anti-unification type of an E-theory may have one of the following forms:

- *Unitary*: $mcsg_E(s,t)$ exists for all $s,t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ and is always singleton.
- *Finitary*: $mcsg_E(s,t)$ exists and is finite for all $s,t \in \mathcal{T}(\mathcal{F},\mathcal{V})$, and there exist $s',t' \in \mathcal{T}(\mathcal{F},\mathcal{V})$ for which $1 < |mcsg_E(s',t')| < \infty$.
- *Infinitary*: $mcsg_E(s,t)$ exists for all $s,t \in \mathcal{T}(\mathcal{F},\mathcal{V})$, and there exist $s',t' \in \mathcal{T}(\mathcal{F},\mathcal{V})$ such that $mcsg_E(s',t')$ is infinite.
- *Nullary*: for some $s,t \in \mathcal{T}(\mathcal{F},\mathcal{V})$, $mcsg_E(s,t)$ does not exist.

Remark. This work considers just anti-unification of ground terms. We can establish a bijection between variables and constants and calculate the `lgg`'s for any pair of terms under this mapping. The terms obtained form a complete set of generalizations by mapping the results back to the original variables. This approach does not affect the solutions and allows for a more manageable analysis of the problem.

Definition 1.3.9 (Linear E-generalization)**.** An E-generalization r of terms s and t is called a *linear E-generalization* if r is a linear term.

Example 1.3.10. Note that the term $f(x, y)$ is a linear $\mathfrak{a}$-generalization of ε_f and $f(a, a)$. While $f(x, x)$ is a $\mathfrak{a}$-generalization that is not linear. $\diamond$

The Definition 1.3.9 extends to linear least general generalizations, complete sets, and minimal complete sets of linear E-generalizations as expected. The task of computing linear E-generalizations of a pair of terms is referred to as the *linear variant* of anti-unification.

Definition 1.3.11 (More General Relation for Substitutions)**.** Let E be an equational theory and V be a set of variables. The substitution σ is *more general modulo E* on V than the substitution θ if there exists a substitution δ such that $x\sigma\delta \approx_E x\theta$ for all $x \in V$. In this case, we write $(\sigma \preceq_E \theta)|_V$.

Definition 1.3.12 (Nested Weight)**.** Let s be a term, and f be an absorptive symbol. The *nested weight of s regarding f* is defined recursively as follows:

$$\begin{aligned} nes_f(s) &:= 0, && \text{if } head(s) \neq f; \\ nes_f(f(s_1, s_2)) &:= 1 + \max\{nes_f(s_1), nes_f(s_2)\}, && \text{otherwise.} \end{aligned}$$

Definition 1.3.13 (Decomposable Position)**.** Let s and t be ground terms. *A decomposable position* of s and t is a position $p \in pos(s) \cap pos(t)$ such that $head(s|_q) = head(t|_q)$, for all $q \sqsubseteq p$. The set of all the decomposable positions is denoted as $decpos(s, t)$.

Definition 1.3.14 (Positions in a Decomposable Context)**.** Let s and t be ground terms. The set $P(s, t)$ of *positions in a decomposable context* of s and t is defined as follows:

$$P(s, t) = \{p \in pos(s) \cap pos(t) \mid \text{for all } q \sqsubset p, q \in decpos(s, t)\}.$$

Definition 1.3.15 (Expansible-Solvable Positions)**.** Let s and t be ground terms. The set of *expansible-solvable positions* of s and t is defined as follows:

$$ES(s, t) = \{p \in P(s, t) \mid \text{there does not exist } p' \text{ with } p \sqsubset p' \text{ and } p' \in P(s, t)\}.$$

Example 1.3.16. Let $s = g(g(b, a), f(\varepsilon_g, \varepsilon_g))$ and $t = g(g(b, b), f(g(a, a), \varepsilon_g))$ be two ground terms. Then, the set of decomposable positions is:

$$decpos(s, t) = \{\varepsilon, 1, 1.1, 2, 2.2\}.$$

Using Definitions 1.3.14 and 1.3.15, the set of positions in a decomposable context and the set of expansible-solvable positions of the terms s and t are:

$$P(s,t) = \{\varepsilon, 1, 1.1, 1.2, 2, 2.1, 2.2\} \text{ and } ES(s,t) = \{1.1, 1.2, 2.1, 2.2\}.$$

Definition 1.3.17 (Expansible, Trivially-Decomposable and Solvable Positions)**.** Let s and t be ground terms and $p \in ES(s,t)$. We classify these positions as below.

- p is an *expansible position* regarding symbol f if $\{head(s|_p), head(t|_p)\} \subseteq \{\varepsilon_f, f\}$. Additionally, an expansible position p is *lateral-expansible* if the cardinality of the set is two, and *both-expansible*, if the set is equal to $\{\varepsilon_f\}$. If p is a lateral-expansible position, the subterm $s|_p$ or $t|_p$ with head f is called the *non-absorbing term at position* p.
- p is a *trivially-decomposable position* if $head(s|_p) = head(t|_p)$.
- p is a *solvable position* if $head(s|_p) \neq head(t|_p)$, and they are not absorptive related symbols.

Example 1.3.18. Considering the same terms s and t of Example 1.3.16 and its respective set of expansible-solvable positions $ES(s,t) = \{1.1, 1.2, 2.1, 2.2\}$. We classify each expansible-solvable position below following Definition 1.3.17.

- $p = 1.1$: $s|_p = b$, $t|_p = b$. Since $head(s|_p) = head(t|_p)$, 1.1 is a *trivially-decomposable position*.
- $p = 1.2$: $s|_p = a$, $t|_p = b$. The heads differ and are not absorptive related symbols. Then 1.2 is a *solvable position*.
- $p = 2.1$: $s|_p = \varepsilon_g$, $t|_p = g(a,a)$. Since $head(s|_p) = \varepsilon_g$ and $head(t|_p) = g$ are related absorptive symbols, 2.1 is an *lateral-expansible position* regarding symbol g.
- $p = 2.2$: $s|_p = \varepsilon_g$, $t|_p = \varepsilon_g$. Since $head(s|_p) = head(t|_p) = \varepsilon_g$, 2.2 is a *trivially-decomposable position* and a *both-expansible position* regarding symbol g. ◇

Lemma 1.3.19. *Let s and t be ground terms. Then all positions in the set $ES(s,t)$ of expansible-solvable positions are pairwise parallel. That is, for all $p, q \in ES(s,t)$, if $p \neq q$, then $p \not\sqsubset q$ and $q \not\sqsubset p$.*

Proof: Let $p, q \in ES(s,t)$ and suppose $p \neq q$. We show that $p \not\sqsubset q$ and $q \not\sqsubset p$. Assume, for contradiction, that $p \sqsubset q$. Since $q \in ES(s,t) \subseteq P(s,t)$, it follows that $q \in P(s,t)$ and is a proper prefix of p. This contradicts the definition of $ES(s,t)$, which requires that no proper prefix of p belongs to $P(s,t)$. The argument is symmetric if we assume that $q \sqsubset p$. Therefore, we have that $p \parallel q$. □

Definition 1.3.20 (Collapsible Term and Subterm)**.** A non-ground term t is called *collapsible* if there exists a variable $y \in var(t)$ such that $t\{y \mapsto \varepsilon_f\} \approx_{\mathfrak{a}} \varepsilon_f$, for some absorbing constant ε_f. In this case, we say that t is collapsible at variable y, or that t is *y-collapsible*.

Furthermore, a subterm s' of s is called a *collapsible subterm* of s, if s' is a collapsible term.

Example 1.3.21. Let f be an $\mathfrak{a}$-symbol and the term $r = g(f(h(f(x, f(y, a))), y), x)$. The subterms $r|_1 = f(h(f(x, f(y, a))), y)$, $r|_{1.1.1} = f(x, f(y, a))$, and $r|_{1.1.1.2} = f(y, a)$ are collapsible subterms of r, since each of them collapses to ε_f under the substitution $\{y \mapsto \varepsilon_f\}$. Moreover, all of them are y-collapsible. ◇

This work utilizes a special constant: *the wild card* ($\star$). This constant plays a crucial role in the algorithm, and its use is explained in Chapter 2. Intuitively, the symbol $\star$ serves as a placeholder that may represent any ground term.

Definition 1.3.22 (Anti-Unification Triple, AUT)**.** Let $x \in \mathcal{V}$ and $s, t \in \mathcal{T}(\mathcal{F})$, an *anti-unification triple* $\mathtt{a}$, for short AUT, is a triple of the following forms:

1. either $s \triangleq_x \star$ or $\star \triangleq_x t$ (*wild AUT*);
2. the triple is of the form $s \triangleq_x t$ where $head(s) \neq head(t)$ and they are not related absorptive symbols (*solved AUT*);
3. $s \triangleq_x t$ and it is neither wild nor solved (*unsolved AUT*).

All of the forms mentioned above can be called AUT. The variable x is called the *label* of the AUT $\mathtt{a}$ ($label(\mathtt{a})$), the term on the left-hand side of the AUT $\mathtt{a}$ is denoted by $lhs(\mathtt{a})$, and the term on the right-hand side of the AUT $\mathtt{a}$ is denoted by $rhs(\mathtt{a})$.

When we want to anti-unify s and t, we say that we have an *anti-unification problem* $s \triangleq t$ or just a problem. Any generalization of s and t is called a *solution* of this anti-unification problem.

Example 1.3.23. Let f be a binary symbol, a, b, ε_f be constants, and $x, y, z \in \mathcal{V}$. The triple $\star \triangleq_z f(a, a)$ is a wild AUT, the triple $a \triangleq_x b$ is a solved AUT, and the triple $\varepsilon_f \triangleq_y f(a, b)$ is an unsolved AUT. ◇

For a set A of AUTs we can extend the *size* operator as:

$$size(A) := \sum_{\mathtt{a} \in A} size(lhs(\mathtt{a})) + size(rhs(\mathtt{a})).$$

and the *set of labels of* A is defined as

$$labels(A) := \{label(\mathtt{a}) \mid \mathtt{a} \in A\}.$$

Definition 1.3.24 (Valid Set of AUTs)**.** A set A of AUTs is *valid* if $|labels(A)| = |A|$.

Example 1.3.25. The sets $\{a \triangleq_x f(b,b), \varepsilon_f \triangleq_y f(a,b)\}$ and $\{\star \triangleq_w b, f(a,b) \triangleq_y \star\}$ are examples of a valid set of AUTs. The sets $\{a \triangleq_x b, f(a,a) \triangleq_x b\}$ and $\{\star \triangleq_x \star, \star \triangleq_y a\}$ are not valid sets of AUTs, since the first one has two AUTs with the same label and the second includes the triple $\star \triangleq_x \star$ which is not an AUT. ◇

Definition 1.3.26 (Nested $\mathfrak{a}$-symbols of an AUT)**.** Let $\mathtt{a}$ be an AUT, and $p_1, \ldots, p_k$ all the lateral-expansible positions of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$. Assume that $p_1, \ldots, p_k$ are lateral-expansible positions regarding the symbols $f_1, \ldots, f_k$, respectively, and that $r_1, \ldots, r_k$ are their respective non-absorbing terms. The *number of nested* $\mathfrak{a}$*-symbols* of $\mathtt{a}$ is defined as:

$$nes(\mathtt{a}) = \begin{cases} \sum_{i=1}^{k} nes_{f_i}(r_i) & \text{if } k \geq 1, \\ 0 & \text{if } \mathtt{a} \text{ has no lateral-expansible position.} \end{cases}$$

Example 1.3.27. Consider the AUT $\mathtt{a} = g(\varepsilon_f, f(a,c)) \triangleq_x g(f(f(a,a),b), \varepsilon_f)$. First, note that the lateral-expansible positions of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$ regarding to f are $1, 2$. Furthermore, it holds that $nes_f(f(f(a,a),b)) = 1 + max\{nes_f(f(a,a)), nes_f(b)\} = 2$. and $nes_f(f(a,c)) = 1 + max\{nes_f(a), nes_f(c)\} = 1$. It follows that

$$nes(\mathtt{a}) = nes_f(f(f(a,a),b)) + nes_f(f(a,c)) = 3.$$

◇

The notion of nested $\mathfrak{a}$-symbols can be extended naturally to a valid set of AUTs A as follows:

$$nes(A) := \sum_{\mathtt{a} \in A} nes(\mathtt{a})$$

Definition 1.3.28 (Merged Set of AUTs)**.** Let A be a valid set of AUTs, A is *merged* if there do not exist different AUTs in A, $\mathtt{a}_1$ and $\mathtt{a}_2$, such that $lhs(\mathtt{a}_1) = lhs(\mathtt{a}_2)$ and $rhs(\mathtt{a}_1) = rhs(\mathtt{a}_2)$.

Example 1.3.29. The sets $\{a \triangleq_x b, f(a,a) \triangleq_y b\}$ and $\{b \triangleq_z f(b,a)\}$ are merged sets of AUTs, while the set $\{a \triangleq_x b, a \triangleq_y b\}$ is not merged. ◇

Definition 1.3.30 (Left and Right Substitutions)**.** Let A be a finite valid set of AUTs. The *left substitution* σ_A^l and *right substitution* σ_A^r related to A are defined as follows:

$$\sigma_A^l = \{label(\mathtt{a}) \mapsto lhs(\mathtt{a}) \mid \mathtt{a} \in A\} \text{ and } \sigma_A^r = \{label(\mathtt{a}) \mapsto rhs(\mathtt{a}) \mid \mathtt{a} \in A\}.$$

Definition 1.3.31 (Set of Left and Right Wild AUTs)**.** Let A be a finite valid set of AUTs. The set of *left wild AUTs* A^l and the set of *right wild AUTs* A^r related to A are defined as follows:

$$A^l = \{\mathtt{a} \in A \mid \mathtt{a} \text{ is wild and } \mathit{lhs}(\mathtt{a}) \neq \star\} \text{ and } A^r = \{\mathtt{a} \in A \mid \mathtt{a} \text{ is wild and } \mathit{rhs}(\mathtt{a}) \neq \star\}.$$

The set $A^n = A - (A^l \cup A^r)$ is called the set of *non-wild AUTs* related to A.

Example 1.3.32. Let $A = \{a \triangleq_w \star, b \triangleq_x \star, \star \triangleq_y f(a,c), \varepsilon_f \triangleq_z \varepsilon_f\}$ be a valid set of AUTs. The left and right wild AUTs are:

$$A^l = \{a \triangleq_w \star, b \triangleq_x \star\} \text{ and } A^r = \{\star \triangleq_y f(a,c)\}.$$

The set of non-wild AUTs related to A is $A^n = \{\varepsilon_f \triangleq_z \varepsilon_f\}$. ◇

Definition 1.3.33 (Substitution Generalization of a Valid Set of AUTs)**.** Let A be a valid set of AUTs over E, where E is either the absorptive or the absorptive commutative theory. A substitution γ is a *substitution generalization* of A if there exist substitutions ρ_l and ρ_r, such that $\mathit{dom}(\gamma\rho_l) = \mathit{dom}(\gamma\rho_r) = \mathit{labels}(A)$ and such that every AUT $\mathtt{a} \in A$ satisfies the following conditions:

- $\mathit{label}(\mathtt{a})\gamma\rho_l \approx_E \mathit{lhs}(\mathtt{a})$ and $\mathit{label}(\mathtt{a})\gamma\rho_r \approx_E \mathit{rhs}(\mathtt{a})$ if $\mathtt{a}$ is not wild,
- $\mathit{label}(\mathtt{a})\gamma\rho_l \approx_E \mathit{lhs}(\mathtt{a})$ if $\mathtt{a} \in A^l$,
- $\mathit{label}(\mathtt{a})\gamma\rho_r \approx_E \mathit{rhs}(\mathtt{a})$ if $\mathtt{a} \in A^r$.

We call the substitutions ρ_l and ρ_r as *associated substitutions* of γ. For short, we will write "γ is a generalization of A". The set of substitution generalizations of A is denoted as $\mathcal{G}_E(A)$.

Notice that in Definition 1.3.33, for each wild AUT of the form $s \triangleq_x \star$ (resp. $\star \triangleq_x t$), only the non-wild term s (resp. t) is required to be generalized by γ through the corresponding substitution ρ_l (resp. ρ_r). The action of the substitution ρ_r (resp. ρ_l) on the generalization $x\gamma$ is unrestricted, since the corresponding side of the AUT is the constant $\star$.

Example 1.3.34. Consider $A = \{g(a,b) \triangleq_x g(a,c), \varepsilon_f \triangleq_y f(c,d), a \triangleq_z c\}$ be a set of valid AUTs over $\mathfrak{a}(\{f\})$. The substitution $\gamma = \{x \mapsto g(a,w_1), y \mapsto f(w_2,d), z \mapsto w_3\}$ is a substitution generalization for A. Observe that for $\rho_l = \{w_1 \mapsto b, w_2 \mapsto \varepsilon_f, w_3 \mapsto a\}$ and $\rho_r = \{w_1 \mapsto c, w_2 \mapsto a, w_3 \mapsto c\}$, we have that $x\gamma\rho_l \approx_{\mathfrak{a}} g(a,b)$, $y\gamma\rho_l \approx_{\mathfrak{a}} \varepsilon_f$, $z\gamma\rho_l \approx_{\mathfrak{a}} a$, and $x\gamma\rho_r \approx_{\mathfrak{a}} g(a,c)$, $y\gamma\rho_r \approx_{\mathfrak{a}} f(c,d)$, $z\gamma\rho_r \approx_{\mathfrak{a}} c$. ◇

Example 1.3.35. Consider $A = \{\star \triangleq_x h(b), f(a,b) \triangleq_y \star\}$ be a set of wild valid AUTs over $\mathfrak{a}(\{f\})$. The substitution $\gamma = \{x \mapsto h(w), y \mapsto f(z_1, z_2)\}$ is a substitution generalization for A, since for $\rho_l = \{w \mapsto a, z_1 \mapsto a, z_2 \mapsto b\}$ and $\rho_r = \{w \mapsto b, z_1 \mapsto b, z_2 \mapsto a\}$ we have that $x\gamma\rho_r \approx_{\mathfrak{a}} h(b)$ and $y\gamma\rho_l \approx_{\mathfrak{a}} f(a,b)$. ◇

Definition 1.3.36 (Meta-terms)**.** Let $\mathcal{F}$ be a signature, $\mathcal{V}$ be a countable set of variables, and $\mathcal{N}$ be a countable set of *non-terminal symbols* of arity 0, such that these three sets are pairwise disjoint. For $\Sigma \subseteq \mathcal{F} \cup \mathcal{V}$ and $N \subseteq \mathcal{N}$, the set $\mathcal{T}(\Sigma, N)$, defined inductively as in Definition 1.1.2, is called the set of *meta-terms*.

Definition 1.3.37 (Regular Tree Grammar)**.** A *regular tree grammar* $\mathfrak{G}$ is a quadruple of the form $\langle S_0; N; \Sigma; R\rangle$, where the symbol $S_0 \in N$ is called the *starting non-terminal*, N is the finite set of *non-terminals*, $\Sigma \subseteq \mathcal{F} \cup \mathcal{V}$, and R is the set of *productions rules* of the form $T \longrightarrow t$ where $T \in N$ and $t \in \mathcal{T}(\Sigma, N)$.

Definition 1.3.38 (Grammar Derivability Relation, Language)**.** Let $\mathfrak{G} = \langle S_0; N; \Sigma; R\rangle$ be a regular tree grammar, the *grammatical derivation relation* $\longrightarrow_{\mathfrak{G}}$ is a relation over $\mathcal{T}(\Sigma, N)$ such that $s \longrightarrow_{\mathfrak{G}} t$ if there exists $p \in pos(s)$ and a rule $T \longrightarrow t'$ such that $s|_p = T$ and $t = s[t']_p$.

Definition 1.3.39 (Langeuage Generated by $\mathcal{L}(\mathfrak{G})$)**.** The *language generated* by $\mathfrak{G}$ is the set of the terms $\mathcal{L}(\mathfrak{G}) := \{t \mid t \in \mathcal{T}(\mathcal{F} \cap \Sigma, \mathcal{V} \cap \Sigma) \text{ and } S_0 \longrightarrow^+_{\mathfrak{G}} t\}$, where $\longrightarrow^+_{\mathfrak{G}}$ is the transitive closure of the relation $\longrightarrow_{\mathfrak{G}}$. Similarly, for any non-terminal symbol T in N, the language of terms obtained by a grammatical derivation from T is the set of terms $\mathcal{L}(\mathfrak{G}, T) = \{t \in \mathcal{T}(\mathcal{F} \cap \Sigma, \mathcal{V} \cap \Sigma) \mid T \longrightarrow^+_{\mathfrak{G}} t\}$.

Given a regular tree grammar $\langle S_0; N; \Sigma; R\rangle$, we will abbreviate production rules in R by joining rules starting with the same non-terminal, i.e., $T \longrightarrow t$ and $T \longrightarrow t'$ are abbreviated as $T \longrightarrow t \mid t'$.

1.4 Related Work

The study of anti-unification over first-order theories began with the works of Gordon Plotkin and John C. Reynolds in the 1970s [39, 41], who introduced algorithms for computing least general generalizations in the syntactic theory $\emptyset$, establishing that the generalization type for this theory is unitary. Building on these foundations, Huet [31] also contributed an algorithm for computing lggs in the first-order syntactic setting. Later, Baader [11] investigated anti-unification in commutative theories, covering settings such as commutative monoids (ACU), commutative idempotent monoids (ACUI), and Abelian groups, and proved that anti-unification in these cases remains unitary. More recent

works consider theories that include symbols satisfying regular equations as axioms. For instance, María Alpuente et al. [1, 2] investigate C, A, and AC theories, providing an algorithm for computing generalizations and determining the respective type of anti-unification in each case.

David Cerna and Temur Kutsia [25, 26] explore regular theories that include symbols satisfying collapse equations as axioms. They analyze the theories I, AI, and CI, which are of infinitary type, and present an algorithm that computes a grammar whose language generates an infinite set of generalizations for a pair of terms. Additionally, they treat $U^{>1}$, which is nullary, meaning it is not possible to find the minimal complete set of generalizations for some pairs of terms, along with some combinations, such as $(AU)^{>1}$, $(CU)^{>1}$, and $(ACU)^{>1}$, which are also nullary. Finally, David Cerna [23] provides a proof that the theories of semirings and commutative semirings are nullary. Given an infinite chain of generalizations starting from a term, it is used to demonstrate the nullarity of the theories $(ACU)^2$, $(\mathfrak{a}ACU)(ACU)$, and $\mathfrak{a}U$. Table 1.2 summarizes the theories mentioned above, including their type and the corresponding authors with references.

In the following sections, we provide a more detailed discussion of the anti-unification problem and the anti-unification type of some of these theories, specifically C-theory, I-theory, and U-theory.

Theory	**Type**	**References**
$\emptyset$	1	G. Plotkin [39] and J. Reynolds [41]
A	ω	M. Alpuente et al. [1];
C	ω	M. Alpuente et al. [1]
AC	ω	M. Alpuente et al. [1]
U^1	ω	D. Cerna and T. Kutsia [26]
$U^{>1}$ (linear variant)	ω	D. Cerna and T. Kutsia [26]
$U^{>1}$ $(AU)^{>1}$, $(CU)^{>1}$, $(ACU)^{>1}$	0	D. Cerna and T. Kutsia [26]
$I^{\geq 1}$	∞	D. Cerna and T. Kutsia [26]
$(UI)^2$, $(ACUI)^2$	0	D. Cerna [23]
SRNG, CSRNG	0	D. Cerna [23]
$(ACU)^2$, $(\mathfrak{a}ACU)(ACU)$, $\mathfrak{a}U$	0	D. Cerna [23]

Table 1.2: First-order anti-unification

1.4.1 Syntactic Anti-Unification

This section illustrates an approach to the syntactic anti-unification problem, where the terms under analysis are not subject to any axioms. This purely syntactic setting represents one of the simplest theories to analyze and was the first to be studied in the context of anti-unification.

The original motivation for introducing anti-unification was its application in automating induction. In his pioneering article An Experiment in Automatic Induction (1971) [40], Popplestone proposed the idea that generalizations—and in particular, least general generalizations—of terms (or literals, in his terminology) could be computed and used as a foundation for inductive inference. He suggested that such generalizations would be useful in discovering patterns or regularities essential for automating inductive reasoning.

Building upon this insight, Plotkin [39] and Reynolds [41] independently developed the first formal algorithms for syntactic anti-unification in 1970. Both proposed procedures for computing a unique `lgg`, modulo variable renaming, of two first-order terms.

Later in 1976, Huet formulated an algorithm for computing `lgg`s based on a recursive definition using equations. His method relies on a bijection ϕ from pairs of terms to variables, ensuring that the resulting generalizations are the least general among all possible ones. The algorithm is formally defined through a recursive function λ on $\mathcal{T}(\mathcal{F},\mathcal{V}) \times \mathcal{T}(\mathcal{F},\mathcal{V})$ defined as follows:

- $\lambda(f(s_1,\ldots,f_n), f(t_1,\ldots,t_n)) = f(\lambda(s_1,t_1),\ldots,\lambda(s_n,t_n))$ for any $f \in \mathcal{F}$
- $\lambda(s,t) = \phi(s,t)$, otherwise.

In 2014, Alpuente et al. [1] revisited syntactic anti-unification through a rule-based algorithm, building on these foundational ideas. Their approach adapted Huet's structural methodology but organized the algorithm as a transformation system acting on structured triples called configurations of the form $\langle A; S; \theta\rangle$, where:

- A is the set that contains the AUTs that still need to be solved.
- S is the store, collecting already processed and solved AUTs.
- θ is a substitution that captures the generalization computed so far.

Since this case does not involve any equational theory, an AUT differs from the one defined in Definition 1.3.22. In this context, an AUT is a triple of the form $s \triangleq_x t$, for arbitrary terms s and t. An AUT is considered solved if $head(s) \neq head(t)$, and unsolved otherwise. The algorithm consists of the inference rules Decomposition (Dec), Solve (Sol), and Recover (Rec) presented in Table 1.3.

$$(\overset{Dec}{\Longrightarrow}) \quad \frac{\langle \{f(s_1,\ldots,s_n) \triangleq_x f(t_1,\ldots,t_n)\} \uplus A; S; \theta\rangle}{\langle \{s_1 \triangleq_{y_1} t_1,\ldots,s_n \triangleq_{y_n} t_n\} \cup A; S; \theta\{x \mapsto f(y_1,\ldots,y_n)\}\rangle}$$

where f is an n-ary symbol, $n \geq 0$, and $y_1,\ldots,y_n$ are fresh variables.

$$(\overset{Sol}{\Longrightarrow}) \quad \frac{\langle \{s \triangleq_x t\} \uplus A; S; \theta\rangle}{\langle A; \{s \triangleq_x t\} \cup S; \theta\rangle}$$

where $s \triangleq_x t$ is a solved AUT.

$$(\overset{Rec}{\Longrightarrow}) \quad \frac{\langle \{s \triangleq_x t\} \uplus A; \{s \triangleq_y t\} \uplus S; \theta\rangle}{\langle A; \{s \triangleq_y t\} \cup S; \theta\{x \mapsto y\}\rangle}$$

Table 1.3: Generalization rules for syntactic theory.

They set that the algorithm starts from an *initial* configuration, of the form $\langle \{s \triangleq_x t\}; \emptyset; id\rangle$, and applies the rules above as long as possible. Moreover, a configuration is *final* if no rule can be applied and is of the form $\langle \emptyset; S'; \theta'\rangle$. It is possible to prove that the algorithm described above is confluent and produces a unique final configuration modulo variable renaming. Indeed, if the final configuration is of the form $\langle \emptyset; S'; \theta'\rangle$, then the computed solution is given by $x\theta$ and is an `lgg` of s and t. We illustrate the algorithm finding the `lgg` of the terms $g(a, g(a, c))$ and $g(b, g(b, a))$ in the example below.

Example 1.4.1. The configuration $\langle \{g(a, g(a, c)) \triangleq_x g(b, g(b, a))\}; \emptyset; \theta\rangle$ is the initial configuration associated with the anti-unification problem of the terms $g(a, g(a, c))$ and $g(b, g(b, a))$. Using exhaustively the rules of Table 1.3 over this configuration, we get the following derivation:

$$\begin{aligned}
\langle \{g(a, g(a, c)) \triangleq_x g(b, g(b, a))\}; \emptyset; id\rangle &\overset{Dec}{\Longrightarrow} \\
\langle \{a \triangleq_u b, g(a, c) \triangleq_v g(b, a)\}; \emptyset; \{x \mapsto g(u, v)\}\rangle &\overset{Sol}{\Longrightarrow} \\
\langle \{g(a, c) \triangleq_v g(b, a)\}; \{a \triangleq_u b\}; \{x \mapsto g(u, v)\}\rangle &\overset{Dec}{\Longrightarrow} \\
\langle \{a \triangleq_y b, c \triangleq_z a\}; \{a \triangleq_u b\}; \{x \mapsto g(u, g(y, z)), v \mapsto g(y, z)\}\rangle &\overset{Rec}{\Longrightarrow} . \\
\langle \{c \triangleq_z a\}; \{a \triangleq_u b\}; \{x \mapsto g(u, g(u, z)), v \mapsto g(u, z), y \mapsto u\}\rangle &\overset{Sol}{\Longrightarrow} \\
\langle \emptyset; \{a \triangleq_u b, c \triangleq_z a\}; \{x \mapsto g(u, g(u, z)), v \mapsto g(u, z), y \mapsto u\}\rangle &
\end{aligned}$$

Notice that the last configuration of this derivation is the final configuration, then the computed `lgg` for the AUT $g(a, g(a, c)) \triangleq_x g(b, g(b, a))$ is given by $x\theta' = g(u, g(u, z))$, where θ' is the substitution of the final configuration. Indeed, $x\theta'\sigma^l_{S'} = g(a, g(a, c))$ and $x\theta\sigma^r_{S'} = g(b, g(b, a))$, where S' is the final store, $\sigma^l_{S'}$ and $\sigma^r_{S'}$ are the substitutions of Definition 1.3.30. ◇

To prove that $x\theta$ is an `lgg` of the AUT $s \triangleq_x t$ for any final configuration derived from the associated initial configuration, it is necessary to prove that the procedure is

terminating, sound, and complete. Moreover, to prove that $x\theta$ is unique, it is required to prove that the algorithm is confluent up to renaming. The proofs of these results will not be presented here, but the respective statements will be stated.

Theorem 1.4.2 (Termination)**.** *For all input $\langle A; S; \theta\rangle$, the algorithm terminates in a final configuration of the form $\langle \emptyset; S'; \theta'\rangle$.*

The proof of Termination can be made over the $size(A)$ as measure; each rule application decreases this measure. The soundness and the completeness are stated as follows:

Theorem 1.4.3 (Soundness)**.** *Let $\langle A; S; \theta\rangle$ be a configuration, and $\langle \emptyset; S'; \theta'\rangle$ be a final configuration after an exhaustive application of the rules of Table 1.3. Then the term $x\theta$ is a generalization of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$, for all $\mathtt{a} \in A$.*

Theorem 1.4.4 (Completeness)**.** *Let $\langle A; S; \theta\rangle$ be a configuration, and γ the substitution generalization of $A \cup S$, then $x\gamma \preceq x\theta'$, where θ' is the substitution of a final configuration derived from $\langle A; S; \theta\rangle$.*

Both proofs proceed by induction on the size $size(A)$, combined with case analysis for each inference rule. Soundness and completeness guarantee that the algorithm computes a complete set of generalizations. However, since the minimal complete set of syntactic generalizations contains only one `lgg`, it is necessary to determine whether the algorithm produces generalizations that are not `lgg`s. In this case, the algorithm computes a unique final configuration, up to variable renaming.

Theorem 1.4.5 (Confluence)**.** *Let $\langle A; S; \theta\rangle$ be a configuration, and let $\langle \emptyset; S_1; \theta_1\rangle$ and $\langle \emptyset; S_2; \theta_2\rangle$ be two final configurations derived from it. Then there exists a renaming of the labels in S_1 and S_2, as well as of the variables in the domains and ranges of θ_1 and θ_2, such that the two configurations become identical, that is, they contain exactly the same AUTs in the store and the same bindings in the substitutions.*

The proof of confluence reduces to local confluence, since termination is guaranteed. To establish local confluence, we analyze all cases in which branching occurs due to different choices of AUTs in the set of unsolved triples. As a consequence of soundness, completeness, and confluence, the algorithm computes a unique `lgg` for each AUT. Therefore, the minimal complete set of generalizations is unitary, which implies that the syntactic anti-unification problem is of unitary type.

1.4.2 Anti-Unification in Commutative Theory

The anti-unification problem over C-theory was studied in [1], where the authors developed an algorithm that computes generalizations for a pair of terms and proved that this theory

is finitary. This subsection presents the approach of [1] for anti-unification in C-theory. In particular, we illustrate some examples of C-generalizations and discuss the proposed algorithm, along with the lemmas and theorems that establish the type of this theory.

To begin, the following example illustrates a C-generalization of two terms headed by a C-symbol g. The possible C-generalizations depend on the structural similarity of their respective arguments. Here, commutativity allows us to swap arguments and determine common substructures.

Example 1.4.6. Consider the terms $g(a, b)$ and $g(b, c)$, where g a C-symbol. Notice that the term $g(b, x)$ is a generalization of the terms s and t. Indeed, $\sigma = \{x \mapsto a\}$ and $\rho = \{x \mapsto c\}$ satisfy $g(b, x)\sigma = g(b, a) \approx_C g(a, b)$ and $g(b, x)\rho = g(b, c)$. $\diamond$

Observe that the type of anti-unification for commutative theory is unitary or finitary, since each equational class of any term is finite, implying that any complete set of generalizations for any pair of terms is finite. The next example illustrates that there exists at least a pair of terms where the minimal complete set of generalizations has two elements.

Example 1.4.7 (Anti-Unification in Commutative Theory is not Unitary)**.** Consider the terms $s = g(g(a, b), b)$ and $t = g(a, g(b, a))$, where g is a C-symbol. The terms $g(g(a, b), x)$ and $g(g(z, y), y)$ are `lgg`s of s and t. Notice that they are not comparable, since if we assume that $g(g(a, b), x) \preceq_C g(g(z, y), y)$, this implies that there exists μ such that $g(g(a, b), x)\mu \approx_C g(g(z, y), y)$, but this not possible because μ does not affect constants. Similarly, if we assume that $g(g(z, y), y)\mu' \preceq_C g(g(a, b), x)$ is because there is μ' such that $g(g(z, y), y)\mu' \approx_C g(g(a, b), x)$, but this is not possible since there does not exist μ' such that $y\mu' = b$ and $y\mu' = x$ simultaneously. Implying that these `lgg`s are not comparable. Hence, the C-theory is not unitary. $\diamond$

The approach of [1] introduces an algorithm that computes C-generalizations for any pair of terms. This algorithm consists of a set of four inference rules, which is terminating and sound. They prove that this algorithm is also complete but not minimal. In fact, the algorithm generates a finite set of complete C-generalizations and in some cases computes comparable C-generalizations. The following example illustrates that the algorithm computes comparable C-generalizations.

Example 1.4.8. Considering the same terms s and t of Example 1.4.7 and applying all the possible inference rules presented in [1] for the anti-unification problem $s \triangleq t$ the algorithm in [1] computes three branches that generate the C-generalizations $g(x_1, y_1)$, $g(g(x_2, y_2), y_2)$, and $g(g(a, b), x_3)$. Notice that $g(x_1, y_1)$ is more general than the other generalizations. This implies that the algorithm is not computing a minimal set of generalizations. $\diamond$

In summary, the algorithm proposed in [1] computes a finite complete set of C-generalizations for any pair of terms. Since Example 1.4.7 demonstrates that this theory is not unitary, it follows that anti-unification in C-theory is finitary. Furthermore, as this set is not minimal but finite, we can build the minimal complete set of generalizations from the computed set.

1.4.3 Anti-Unification in Idempotent Theory

As was introduced in Definition 1.2.11 the idempotent theory I is a theory with one or more function symbols f holding the axiom $f(x, x) \approx x$. This subsection presents the approach of [26] for solving the anti-unification in I-theory. They introduce two algorithms that compute a grammar in which language generates a complete and minimal set of I-generalizations for any pair of terms. Additionally, they provide a pair of terms where the minimal complete set of I-generalizations is infinite.

Example 1.4.9. Let $f(a, b)$ and $f(b, a)$ be terms, where f is an I-symbol. Notice that the term $f(f(y, b), f(a, z))$ is an I-generalization that is not a syntactic generalization. Indeed, $\sigma = \{y \mapsto a, z \mapsto b\}$ and $\rho = \{y \mapsto b, z \mapsto a\}$ satisfy $f(f(y, b), f(a, z))\sigma \approx_I f(a, b)$ and $f(f(y, b), f(a, z))\rho \approx_I f(b, a)$. $\diamond$

The algorithm I-PreGen proposed in [26] is a rule-based algorithm of nine rules, which generates a computed set of bindings for any anti-unification problem. They use a minimized and merged version of these bindings modulo I to build a finite tree grammar $\mathfrak{G}$ with the procedure I-Gen as a finite representation of the computed I-generalizations. Moreover, they prove that both procedures are terminating and that any term of the language generated by the grammar, say $\mathcal{L}(\mathfrak{G})$, is an I-generalization of the respective anti-unification problem and that $\mathcal{L}(\mathfrak{G})$ is a complete set of I-generalizations, that is, they prove that I-Gen is sound and complete. The following example illustrates a pair of terms which language is an infinite set.

Example 1.4.10. Considering the same terms $f(a, b)$ and $f(b, a)$ of the Example 1.4.9. The algorithm I-Gen of [26] generates a grammar $\mathfrak{G} = \langle \boldsymbol{x}; \{\boldsymbol{x}, \boldsymbol{y}\}; \{f, a, b, z_1, z_2\}; R\rangle$, where

$$R = \begin{Bmatrix} \boldsymbol{x} \longrightarrow \boldsymbol{y},\ \boldsymbol{y} \longrightarrow f(\boldsymbol{x}, \boldsymbol{x}),\ \boldsymbol{y} \longrightarrow f(f(z_1, b), f(a, z_2)), \\ \boldsymbol{y} \longrightarrow f(f(z_1, a), f(b, z_2)),\ \boldsymbol{y} \longrightarrow f(\boldsymbol{y}, \boldsymbol{y}) \end{Bmatrix}$$

With $\boldsymbol{x}$ and $\boldsymbol{y}$ are non-terminal symbols. The language $\mathcal{L}(\mathfrak{G})$ is generated by applying these rules starting from $\boldsymbol{x}$ until the obtained term has only non-terminal symbols. Notice that the rule $\boldsymbol{y} \longrightarrow f(\boldsymbol{y}, \boldsymbol{y})$ can be applied recursively an infinite number of times, this implies that the generated language set is infinite. The following terms:

$f(f(z_1, b), f(a, z_2))$, $f(f(z_1, a), f(b, z_2))$, $f(f(f(z_1, b), f(a, z_2)), f(f(z_1, b), f(a, z_2)))$, and $f(f(f(z_1, a), f(b, z_2)), f(f(z_1, a), f(b, z_2)))$ are some of the I-generalizations generated by the language $\mathcal{L}(\mathfrak{G})$. These I-generalizations are pairwise incomparable, since any instantiation of the variables z_1 and z_2 leads to terms that are not I-generalizations of $f(a, b)$ and $f(b, a)$. $\diamond$

The next step is to prove that for any terms s and t, the language generated by the grammar computed by I-Gen is minimal. For this purpose, they prove that the way that the language is generated does not allow the production of an infinite chain of I-generalizations and that two different generalizations of the language generated are not comparable over the relation $\preceq_I$. Considering these facts together with the fact that the Example 1.4.10 generates a language that is infinite, they conclude that idempotent generalization is infinitary.

1.4.4 Anti-Unification in Unital Theory

The goal of this subsection is to present the approach of [26] to the anti-unification problem in unital theory, as defined in Definition 1.2.11. In particular, we discuss the anti-unification problem when the theory includes at least two U-symbols. For this purpose, we assume that all terms are taken from the set $\mathcal{T}(f, g, e_f, e_g, \mathcal{V})$, where both f and g are U-symbols, and e_f and e_g are their respective unit constants. The axioms of this theory, like those of idempotent theory, are collapse equations. Moreover, each equivalence class $[t]_{\approx_U}$ contains an infinite set of elements, implying that, in some cases, a pair of terms may have an infinite set of U-generalizations.

Example 1.4.11. Let e_f and e_g be terms, where e_f and e_g are unit constants, respectively, related with the U-symbols f and g. The term $f(x, g(x, y))$ is a U-generalization of e_f and e_g with associated substitutions $\sigma = \{x \mapsto e_f, y \mapsto e_g\}$ and $\rho = \{x \mapsto e_g, y \mapsto e_f\}$. $\diamond$

Observe that there are an infinite set of U-generalizations for the terms e_f and e_g of the Example 1.4.11. Indeed, if we apply the substitution $\mu = \{x \mapsto f(x, g(x, y))\}$ to the U-generalization $f(x, g(x, y))$, we obtain the term $f(f(x, g(x, y)), g(f(x, g(x, y)), y))$ which is a U-generalization of e_f and e_g through the same substitutions σ and ρ. We can apply μ recursively, and each resulting term will be a U-generalization of e_f and e_g. The authors in [26] characterize this kind of U-generalizations with the definition below.

Definition 1.4.12 (Reduced U-Generalization)**.** Let r be a U-generalization in U-normal form of s and t, and let σ_1 and σ_2 the normal associated substitutions of r. We say that r is a *reduced generalization* form if the following conditions hold:

1. For $x \in var(r)$, we have that $x\sigma_1 \not\approx_U x\sigma_2$

2. For all $x, y \in var(r)$ either $x = y$, or for some $\theta \in \{\sigma_1, \sigma_2\}$, $x\theta \not\approx_U y\theta$.

Example 1.4.13 (Reduced U-Generalization)**.** Consider the anti-unification problem mentioned in the Example 1.4.11. The term $f(x, g(x, y))$ is a reduced U-generalization of s and t with the associated substitutions σ and ρ mentioned in Example 1.4.11. While the term $f(x, g(y, z))$ is not, since the associated substitutions $\sigma_1 = \{x \mapsto e_f, y \mapsto e_f, z \mapsto e_g\}$ and $\sigma_2 = \{x \mapsto e_f, y \mapsto e_g, z \mapsto e_g\}$ does not hold the first item of Definition 1.4.12. Indeed, $x\sigma_1 \approx_U x\sigma_2$ and $z\sigma_1 \approx_U z\sigma_2$. ◇

With this characterization, it is possible to show that we can build a more specific reduced U-generalization from any U-generalization of the terms e_f and e_g. Indeed, they prove the following lemma:

Lemma 1.4.14. *For every U-generalization r in U-normal form of e_f and e_g there exists a substitution μ such that $r\mu$ is a reduced U-generalization of e_f and e_g.*

The proof sketch consists of building the substitution μ from the U-generalization r and the respective associated substitutions σ_1 and σ_2. They use the fact that any reduced U-generalization of e_f and e_g contains exactly two distinct variables and consider the cases where σ_1 and σ_2 violate some of the conditions of Definition 1.4.12. After this, they prove the lemma below.

Lemma 1.4.15. *Let r be a reduced U-generalization of e_f and e_g. Then there exists a reduced U-generalization r' of e_f and e_g such that $r \preceq_U r'$.*

The proof consists of showing that any instantiation of the form $\{x \mapsto f(x, g(x, y))\}$ for $x \in \{x, y\} = var(r)$ produces a more specific reduced U-generalization of e_f and e_g.

Considering Lemma 1.4.14 and 1.4.15, from any U-generalization r of a complete set G of e_f and e_g, we can build μ such that $r\mu$ is a reduced U-generalization, and from this generalization there exists r' such that $r\mu \preceq_U r'$ with r' also a reduced generalization, and since G is a complete set, there exists $r'' \in G$ such that $r' \preceq_U r''$. This implies that it is not possible to build the minimal complete set of generalizations for the terms e_f and e_g. Therefore, the unital anti-unification is nullary.

It is important to remark that also in [26], they prove that the linear variant of the anti-unification problem in the theory $\mathrm{U}^{>1}$ is of finitary type. Notice that the only linear `lgg` of e_f and e_f is a variable, say x, since another linear U-generalization of e_f and e_g is more general than a variable. We illustrate an example of this below.

Example 1.4.16. Consider the linear U-generalization $f(x, g(y, z))$ of e_f and e_g. We have that $f(x, g(y, z))\{x \mapsto e_f, y \mapsto e_g\} \approx_U z$, this implies that $f(x, g(y, z)) \preceq_U z$. ◇

This means that in the linear variant, it is possible to build the minimal complete set of linear U generalizations of e_f and e_g, this minimal complete set is only a variable. They provide an algorithm sound and complete in [26], this algorithm computes a complete (no minimal) finite set of linear U-generalizations. We illustrate an example in which the minimal complete set of linear U-generalizations contains at least two generalizations.

Example 1.4.17 (Anti-Unification for Linear Variant in Unital Theory is not Unitary)**.** Let $s = g(f(a, a), a)$ and $t = g(a, a)$ be terms with f and g U-symbols. The terms $r_1 = g(f(x, a), a)$ and $r_2 = g(f(a, x), a)$ are linear U-generalizations of s and t. Observe that it is not possible to find substitutions σ_1 and σ_2 such that $r_1\sigma_1 \approx_U r_2$ and $r_2\sigma_2 \approx_U r_1$, then r_1 and r_2 are incomparable. Moreover, these terms are linear `lgg`s of s and t. $\diamond$

As was illustrated in the Example 1.4.17 there is a pair of terms with at least two incomparable `lgg`s. This implies that the linear variant of U-theory is finitary. In general, it is possible to build this set by removing the more general linear U-generalizations from the complete set of linear U-generalizations computed by the proposed algorithm.

Notice that by restricting the problem to linear generalizations we reduce the anti-unification type from nullary to finitary. This fact aligns with intuition, but such restrictions do not always reduce the type. For instance, the following example provided by Temur Kutsia and David Cerna behaves differently.

Example 1.4.18. Consider the anti-unification problem of terms $f(a, h(a))$ and $f(b, h(b))$ over the theory induced by the axiom $\{f(f(x, h(x)), h(x)) \approx f(x, h(x))\}$. Notice that this problem has only one `lgg`, the term $f(x, h(x))$, but when we restrict our attention to the linear variant, the problem becomes nullary. In fact, it is not possible to construct a minimal complete set of generalizations for the terms $f(a, h(a))$ and $f(b, h(b))$, since all non-trivial linear generalization is either of the form $f(x, y)$, or can be generated recursively by applying the substitution $\{x \mapsto f(x, h(z))\}$, where z is a fresh variable, to the term $f(x, h(y))$. Then, for any complete set we can find an infinite chain of linear generalizations. Indeed, any infinite chain is formed by some sequence of terms $x, f(x, y), f(x, h(y)), f(f(x, h(z)), h(y)), f(f(f(x, h(w)), h(z)), h(y)), \ldots$, implying that the linear variant of this theory is nullary. $\diamond$

Chapter 2

A Sound Algorithm for Absorptive Anti-Unification Problem

This section introduces the two processes developed to compute generalizations for terms over the equational theory $\mathfrak{a}$: the $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ algorithm in Section 2.1 and the set of abstraction substitutions Ψ in Section 2.2. First, we introduce $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$, a rule-based algorithm over configurations, a structure that encodes anti-unification problems. After that, the notions of abstraction sets and abstraction substitutions are defined as tools for computing more specific generalizations than those generated by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$. Finally, Section 2.3 presents a proof of the soundness of the two procedures using the termination and preservation configuration, which is also proved in this chapter.

2.1 Generalization Algorithm for $\mathfrak{a}$-Theories

Absorptive is a fundamental property in many algebraic structures, such as semirings, rings, and Boolean algebras, where the additive unity acts as an absorbing constant for the product. Although prior work on anti-unification over $\mathfrak{a}$-theories has focused on rich algebraic settings (e.g., semirings [23]), this study explores pure $\mathfrak{a}$-theories with one or more $\mathfrak{a}$-symbols, as introduced in Definition 1.2.10, as part of a broader investigation into anti-unification in subterm-collapsing theories.

Definition 2.1.1 (Configuration)**.** A *configuration*, denoted as $\mathcal{C}$, is a quadruple of the form $\langle A; S; D; \theta \rangle$, where:

- A is a valid set of non-wild AUTs (*active set*);
- S is a valid set of solved AUTs (*store*);
- D is a valid set of AUTs that are either *wild* or of the form $\varepsilon_f \triangleq_x \varepsilon_f$, for some label x and ε_f an absorbing constant (*delayed set*);
- θ is a *substitution* (*computed substitution*);

and such that the following conditions are satisfied:

1. $\mathcal{V}ran(\theta) = labels(A) \cup labels(S) \cup labels(D)$;
2. $labels(A), labels(S), labels(D)$, and $dom(\theta)$ are pairwise disjoint;
3. all terms occurring in a configuration are in their $\mathfrak{a}$-normal forms.

Remark. Notice that θ is idempotent by conditions 1 and 2 in the definition above.

Each configuration component stores the information required to compute specific generalizations for a set of term pairs. The *active set* contains the AUTs with the problems to be solved. The *store* has the AUTs containing a pair of terms in which the most specific generalization is a variable. The *delayed set* stores the AUTs related to the subterms in expanding an absorbing constant. The *computed substitution* is a substitution mapping the labels of the AUEs to their respective generalizations.

Remark. Since the goal of the algorithm is to compute generalizations from a given set of pairs of terms, we associate a configuration to these problems, the *initial configuration* of the form $\langle A; \emptyset; \emptyset; \iota \rangle$, where A is a valid set of AUTs, each AUT associated a label to a pair of terms. In addition, ι the *starting substitution* is a disjoint renaming, with $ran(\iota) = labels(A)$ and $dom(\iota)$ a set of fresh variables. The stated substitution ι is built this way to guarantee the configuration definition.

For the rest of the work, we only consider configurations derived from initial configurations.

Example 2.1.2. Consider the following quadruple $\langle \{a \triangleq_{w_1} b, \varepsilon_f \triangleq_{w_2} c\}; \emptyset; \emptyset; \theta \rangle$, where $\theta = \{x_1 \mapsto g(w_1, w_2), y_1 \mapsto g(w_1, w_2)\}$, and x_1 and y_1 denote the labels of the previous configurations from which this configuration was derived. This is indeed a configuration, since every set of AUTs is valid, $\mathcal{V}ran(\theta) = \{w_1, w_2\}$ coincides exactly with the set of all labels, the labels are pairwise distinct, and the variables in the domain of the computed substitution are not labels of the configuration. ◇

Definition 2.1.3 (Substitution Generalization of a configuration)**.** Let $\mathcal{C} = \langle A; S; D; \theta \rangle$ be a configuration. The substitution γ is called the substitution generalization of $\mathcal{C}$ if $\gamma \in \mathcal{G}_{\mathfrak{a}}(A \cup S \cup D)$ and the set of substitutions generalizations of $\mathcal{C}$ is denoted as $\mathcal{G}_{\mathfrak{a}}(\mathcal{C})$.

Example 2.1.4. Let $\mathcal{C} = \langle \{h(a) \triangleq_x h(b)\}; \{\varepsilon_f \triangleq_y d\}; \{\star \triangleq_z h(\varepsilon_f)\}; \{w \mapsto g(x, f(y, z))\} \rangle$ be a configuration. The substitution $\gamma = \{x \mapsto h(x'), y \mapsto y', z \mapsto h(f(y', z'))\}$ is a substitution generalization of $\mathcal{C}$. Indeed, note that the associated substitutions of γ are $\rho_l = \{x' \mapsto a, y' \mapsto \varepsilon_f, z' \mapsto \varepsilon_f\}$ and $\rho_r = \{x' \mapsto b, y' \mapsto d, z' \mapsto \varepsilon_f\}$, it holds that for all the non-wild AUTs $\mathtt{a} \in A \cup S$, $label(\mathtt{a})\gamma\rho_l \approx_{\mathfrak{a}} lhs(\mathtt{a})$ and $label(\mathtt{a})\gamma\rho_r \approx_{\mathfrak{a}} rhs(\mathtt{a})$, and for the wild AUT it holds that $h(f(y', z'))\rho_r \approx_{\mathfrak{a}} h(\varepsilon_f)$. ◇

Table 2.1: Generalization $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ rules for $\mathfrak{a}$-theory.

$$(\overset{Dec}{\Longrightarrow}) \quad \frac{\langle \{f(s_1,\ldots,s_n) \triangleq_x f(t_1,\ldots,t_n)\} \uplus A; S; D; \theta\rangle}{\langle \{s_1 \triangleq_{y_1} t_1,\ldots,s_n \triangleq_{y_n} t_n\} \cup A; S; D; \theta\{x \mapsto f(y_1,\ldots,y_n)\}\rangle}$$

where f is an n-ary symbol, $n \geq 0$, and $y_1,\ldots,y_n$ are fresh variables.

$$(\overset{Sol}{\Longrightarrow}) \quad \frac{\langle \{s \triangleq_x t\} \uplus A; S; D; \theta\rangle}{\langle A; \{s \triangleq_x t\} \cup S; D; \theta\rangle}$$

where $s \triangleq_x t$ is a solved AUT.

$$(\overset{Mer}{\Longrightarrow}) \quad \frac{\langle \emptyset; \{s \triangleq_x t, s \triangleq_y t\} \uplus S; D; \theta\rangle}{\langle \emptyset; \{s \triangleq_y t\} \cup S; D; \theta\{x \mapsto y\}\rangle}$$

$$(\overset{ExpB}{\Longrightarrow}) \quad \frac{\langle \{\varepsilon_f \triangleq_x \varepsilon_f\} \uplus A; S; D; \theta\rangle}{\langle A; S; \{\varepsilon_f \triangleq_x \varepsilon_f\} \cup D; \theta\rangle}$$

where ε_f is an absorbing constant.

For the following rules, f is an absorptive symbol and y_1, y_2 are fresh variables:

$$(\overset{ExpLA1}{\Longrightarrow}) \quad \frac{\langle \{\varepsilon_f \triangleq_x f(t_1,t_2)\} \uplus A; S; D; \theta\rangle}{\langle \{\varepsilon_f \triangleq_{y_1} t_1\} \cup A; S; \{\star \triangleq_{y_2} t_2\} \cup D; \theta\{x \mapsto f(y_1,y_2)\}\rangle}$$

$$(\overset{ExpLA2}{\Longrightarrow}) \quad \frac{\langle \{\varepsilon_f \triangleq_x f(t_1,t_2)\} \uplus A; S; D; \theta\rangle}{\langle \{\varepsilon_f \triangleq_{y_2} t_2\} \cup A; S; \{\star \triangleq_{y_1} t_1\} \cup D; \theta\{x \mapsto f(y_1,y_2)\}\rangle}$$

$$(\overset{ExpRA1}{\Longrightarrow}) \quad \frac{\langle \{f(s_1,s_2) \triangleq_x \varepsilon_f\} \uplus A; S; D; \theta\rangle}{\langle \{s_1 \triangleq_{y_1} \varepsilon_f\} \cup A; S; \{s_2 \triangleq_{y_2} \star\} \cup D; \theta\{x \mapsto f(y_1,y_2)\}\rangle}$$

$$(\overset{ExpRA2}{\Longrightarrow}) \quad \frac{\langle \{f(s_1,s_2) \triangleq_x \varepsilon_f\} \uplus A; S; D; \theta\rangle}{\langle \{s_2 \triangleq_{y_2} \varepsilon_f\} \cup A; S; \{s_1 \triangleq_{y_1} \star\} \cup D; \theta\{x \mapsto f(y_1,y_2)\}\rangle}$$

Definition 2.1.5 (Algorithm $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$)**.** $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ is a rule-based algorithm over configurations. $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ consists of an exhaustive application of the rules presented in Table 2.1 (the symbol $\uplus$ stands for disjoint union) in any possible manner, i.e., it is possible to apply a rule to any AUT in its active set, and in some cases, it is possible to use two or more rules to the same AUT creating branches.

By $\mathcal{C} \Longrightarrow \mathcal{C}'$ we denote the application of some inference rule of Table 2.1 to $\mathcal{C}$ resulting in $\mathcal{C}'$. By $\mathcal{C} \Longrightarrow^* \mathcal{C}'$ we denote a finite sequence of inference rule applications starting at $\mathcal{C}$ and ending with $\mathcal{C}'$. In both cases we say $\mathcal{C}'$ is *derived* from $\mathcal{C}$.

Each rule transforms configurations into configurations (see Lemma 2.1.11). A detailed explanation of each rule introduced in the Table 2.1 is presented below).

(Dec): **Decompose**

$$\langle \{f(s_1,\ldots,s_n) \triangleq_x f(t_1,\ldots,t_n)\} \uplus A; S; D; \theta\rangle \overset{Dec}{\Longrightarrow}$$
$$\langle \{s_1 \triangleq_{y_1} t_1,\ldots,s_n \triangleq_{y_n} t_n\} \cup A; S; D; \theta\{x \mapsto f(y_1,\ldots,y_n)\}\rangle,$$

where f is an n-ary symbol, $n \geq 0$, and $y_1, \ldots, y_n$ are fresh variables.

Explanation: The *Decompose* rule applies when the terms in the selected AUT share the same head symbol. In this case, the generalization variable x, which initially stands for a generalization of the terms $f(s_1, \ldots, s_n)$ and $f(t_1, \ldots, t_n)$, is refined in θ to the more specific term $f(y_1, \ldots, y_n)$. Each y_i is a fresh variable intended to generalize the corresponding pair of subterms s_i and t_i, for $1 \leq i \leq n$.

Example 2.1.6. Consider g and f binary function symbols, a and ε_f constants with f and ε_f related absorptive symbols. We want to compute a generalization for the terms $g(\varepsilon_f, a)$ and $g(f(h(\varepsilon_f), a), \varepsilon_f)$. The associated initial configuration is given by $\langle \{g(\varepsilon_f, a) \triangleq_x g(f(h(\varepsilon_f), a), \varepsilon_f)\}; \emptyset; \emptyset; \{x' \mapsto x\}\rangle$. Notice that it is possible to apply the rule (Dec) since $head(g(\varepsilon_f, a)) = g = head(g(f(h(\varepsilon_f), a), \varepsilon_f))$, and obtaining the following configuration $\langle \{\varepsilon_f \triangleq_{y_1} f(h(\varepsilon_f), a), a \triangleq_{y_2} \varepsilon_f\}; \emptyset; \emptyset; \{x' \mapsto f(y_1, y_2), x \mapsto f(y_1, y_2)\}\rangle$. $\diamond$

Remark. From now, the subset of bindings of the computed substitution related to the initial configuration will be omitted.

(Sol): **Solve**

$$\langle \{s \triangleq_x t\} \uplus A; S; D; \theta\rangle \overset{Sol}{\Longrightarrow} \langle A; \{s \triangleq_x t\} \cup S; D; \theta\rangle,$$

where $s \triangleq_x t$ is a solved AUT.

Explanation: The *Solve* rule is applied when the heads of the terms in the selected AUT are different and unrelated under the theory $\mathfrak{a}$. Since it is not possible to compute a more specific generalization than the current label x, the rule simply moves the AUT to the store.

Example 2.1.7. Considering the configuration obtained in Example 2.1.6, a solve rule is applied over the second AUT, since $head(a) \neq head(\varepsilon_f)$, then this AUT is solved and the rule move it to the store generating the following configuration:

$$\langle \{\varepsilon_f \triangleq_{y_1} f(h(\varepsilon_f)), a)\}; \{a \triangleq_{y_2} \varepsilon_f\}; \emptyset; \{x \mapsto g(y_1, y_2)\}\rangle.$$

$\diamond$

(Mer): **Merge**

$$\langle \emptyset; \{s \triangleq_x t, s \triangleq_y t\} \uplus S; D; \theta\rangle \overset{Mer}{\Longrightarrow} \langle \emptyset; \{s \triangleq_y t\} \cup S; D; \theta\{x \mapsto y\}\rangle,$$

Explanation: Since the goal is to compute the most specific generalization, the *Merge* rule ensures that all solved AUTs in the store corresponding to the same pair of terms

are associated with the same generalization variable. This rule is applied only when the active set is empty.

Example 2.1.8. Let a, b, c be constants and g a binary function symbol. Consider the configuration $\langle \emptyset; \{a \triangleq_w b, b \triangleq_y c, a \triangleq_z b\}; \emptyset; \{x \mapsto g(g(w, y), z), w' \mapsto g(w, y)\}\rangle$. Since there are two AUTs: $a \triangleq_w b$ and $a \triangleq_z b$ in the store, where the left and right terms in the respective triples are identical then the (Mer) is applied, generating the configuration

$$\langle \emptyset; \{b \triangleq_y c, a \triangleq_z b\}; \emptyset; \{x \mapsto g(g(z, y), z), w' \mapsto g(z, y), w \mapsto z\}\rangle$$

$\diamond$

(ExpB): **Expansion Absorptive in Both Sides**

$$\langle \{\varepsilon_f \triangleq_x \varepsilon_f\} \uplus A; S; D; \theta\rangle \overset{ExpB}{\Longrightarrow} \langle A; S; \{\varepsilon_f \triangleq_x \varepsilon_f\} \cup D; \theta\rangle,$$

where ε_f is an absorbing constant.

Explanation: The rule *expansion absorptive in both sides* handles a special case of expansion, where the absorbing constant appears on both sides of the selected AUT. Specifically, it moves the AUT $\varepsilon_f \triangleq_x \varepsilon_f$ from the active set to the delayed set. This rule, along with the abstraction set introduced in Section 2.2, will be used to capture specific generalizations for certain pairs of terms, as illustrated in Example 2.2.21.

Example 2.1.9. Let g be a function symbol of arity 3, and ε_f and f be related absorptive symbols. Notice that either (ExpB) rule or (Dec) rule can be applied to the configuration

$$\langle \{\varepsilon_f \triangleq_w \varepsilon_f\}; \{\varepsilon_f \triangleq_y a, b \triangleq_z \varepsilon_f\}; \emptyset; \{x \mapsto g(w, y, z)\}\rangle,$$

generating branching into the respectively configurations :

$$\langle \emptyset; \{\varepsilon_f \triangleq_y a, b \triangleq_z \varepsilon_f\}; \{\varepsilon_f \triangleq_w \varepsilon_f\}; \{x \mapsto g(w, y, z)\}\rangle$$

$$\langle \emptyset; \{\varepsilon_f \triangleq_y a, b \triangleq_z \varepsilon_f\}; \emptyset; \{x \mapsto g(\varepsilon_f, y, z), w \mapsto \varepsilon_f\}\rangle.$$

$\diamond$

(ExpLA1): **Expansion for Left Absorptive 1**

$$\langle \{\varepsilon_f \triangleq_x f(t_1, t_2)\} \uplus A; S; D; \theta\rangle \overset{ExpLA1}{\Longrightarrow} \langle \{\varepsilon_f \triangleq_{y_1} t_1\} \cup A; S; \{\star \triangleq_{y_2} t_2\} \cup D; \theta\{x \mapsto f(y_1, y_2)\}\rangle,$$

where f is an $\mathfrak{a}$-symbol and y_1, y_2 are fresh variables.

Explanation: This expansion rule applies when the selected AUT contains an absorbing constant, say ε_f, on the left-hand side. The rule can be interpreted as a shorthand for two steps: first, replacing ε_f with one of its equivalent forms, such as $f(\varepsilon_f, \star)$; and second, decomposing the resulting AUT $f(\varepsilon_f, \star) \triangleq_x f(t_1, t_2)$ into the AUTs $\varepsilon_f \triangleq_{y_1} t_1$ and $\star \triangleq_{y_2} t_2$. This results in the more specific generalization $f(y_1, y_2)$ being added to the substitution for x. After decomposition, the AUT labeled with y_1 is placed in the active set, while the one labeled with y_2 is placed in the delayed set.

(ExpLA2): **Expansion for Left Absorptive 2**

$$\langle \{\varepsilon_f \triangleq_x f(t_1, t_2)\} \uplus A; S; D; \theta \rangle \overset{ExpLA2}{\Longrightarrow}$$
$$\langle \{\varepsilon_f \triangleq_{y_2} t_2\} \cup A; S; \{\star \triangleq_{y_1} t_1\} \cup D; \theta\{x \mapsto f(y_1, y_2)\} \rangle,$$

where f is an $\mathfrak{a}$-symbol and y_1, y_2 are fresh variables.

Explanation: This rule is analogous to (ExpLA1), with the key difference that the selected expansion of ε_f is the term $f(\star, \varepsilon_f)$. The subsequent decomposition produces the AUTs $\star \triangleq_{y_1} t_1$ and $\varepsilon_f \triangleq_{y_2} t_2$. In this case, the rule places the AUT labeled by y_1 into the delayed set, and the AUT labeled by y_2 into the active set.

Example 2.1.10. Following the configuration obtained in Example 2.1.7. Applying (ExpLA1) or (ExpLA2) over the left AUT in the configuration is possible. This application of these rules leads to the following configurations respectively:

$$\langle \{\varepsilon_f \triangleq_{z_1} h(\varepsilon_f)\}; \{a \triangleq_{y_2} \varepsilon_f\}; \{\star \triangleq_{z_2} a\}; \{x \mapsto g(f(z_1, z_2), y_2), y_1 \mapsto f(z_1, z_2)\} \rangle,$$

$$\langle \{\varepsilon_f \triangleq_{z_4} a\}; \{a \triangleq_{y_2} \varepsilon_f\}; \{\star \triangleq_{z_3} h(\varepsilon_f)\}; \{x \mapsto g(f(z_3, z_4), y_2), y_1 \mapsto f(z_3, z_4)\} \rangle.$$

$\diamond$

(ExpRA1): **Expansion for Right absorptive 1**

$$\langle \{f(s_1, s_2) \triangleq_x \varepsilon_f\} \uplus A; S; D; \theta \rangle \overset{ExpRA1}{\Longrightarrow}$$
$$\langle \{s_1 \triangleq_{y_1} \varepsilon_f\} \cup A; S; \{s_2 \triangleq_{y_2} \star\} \cup D; \theta\{x \mapsto f(y_1, y_2)\} \rangle,$$

where f is an absorptive symbol and y_1, y_2 are fresh variables.

(ExpRA2): **Expansion for Right Absorptive 2**

$$\langle \{f(s_1, s_2) \triangleq_x \varepsilon_f\} \uplus A; S; D; \theta \rangle \overset{ExpRA2}{\Longrightarrow}$$
$$\langle \{s_2 \triangleq_{y_2} \varepsilon_f\} \cup A; S; \{s_1 \triangleq_{y_1} \star\} \cup D; \theta\{x \mapsto f(y_1, y_2)\} \rangle,$$

where f is an absorptive symbol and y_1, y_2 are fresh variables.

Explanation: The two rules of Expansion for Right absorptive are analogous to the rules for Expansion for Left Absorptive, with the difference that the absorbing constant ε_f occurs on the right-hand side of the selected AUT.

The four rules of expansion, (ExpLA1), (ExpLA2), (ExpRA1) and (ExpRA2) are called the *lateral expansion rules*. In addition, the rules presented above are summarized in the Table 2.1.

Lemma 2.1.11 (Preservation Configuration)**.** *Any quadruple* $\mathcal{C}' = \langle A'; S'; D'; \theta' \rangle$ *obtained after a rule application over a configuration* $\mathcal{C} = \langle A; S; D; \theta \rangle$ *is a configuration.*

PROOF: The proof is made by case analysis for each rule of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$.

- (Dec): the label of the selected AUT in A is moved to $dom(\theta')$ and is not anymore a label of $\mathcal{C}'$, additionally the introduced AUT has labels of fresh variables, thus $labels(A')$, $labels(S')$, $labels(D')$ and $dom(\theta')$ are pairwise disjoint. In the other hand, the new added term in the range of the substitution has exactly the fresh variables introduced as labels of the new AUTs, implying that $\mathcal{V}ran(\theta') = labels(A') \cup labels(S') \cup labels(D')$.

- (Sol): the selected AUT in A is moved to the store of $\mathcal{C}'$, as there was no introduction of variables, and any set of AUTs is valid then all the properties of configurations stands for $\mathcal{C}'$.

- (Mer): The selected AUT in the store is deleted in the store and the label of this AUT is added to the domain of θ', implying that $labels(A')$, $labels(S')$, $labels(D')$ and $dom(\theta')$ are pairwise disjoint. The variable introduced in the $\mathcal{V}ran(\theta')$ is one of the labels of an AUT in S', then $\mathcal{V}ran(\theta') = labels(A') \cup labels(S') \cup labels(D')$.

- (ExpB): the selected AUT in A is moved to the delayed set of $\mathcal{C}'$, since there was no introduction of variables, and as any set of AUT of $\mathcal{C}'$ is valid, then all the properties of configuration holds for $\mathcal{C}'$.

- (ExpLA1): the label of the selected AUT in A is added to the $dom(\theta')$, additionally the labels in the introduced AUT are fresh, thus $labels(A')$, $labels(S')$, $labels(D')$ and $dom(\theta')$ are pairwise disjoint. In the other hand, the new added term in the range of the substitution has exactly the fresh variables introduced as labels of the new AUTs, implying that $\mathcal{V}ran(\theta') = labels(A') \cup labels(S') \cup labels(D')$.

- (ExpLA2), (ExpRA1), (ExpRA2) are analogous to the (ExpLA1) case. □

Definition 2.1.12 (Final Configuration)**.** A configuration $\mathcal{C}$ is called a *final configuration* if its active set is empty and its store is merged. We usually denote a final configuration by $\mathcal{C}_f$.

Lemma 2.1.13 (Final Configuration Characterization)**.** *A configuration $\mathcal{C}$ is a final configuration if and only if no inference rule of $\mathcal{A}_{\mathfrak{a}\text{-}AU_{NIF}}$ can be applied to $\mathcal{C}$.*

Proof: Suppose that $\mathcal{C}$ is a final configuration, i.e., $\mathcal{C}$ has an empty active set and a merged store; as the active set is empty, then the only possible rule that can be applied is (Mer), but from the Definition 1.3.28, this is not possible; then no inference rule can be applied to $\mathcal{C}$.
Now we assume that no rule of Table 2.1 can be applied to $\mathcal{C}$, since the rules cover all the cases for any pair of terms in an AUT in the active set, then it should be empty, and as (Mer) rule can not be applied then the store should be merged, it means that $\mathcal{C}$ is a final configuration. □

Theorem 2.1.14 (Termination)**.** *$\mathcal{A}_{\mathfrak{a}\text{-}AU_{NIF}}$ is terminating for any configuration $\mathcal{C}$ and it outputs a finite set of configurations.*

Proof: Let $\mathcal{C} = \langle A; S; D; \theta \rangle$. We define $\varphi(\mathcal{C}) := (size(A), size(S))$ as termination measure and compare these pairs lexicographically. This ordering is well-founded since the size of a set of AUTs is a natural number. By case analysis from any rule application $\mathcal{C} \Longrightarrow \mathcal{C}'$, every rule application decreases the number of symbols in A except (Mer) rule, which preserves the size of the active set and decrease the number of symbols in the respective store, implying that $\varphi(\mathcal{C}) >_{lex} \varphi(\mathcal{C}')$ in each case. Thus, every sequence of rule applications terminates. Furthermore, any configuration can be transformed by rules from Table 2.1 in finitely many ways. Thus, by König's Lemma [12], the output is a finite set of configurations. □

Definition 2.1.15 (Set of Final Configurations)**.** Let $\mathcal{C}$ be a configuration. We define the *set of final configurations* reached from $\mathcal{C}$ as

$$\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}) = \{\mathcal{C}_f \mid \mathcal{C} \Longrightarrow^* \mathcal{C}_f, \text{ such that } \mathcal{C}_f \text{ is a final configuration}\}$$

From termination, Theorem 2.1.14 and Lemma 2.1.13, the set $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$ is finite for each configuration $\mathcal{C}$. Each final configuration, $\mathcal{C}_f \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$, contains the information for building generalizations of any equation in an AUT $\mathtt{a}$ of the active set A' of any configuration $\mathcal{C}'$ in the derivation from $\mathcal{C}$ to $\mathcal{C}_f$: $\mathcal{C} \Longrightarrow^* \mathcal{C}' \Longrightarrow^* \mathcal{C}_f$. Indeed, from the

Soundness Theorem 2.3.2 presented in the next section, we observe that $label(\mathtt{a})\theta_f$ is a generalization of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$, where θ_f is the computed substitution of $\mathcal{C}_f$.

We illustrate how $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ computes each final configuration and its respective generalization in the two examples below.

Example 2.1.16. Consider terms $g(\varepsilon_f, a)$ and $g(f(h(\varepsilon_f), a), \varepsilon_f)$ of Example 2.1.6. The associated initial configuration is $\mathcal{C} = \langle\{g(\varepsilon_f, a) \triangleq_x g(f(h(\varepsilon_f), a), \varepsilon_f)\}; \emptyset; \emptyset; \{x' \mapsto x\}\rangle$. Applying exhaustively $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ over this configuration, we get the following two derivations that lead to two final configurations:

$$\begin{aligned}
\textbf{Derivation } 1: \quad & \langle\{g(\varepsilon_f, a) \triangleq_x g(f(h(\varepsilon_f), a), \varepsilon_f)\}; \emptyset; \emptyset; \{x' \mapsto x\}\rangle \overset{Dec}{\Longrightarrow} \\
& \langle\{\varepsilon_f \triangleq_{w_1} f(h(\varepsilon_f), a), a \triangleq_{w_2} \varepsilon_f\}; \emptyset; \emptyset; \{x \mapsto g(w_1, w_2)\}\rangle \overset{ExpLA1}{\Longrightarrow} \\
& \langle\{\varepsilon_f \triangleq_{y_1} h(\varepsilon_f), a \triangleq_{w_2} \varepsilon_f\}; \emptyset; \{\star \triangleq_{y_2} a\}; \\
& \{x \mapsto g(f(y_1, y_2), w_2), w_1 \mapsto f(y_1, y_2)\}\rangle \overset{Sol \times 2}{\Longrightarrow} \\
& \langle\emptyset; \{\varepsilon_f \triangleq_{y_1} h(\varepsilon_f), a \triangleq_{w_2} \varepsilon_f\}; \{\star \triangleq_{y_2} a\}; \\
& \{x \mapsto g(f(y_1, y_2), w_2), w_1 \mapsto f(y_1, y_2)\}\rangle.
\end{aligned}$$

The second derivation is generated after an application of the (ExpLA2) in the configuration derived from decomposition:

$$\begin{aligned}
\textbf{Derivation } 2: \quad & \langle\{g(\varepsilon_f, a) \triangleq_x g(f(h(\varepsilon_f), a), \varepsilon_f)\}; \emptyset; \emptyset; \{x' \mapsto x\}\rangle \overset{Dec}{\Longrightarrow} \\
& \langle\{\varepsilon_f \triangleq_{w_1} f(h(\varepsilon_f), a), a \triangleq_{w_2} \varepsilon_f\}; \emptyset; \emptyset; \{x \mapsto g(w_1, w_2)\}\rangle \overset{ExpLA2}{\Longrightarrow} \\
& \langle\{\varepsilon_f \triangleq_{y_4} a, a \triangleq_{w_2} \varepsilon_f\}; \emptyset; \{\star \triangleq_{y_3} h(\varepsilon_f)\}; \\
& \{x \mapsto g(f(y_3, y_4), w_2), w_1 \mapsto f(y_3, y_4)\}\rangle \overset{Sol \times 2}{\Longrightarrow} \\
& \langle\emptyset; \{\varepsilon_f \triangleq_{y_4} a, a \triangleq_{w_2} \varepsilon_f\}; \{\star \triangleq_{y_3} h(\varepsilon_f)\}; \\
& \{x \mapsto g(f(y_3, y_4), w_2), w_1 \mapsto f(y_3, y_4)\}\rangle.
\end{aligned}$$

Then the set of final configurations of $\mathcal{C}$ is given by

$$\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}) = \left\{ \begin{array}{l} \langle\emptyset; \{\varepsilon_f \triangleq_{y_1} h(\varepsilon_f), a \triangleq_{w_2} \varepsilon_f\}; \{\star \triangleq_{y_2} a\}; \theta_1\rangle, \\ \langle\emptyset; \{\varepsilon_f \triangleq_{y_4} a, a \triangleq_{w_2} \varepsilon_f\}; \{\star \triangleq_{y_3} h(\varepsilon_f)\}; \theta_2\rangle \end{array} \right\},$$

where

$$\begin{aligned}
\theta_1 &= \{x \mapsto g(f(y_1, y_2), w_2), w_1 \mapsto f(y_1, y_2)\} \\
\theta_2 &= \{x \mapsto g(f(y_3, y_4), w_2), w_1 \mapsto f(y_3, y_4)\}.
\end{aligned}$$

Applying the computed substitutions θ_1 and θ_2 in the label of the initial configuration, we obtain $x\theta_1 = g(f(y_1, y_2), w_2)$ and $x\theta_2 = g(f(y_3, y_4), w_2)$. These terms are generalizations of their respective terms. $\diamond$

Example 2.1.17. Let g be a 3-ary function symbol and f and ε_f related absorptive symbols. Consider the pair terms $g(\varepsilon_f, \varepsilon_f, a)$ and $g(\varepsilon_f, b, \varepsilon_f)$. The associated initial configuration $\mathcal{C} = \langle\{g(\varepsilon_f, \varepsilon_f, a) \triangleq_x g(\varepsilon_f, b, \varepsilon_f)\}; \emptyset; \emptyset; \iota\rangle$, applying $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ over this configuration generates two derivations and leads to the final configurations as follows.

$$\begin{aligned}
\textbf{Derivation } 1: \quad & \langle\{g(\varepsilon_f, \varepsilon_f, a) \triangleq_x g(\varepsilon_f, b, \varepsilon_f)\}; \emptyset; \emptyset; \{x' \mapsto x\}\rangle \overset{Dec}{\Longrightarrow} \\
& \langle\{\varepsilon_f \triangleq_{w_1} \varepsilon_f, \varepsilon_f \triangleq_{w_2} b, a \triangleq_{w_3} \varepsilon_f\}; \emptyset; \emptyset; \{x \mapsto g(w_1, w_2, w_3)\}\rangle \overset{Dec}{\Longrightarrow} \\
& \langle\{\varepsilon_f \triangleq_{w_2} b, a \triangleq_{w_3} \varepsilon_f\}; \emptyset; \emptyset; \{x \mapsto g(\varepsilon_f, w_2, w_3), w_1 \mapsto \varepsilon_f\}\rangle \overset{Sol \times 2}{\Longrightarrow} \\
& \langle\emptyset; \{\varepsilon_f \triangleq_{w_2} b, a \triangleq_{w_3} \varepsilon_f\}; \emptyset; \{x \mapsto g(\varepsilon_f, w_2, w_3), w_1 \mapsto \varepsilon_f\}\rangle.
\end{aligned}$$

$$\begin{aligned}
\textbf{Derivation } 2: \quad & \langle\{g(\varepsilon_f, \varepsilon_f, a) \triangleq_x g(\varepsilon_f, b, \varepsilon_f)\}; \emptyset; \emptyset; \iota\rangle \overset{Dec}{\Longrightarrow} \\
& \langle\{\varepsilon_f \triangleq_{w_1} \varepsilon_f, \varepsilon_f \triangleq_{w_2} b, a \triangleq_{w_3} \varepsilon_f\}; \emptyset; \emptyset; \{x \mapsto g(w_1, w_2, w_3)\}\rangle \overset{ExpB}{\Longrightarrow} \\
& \langle\{\varepsilon_f \triangleq_{w_2} b, a \triangleq_{w_3} \varepsilon_f\}; \emptyset; \{\varepsilon_f \triangleq_{w_1} \varepsilon_f\}; \{x \mapsto g(w_1, w_2, w_3)\}\rangle \overset{Sol \times 2}{\Longrightarrow} \\
& \langle\emptyset; \{\varepsilon_f \triangleq_{w_2} b, a \triangleq_{w_3} \varepsilon_f\}; \{\varepsilon_f \triangleq_{w_1} \varepsilon_f\}; \{x \mapsto g(w_1, w_2, w_3)\}\rangle.
\end{aligned}$$

The set of final configurations:

$$\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}) = \left\{ \begin{array}{l} \langle\emptyset; \{\varepsilon_f \triangleq_{w_2} b, a \triangleq_{w_3} \varepsilon_f\}; \emptyset; \theta_1\rangle, \\ \langle\emptyset; \{\varepsilon_f \triangleq_{w_2} b, a \triangleq_{w_3} \varepsilon_f\}; \{\varepsilon_f \triangleq_{w_1} \varepsilon_f\}; \theta_2\rangle \end{array} \right\},$$

where

$$\begin{aligned}
\theta_1 &= \{x \mapsto g(\varepsilon_f, w_2, w_3), w_1 \mapsto \varepsilon_f\}, \\
\theta_2 &= \{x \mapsto g(w_1, w_2, w_3)\}.
\end{aligned}$$

Notice that the terms $x\theta_1 = g(\varepsilon_f, w_2, w_3)$, and $x\theta_2 = g(w_1, w_2, w_3)$ are generalizations of $g(\varepsilon_f, \varepsilon_f, a)$ and $g(\varepsilon_f, b, \varepsilon_f)$. ◇

Remark. Notice that each rule application introduces fresh variables concerning all previously computed configurations.

2.2 Abstraction Set and Substitutions

We construct the *abstraction set* and *abstraction substitutions* from the store and delayed sets of the final configurations derived by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ algorithm. Let $\langle A; \emptyset; \emptyset; \iota\rangle$ be an initial configuration and $\langle\emptyset; S; D; \theta\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\langle A; \emptyset; \emptyset; \iota\rangle)$. While for $\mathtt{a} \in A$, the term $\mathit{label}(\mathtt{a})\theta$

may be more specific than the syntactic generalization of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$ as was illustrated in Examples 2.1.16 and 2.1.17, any use of the $\mathfrak{a}$-theory while computing $label(\mathtt{a})\theta$ is completely dependent on the presence of $\mathfrak{a}$-symbols and absorbing constants within $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$. Absorptive theories allow the introduction of additional structure beyond what is present in the initial AUTs. For example, $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ computes the generalization $f(x, y)$ for the terms ε_f and $f(h(\varepsilon_f), h(h(\varepsilon_f)))$, yet $\mathfrak{a}$-theory allows a more specific generalization, $f(x, h(x))$. In more extreme cases, infinitely many more specific generalizations may exist.

Definition 2.2.1 (Abstraction Set)**.** Let t be a ground term in $\mathfrak{a}$-normal form, and σ be a substitution whose range is in $\mathfrak{a}$-normal form. The *abstraction set of* t with respect to σ is the set

$$\uparrow(t, \sigma) := \{r \mid r\sigma \approx_{\mathfrak{a}} t,\ r \text{ is in } \mathfrak{a}\text{-normal form, and } var(r) \subseteq dom(\sigma)\}.$$

Observe that $t \in \uparrow(t, \sigma)$ since $var(t) = \emptyset \subseteq dom(\sigma)$ and $t\sigma = t$. To obtain an $r \in \uparrow(t, \sigma)$, we abstract some occurrences of some $x\sigma$'s in t by x, where $x \in dom(\sigma)$; this is the origin of the term "abstraction set".

Example 2.2.2. Let $t = g(\varepsilon_f, f(h(a), b))$ and $\sigma = \{x \mapsto a, y \mapsto f(h(a), b), z \mapsto b\}$. Then the abstraction set of t with respect to σ is

$$\uparrow(t, \sigma) = \{t,\ g(\varepsilon_f, y),\ g(\varepsilon_f, f(h(x), b)),\ g(\varepsilon_f, f(h(a), z)),\ g(\varepsilon_f, f(h(x), z))\}.$$

◇

Example 2.2.3. Let h be a unary function symbol and ε_f an absorbing constant. Consider $t = h(\varepsilon_f)$ and $\sigma = \{y \mapsto a, z \mapsto \varepsilon_f\}$. The abstraction set of t with respect to σ is

$$\uparrow(t, \sigma) = \{h(\varepsilon_f),\ h(z), h(f(z, z)), h(f(z, y)), h(f(a, z)), \ldots\}$$

Observe that $\uparrow(t, \sigma)$ contains infinitely many terms. Indeed, any term $s = h(u)$, where u is a z-collapsible $\mathfrak{a}$-normal term, and whose variables are restricted to the set $dom(\sigma)$, belongs to $\uparrow(t, \sigma)$. ◇

The abstraction set operator can be described in an algorithmic way. This construction is based on regular tree grammars (see Definition 1.3.37). We assume a finite set of function symbols $\mathcal{F}$ and countably infinite sets $\mathcal{V}$ and $\mathcal{N}$ of variables and *non-terminal* symbols, respectively. Moreover, let $\mathfrak{S}, \mathfrak{C}, \mathfrak{N} \subset \mathcal{N}$ be pairwise disjoint subsets such that we associate to each term $r \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F})$ a non-terminal symbol in $\mathfrak{S}$, denoted as S^r, and to each absorptive symbol f a non-terminal symbol C_f in $\mathfrak{C}$ and N_f in $\mathfrak{N}$, respectively.

Given a set of non-terminal symbols $N \subset \mathcal{N}$ and a ranked alphabet $\Sigma \subseteq \mathcal{F}$. Additionally, we assume that all the substitutions are ground.

Definition 2.2.4 ((t,σ)-Abstracted Positions)**.** Let $t \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F})$, σ be a substitution, the set of (t,σ)-*abstracted positions* $P^t_\sigma = \{p \in pos(t) - \{\epsilon\} \mid t|_p \in ran(\sigma)\}$ is the set of positions of the proper subterms of t which belong to $ran(\sigma)$. The set of (t,σ)-*non-terminals* is defined as $\mathfrak{S}^t_\sigma = \{S^r \in \mathfrak{S} \mid r \in ran(\sigma) \cap subt(t)\}$.

Notice that for any term $r \in ran(\sigma) \cap subt(t)$ there exists $p \in P^t_\sigma \cup \{\epsilon\}$ such that $t|_p = r$. We associate a non-terminal symbol from $\mathfrak{S}$ to any subterm of t which is in $ran(\sigma)$.

Definition 2.2.5 ((t,σ)-Maximal Positions)**.** We refer to $p \in P^t_\sigma$ as (t,σ)-*maximal* if there does not exists q with $p \sqsubset q$ and $q \in P^t_\sigma$. The set of (t,σ)-maximal positions is denoted by $max_{\sqsubseteq}(t,\sigma)$.

Example 2.2.6. Let $t = g(h(h(a)), h(a))$ and $\sigma = \{x \mapsto h(h(a)), y \mapsto h(a), z \mapsto a\}$. Notice that $ran(\sigma) \cap subtp(t) = \{h(h(a)), h(a), a\}$ then the positions of t where these subterms occurs are $P^t_\sigma = \{1, 2, 1.1, 1.1.1, 2.1\}$, the maximal positions $max_{\sqsubseteq}(t,\sigma) = \{1, 2\}$, and the associated (t,σ)-non-terminals $\mathfrak{S}^t_\sigma = \{S^{h(h(a))}, S^{h(a)}, S^a\}$. ◇

Definition 2.2.7 (σ-Lifting)**.** Let $t \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F})$ and σ be a substitution. The σ-*lifting* of t is the metaterm $l(t,\sigma) \in \mathcal{T}(\mathcal{F},\mathcal{N})$, defined as follows $pos(l(t,\sigma)) = (pos(t) - P^t_\sigma) \cup max_{\sqsubseteq}(t,\sigma)$, and for all $p \in max_{\sqsubseteq}(t,\sigma)$, we have $l(t,\sigma)|_p = S^{t|_p}$ and for $q \in pos(t) - P^t_\sigma$ we have that $head(l(t,\sigma)|_q) = head(t|_q)$.

Example 2.2.8. Consider the term t and the substitution σ of the Example 2.2.6. The σ-lifting of $g(h(h(a)), h(a))$ is equal to $g(S^{h(h(a))}, S^{h(a)})$, since for $1, 2 \in max_{\sqsubseteq}(t,\sigma)$ we have that $l(t,\sigma)|_1 = S^{h(h(a))}$, $l(t,\sigma)|_2 = S^{h(a)}$ and for $\epsilon \in pos(t) - P^t_\sigma$ it holds that $head(l(t,\sigma)|_\epsilon) = g = head(t|_\epsilon)$. ◇

Definition 2.2.9 ((t,σ)-Abstraction Grammar)**.** Let σ be a substitution, $t \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F})$. The (t,σ)-*abstraction grammar* is given by $\mathfrak{G}(t,\sigma) = \langle S_0; N; \Sigma; R\rangle$ and defined as follows:

- If there exists an absorbing constant $\varepsilon_f \in ran(\sigma) \cap subt(t)$, $N = \{S_0, T\} \cup \mathfrak{S}^t_\sigma \cup \mathfrak{C}^t_\sigma \cup \mathfrak{N}$, where $\mathfrak{C}^t_\sigma = \{C_f \in \mathfrak{C} \mid \varepsilon_f \in ran(\sigma) \cap subt(t)\}$, $\Sigma = \mathcal{F} \cup dom(\sigma)$, and R contains exactly the following production rules:
  - $S_0 \longrightarrow S^t$ if $t \in ran(\sigma)$,
    $S_0 \longrightarrow l(t,\sigma)$, otherwise;
  - $S^{\varepsilon_f} \longrightarrow \varepsilon_f \mid C_f$, if $S^{\varepsilon_f} \in \mathfrak{S}^t_\sigma$;

- $S^r \longrightarrow l(r,\sigma) \mid x$, for $S^r \in \mathfrak{S}^t_\sigma$, $r \neq \varepsilon_f$, and all $x \in dom(\sigma)$ with $x\sigma = r$;
- $C_f \longrightarrow f(C_f, N_f) \mid f(N_f, C_f) \mid x$, for all $C_f \in \mathfrak{C}^t_\sigma$ and $x \in dom(\sigma)$ such that $x\sigma = \varepsilon_f$;
- $N_f \longrightarrow g(N_g, N_g) \mid h(T, \ldots, T) \mid x$, for all $N_f \in \mathfrak{N}$, $x \in dom(\sigma)$, $g \in \texttt{Abs}$, and $h \in \mathcal{F} - (\texttt{Abs} \cup \{\varepsilon_f\})$ with the appropriate number of non-terminals as arguments (depending on the arity of h).
- $T \longrightarrow N_g$, for all the $N_g \in \mathfrak{N}$.

- Otherwise, $N = \{S_0\} \cup \mathfrak{S}^t_\sigma$ and $\Sigma = \mathcal{F} \cup V$, with $V = \{x \in dom(\sigma) \mid x\sigma \in subt(t)\}$. The set R contains exactly the following production rules:
  - $S_0 \longrightarrow S^t$ if $t \in ran(\sigma)$,
    $S_0 \longrightarrow l(t,\sigma)$, otherwise;
  - $S^r \longrightarrow l(r,\sigma) \mid x$, for $S^r \in \mathfrak{S}^t_\sigma$, $r \neq \varepsilon_f$, and all $x \in dom(\sigma)$ with $x\sigma = r$.

Example 2.2.10. Continuing with Example 2.2.8, where $t = g(h(h(a))), h(a))$, $\sigma = \{x \mapsto h(h(a)), y \mapsto h(a), z \mapsto a\}$ and $l(t,\sigma) = g(S^{h(h(a))}, S^{h(a)})$, and considering $\mathcal{F} = \{a, g, h\}$. The abstraction grammar is $\mathfrak{G}(t,\sigma) = \langle S_0; \{S_0, S^a, S^{h(a)}, S^{h(h(a))}\}; \{a, g, h, x, y, z\}; R\rangle$, where R contains the following productions:

$$\begin{aligned} S_0 &\longrightarrow g(S^{h(h(a))}, S^{h(a)}) \\ S^{h(h(a))} &\longrightarrow h(S^{h(a)}) \mid x \\ S^{h(a)} &\longrightarrow h(S^a) \mid y \\ S^a &\longrightarrow a \mid z \end{aligned}$$

The language of $\mathfrak{G}(t,\sigma)$ is finite and contains all the terms listed below:

$$\mathcal{L}(\mathfrak{G}(t,\sigma)) = \left\{ \begin{array}{lll} g(x,y), & g(x,h(z)), & g(x,h(a)), \\ g(h(y),y), & g(h(y),h(z)), & g(h(y),h(a)), \\ g(h(h(z)),y), & g(h(h(z)),h(z)), & g(h(h(z)),h(a)), \\ g(h(h(a)),y), & g(h(h(a)),h(z)), & g(h(h(a)),h(a)) \end{array} \right\}$$

◇

The three examples below illustrate the (t,σ)-abstraction grammar $\mathfrak{G}(t,\sigma)$ which language $\mathcal{L}(\mathfrak{G}(t,\sigma))$ describes all the terms in the abstraction set $\uparrow(t,\sigma)$.

Example 2.2.11. Let $t = g(g(h(a),a),b)$ be a term and $\sigma = \{x \mapsto h(a), y \mapsto a, z \mapsto c\}$ be a substitution. Notice, we have that $P^t_\sigma = \{1.1, 1.2, 1.1.1\}$, $max_\sqsubseteq(t,\sigma) = \{1.1, 1.2\}$, and

$\mathfrak{S}^t_\sigma = \{S^{h(a)}, S^a\}$. The $l(t,\sigma) = g(g(S^{h(a)}, S^a), b)$ since for $1.1, 1.2 \in max_{\sqsubseteq}(t\sigma)$ we have that

$$l(t,\sigma)|_{1.1} = S^{h(a)} = S^{t|_{1.1}},$$
$$l(t,\sigma)|_{1.2} = S^a = S^{t|_{1.2}},$$

and for $\epsilon, 1, 2 \in pos(t) - P^t_\sigma$ we have that

$$head(l(t,\sigma)|_\epsilon) = g = head(t|_\epsilon),$$
$$head(l(t,\sigma)|_1) = g = head(t|_1),$$
$$head(l(t,\sigma)|_2) = b = head(t|_2).$$

Furthermore, the abstraction grammar is $\mathfrak{G}(t,\sigma) = \langle S_0; \{S_0, S^{h(a)}, S^a\}; \{a, b, g, h, x, y\}; R\rangle$, where R contains the following productions:

$$\begin{aligned} S_0 &\longrightarrow g(g(S^{h(a)}, S^a), b) \\ S^{h(a)} &\longrightarrow h(S^a) \mid x \\ S^a &\longrightarrow a \mid y \end{aligned}$$

The language of $\mathfrak{G}(t,\sigma)$ is finite and contains all the terms listed below:

$$\mathcal{L}(\mathfrak{G}(t,\sigma)) = \left\{ \begin{array}{lll} g(g(x,y),b), & g(g(x,a),b), & g(g(h(y),y),b), \\ g(g(h(a),y),b), & g(g(h(y),a),b), & g(g(h(a),a),b) \end{array} \right\}$$

$\diamond$

Example 2.2.12. Let $\mathcal{F} = \{a, \varepsilon_f, \varepsilon_g, f, g, h\}$, $t = h(\varepsilon_f)$ and $\sigma = \{x \mapsto \varepsilon_f, y \mapsto a\}$. Then $P^t_\sigma = max_{\sqsubseteq}(t,\sigma) = \{1\}$, $\mathfrak{S}^t_\sigma = \{S^{\varepsilon_f}\}$, and $l(t,\sigma) = h(S^{\varepsilon_f})$. The abstraction grammar is $\mathfrak{G}(h(\varepsilon_f),\sigma) = \langle S_0; \{S_0, S^{\varepsilon_f}, C_f, N_f, N_g, T\}; \{a, \varepsilon_f, \varepsilon_g, f, g, h, x, y\}; R\rangle$ where R contains the following productions:

$$\begin{aligned} S_0 &\longrightarrow h(S^{\varepsilon_f}) \\ S^{\varepsilon_f} &\longrightarrow \varepsilon_f \mid C_f \\ C_f &\longrightarrow f(C_f, N_f) \mid f(N_f, C_f) \mid x \\ N_f &\longrightarrow a \mid \varepsilon_g \mid f(N_f, N_f) \mid g(N_g, N_g) \mid h(T) \mid x \mid y \\ N_g &\longrightarrow a \mid \varepsilon_f \mid f(N_f, N_f) \mid g(N_g, N_g) \mid h(T) \mid x \mid y \\ T &\longrightarrow a \mid \varepsilon_f \mid \varepsilon_g \mid f(N_f, N_f) \mid g(N_g, N_g) \mid h(T) \mid x \mid y \end{aligned}$$

The language $\mathcal{L}(\mathfrak{G}(h(\varepsilon_f),\sigma))$ has infinitely many terms as $\uparrow(h(\varepsilon_f),\sigma)$. Indeed, note that $S_0 \longrightarrow_{\mathfrak{G}} h(S^{\varepsilon_f}) \longrightarrow_{\mathfrak{G}} h(C_f)$. Given that C_f recursively calls itself (increasing the

depth of the output term), there are infinitely many terms t' such that $C_f \longrightarrow^+_{\mathfrak{G}} t'$. $\diamond$

Example 2.2.13. Let $\mathcal{F} = \{a, \varepsilon_f, \varepsilon_g, f, g, h\}$, $t = \varepsilon_f$, and $\sigma = \{x \mapsto \varepsilon_f, y \mapsto a\}$. The abstraction grammar is $\mathfrak{G}(\varepsilon_f, \sigma) = \langle S_0; \{S_0, S^{\varepsilon_f}, C_f, N_f, N_g, T\}; \mathcal{F} \cup \{x, y\}; R\rangle$ where R contains the following productions:

$$\begin{array}{rcl} S_0 & \longrightarrow & S^{\varepsilon_f} \\ S^{\varepsilon_f} & \longrightarrow & \varepsilon_f \mid C_f \\ C_f & \longrightarrow & f(C_f, N_f) \mid f(N_f, C_f) \mid x \\ N_f & \longrightarrow & a \mid \varepsilon_g \mid f(N_f, N_f) \mid g(N_g, N_g) \mid h(T) \mid x \mid y \\ N_g & \longrightarrow & a \mid \varepsilon_f \mid f(N_f, N_f) \mid g(N_g, N_g) \mid h(T) \mid x \mid y \\ T & \longrightarrow & N_f \mid N_g \end{array}$$

$\diamond$

Lemma 2.2.14. *Let $\mathfrak{G}(t, \sigma) = \langle S_0; N; \Sigma; R\rangle$ be the (t, σ)-abstraction grammar such that $\varepsilon_f \in subt(t) \cap ran(\sigma)$. Then, the following holds for all $N_f, T \in N$:*

- $\mathcal{L}(\mathfrak{G}(t, \sigma), N_f) = \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma)) - \{\varepsilon_f\}$;
- $\mathcal{L}(\mathfrak{G}(t, \sigma), T) = \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$.

Proof: We prove the soundness and completeness for both languages.

Soundness

We need to prove that s is a $\mathfrak{a}$-normal and different from ε_f if $N_f \longrightarrow^*_{\mathfrak{G}} s$. We use induction on the structure of s:

- If $s = x \in dom(\sigma)$ or $s = c \neq \varepsilon_f$, i.e., we have the one step derivation $N_f \longrightarrow_{\mathfrak{G}} s$. In both cases $s \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma)) - \{\varepsilon_f\}$.
- If $s = g(s_1, s_2)$, where $g \in \mathtt{Abs}$, i.e., $N_f \longrightarrow_{\mathfrak{G}} g(N_g, N_g) \longrightarrow^+_{\mathfrak{G}} g(s_1, s_2)$. By induction hypothesis we have that $s_1, s_2 \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma)) - \{\varepsilon_g\}$. This implies that $g(s_1, s_2) \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma)) - \{\varepsilon_g\}$.
- If $s = h(u_1, \ldots, u_m)$, where $h \in \mathcal{F} - (\mathtt{Abs} \cup \{\varepsilon_f\})$ and arity $m \geq 1$. From the definition of N_f, we have that $N_f \longrightarrow_{\mathfrak{G}} h(T, \ldots, T) \longrightarrow^+_{\mathfrak{G}} h(u_1, \ldots, u_m)$. To complete the proof, we need to prove that if $T \longrightarrow^*_{\mathfrak{G}} u_i$ then $u_i \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$, i.e., u_i is an $\mathfrak{a}$-normal form, for all $1 \leq i \leq m$. We use induction on the structure of u_i.
  - If $u_i = x \in dom(\sigma)$, u_i is an $\mathfrak{a}$-normal form.
  - If u_i is any constant, then u_i is an $\mathfrak{a}$-normal form.
  - If $u_i = g(v_1, v_2)$ with $g \in \mathtt{Abs}$, i.e., from definition of T we have the derivation $T \longrightarrow_{\mathfrak{G}} N_g \longrightarrow^+_{\mathfrak{G}} g(v_1, v_2)$. By induction hypothesis if $N_g \longrightarrow^+_{\mathfrak{G}} g(v_1, v_2)$ then $g(v_1, v_2) \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma)) - \{\varepsilon_g\} \subseteq \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$.

– If $u_i = h(v_1, \ldots, v_m)$, where $h \in \mathcal{F} - \mathtt{Abs}$ with $m \geq 1$. By definition of the rules of T we have the derivation $T \longrightarrow_{\mathfrak{G}} N_g \longrightarrow_{\mathfrak{G}} h(T, \ldots, T) \longrightarrow^*_{\mathfrak{G}} h(v_1, \ldots, v_m)$. By induction hypothesis, each $v_j \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$, for $1 \leq j \leq m$, which implies that $h(v_1, \ldots, v_m) \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$.

Hence, for all $1 \leq i \leq m$, it holds that $u_i \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$, then the term $h(u_1, \ldots, u_m)) \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma)) - \{\varepsilon_f\}$.

Completeness

We need to prove that if $s \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma)) - \{\varepsilon_f\}$ then there exists a derivation of the form $N_f \longrightarrow^*_{\mathfrak{G}} s$. We use induction over the structure of s.

– If $s = x \in dom(\sigma)$ then we have that $N_f \longrightarrow_{\mathfrak{G}} x$.

– If $s = c \neq \varepsilon_f$, we have that $N_f \longrightarrow_{\mathfrak{G}} c$.

– If $s = g(s_1, s_2)$ for $g \in \mathtt{Abs}$, by induction hypothesis there exist derivations $N_g \longrightarrow^*_{\mathfrak{G}} s_1$ and $N_g \longrightarrow^*_{\mathfrak{G}} s_2$ then
$$N_f \longrightarrow_{\mathfrak{G}} g(N_g, N_g) \longrightarrow^*_{\mathfrak{G}} g(s_1, N_g) \longrightarrow^*_{\mathfrak{G}} g(s_1, s_2).$$

– If $s = h(u_1, \ldots, u_m)$, where $h \in \mathcal{F} - (\mathtt{Abs} \cup \{\varepsilon_f\})$, and $m \geq 1$. Since h is not an absorptive symbol then u_i can be any $\mathfrak{a}$-normal form, i.e, $u_i \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$. By the definition of N_f we have that
$$N_f \longrightarrow_{\mathfrak{G}} h(T, \ldots, T),$$
it remains to prove that T generates each u_i for $1 \leq i \leq m$, it means that T is generating $\mathfrak{a}$-normal forms restricted to the variables in the domain of σ. Since $T \longrightarrow_{\mathfrak{G}} N_g$ for all $N_g \in \mathfrak{N}$, then by induction hypothesis for each $1 \leq i \leq m$, we have that $T \longrightarrow N_g \longrightarrow^*_{\mathfrak{G}} u_i$ for the respective g, and it holds that $N_f \longrightarrow_{\mathfrak{G}} h(T, \ldots, T) \longrightarrow^*_{\mathfrak{G}} h(u_1, \ldots, u_m)$.

This implies that there exists $N_f \longrightarrow^*_{\mathfrak{G}} s$ for all $s \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$.

Therefore, from soundness and completeness it holds that $\mathcal{L}(\mathfrak{G}(t, \sigma), T) = \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$ and $\mathcal{L}(\mathfrak{G}(t, \sigma), N_f) = \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma)) - \{\varepsilon_f\}$. □

Lemma 2.2.15. *Let $t \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F})$, and σ be a substitution. If $\varepsilon_f \in ran(\sigma)$, it holds that $\mathcal{L}(\mathfrak{G}(t, \sigma), C_f) = \nearrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$.*

PROOF: Since $\varepsilon_f \in ran(\sigma) \cap subt(t)$, $\mathfrak{G}(t, \sigma) = \langle S_0; \{S_0, T\} \cup \mathfrak{S}^t_\sigma \cup \mathfrak{C}^t_\sigma \cup \mathfrak{N}; \mathcal{F} \cup dom(\sigma); R\rangle$, where R includes the rules:

$$
\begin{array}{lll}
S^{\varepsilon_f} & \longrightarrow \varepsilon_f \mid C_f & \\
 & \text{for all } x \in dom(\sigma) \text{ such that } x\sigma = \varepsilon_f & \\
N_f & \longrightarrow g(N_g, N_g) \mid h(T, \ldots, T) \mid x & \text{for all } N_f \in \mathfrak{N}, x \in dom(\sigma), g \in \mathtt{Abs}, \\
 & & \text{and } h \in \mathcal{F} - (\mathtt{Abs} \cup \{\varepsilon_f\}) \\
T & \longrightarrow N_g & \text{for all } N_g \in \mathfrak{N}.
\end{array}
$$

We want to prove that $s \in \uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$ iff $C_f \longrightarrow^*_{\mathfrak{G}(\varepsilon_f,\sigma)} s$. We prove this using induction as follows:

Soundness

We need to prove that for any σ, all terminal trees s generated by the non-terminal C_f (i.e. $C_f \longrightarrow^*_{\mathfrak{G}} s$), belong to $\uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$. For instance, from C_f, only $x \in dom(\sigma)$ with $x\sigma = \varepsilon_f$ and terminal trees generated by $f(C_f, N_f)$ or $f(N_f, C_f)$ can be generated. We use induction on the structure of s.

1. If $s = x$, by the definition of the rule we have that $x\sigma = \varepsilon_f$, then $x \in \uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$.
2. If $s = f(s_1, s_2)$, we have that either $C_f \longrightarrow_{\mathfrak{G}} f(C_f, N_f) \longrightarrow^+_{\mathfrak{G}} f(s_1, s_2)$ or $C_f \longrightarrow_{\mathfrak{G}} f(N_f, C_f) \longrightarrow^+_{\mathfrak{G}} f(s_1, s_2)$, the analysis of these cases is analogous. We prove the first case: $C_f \longrightarrow_{\mathfrak{G}} f(C_f, N_f) \longrightarrow^+_{\mathfrak{G}} f(s_1, s_2)$. By induction hypothesis we got that $s_1 \in \uparrow(\varepsilon_f, \sigma)$. Then s_1 is in $\mathfrak{a}$-normal form and $s_1\sigma \approx_{\mathfrak{a}} \varepsilon_f$, this implies that $f(s_1, s_2)\sigma \approx_{\mathfrak{a}} \varepsilon_f$ and by Lemma 2.2.14 it holds that $s_2 \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$. Thus, we have that $f(s_1, s_2) \in \uparrow(\varepsilon_f, \sigma)$.

Completeness

We need to prove that for any term s in the set $\uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$ there exists a derivation such that $C_f \longrightarrow^*_{\mathfrak{G}} s$. We use induction on the structure of s.

- If $s = x \in dom(\sigma)$ with $x\sigma = \varepsilon_f$, we have that $C_f \longrightarrow_{\mathfrak{G}} x$.
- If $s = f(s_1, s_2)$, since $s \in \uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$ then $f(s_1\sigma, s_2\sigma) \approx_{\mathfrak{a}} \varepsilon_f$, this implies that either $s_1\sigma \approx_{\mathfrak{a}} \varepsilon_f$ or $s_2\sigma \approx_{\mathfrak{a}} \varepsilon_f$, where s_1 and s_2 are different to ε_f, since s is an $\mathfrak{a}$-normal form. We analyze the following subcases:
  - If $s_1 \in \uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$, then by induction hypothesis there exists $C_f \longrightarrow^*_{\mathfrak{G}} s_1$, implying that
    $$C_f \longrightarrow_{\mathfrak{G}} f(C_f, N_f) \longrightarrow^*_{\mathfrak{G}} f(s_1, N_f),$$
    and since $s_2 \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma)) - \{\varepsilon_f\}$ by Lemma 2.2.14 there exists $N_f \longrightarrow^*_{\mathfrak{G}} s_2$. Therefore, we have the following derivation:
    $$C_f \longrightarrow_{\mathfrak{G}} f(C_f, N_f) \longrightarrow^*_{\mathfrak{G}} f(s_1, N_f) \longrightarrow^*_{\mathfrak{G}} f(s_1, s_2).$$
  - If $s_2 \in \uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$, this case is analogous to the case above.

– If $s_1, s_2 \in \uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$, then by induction hypothesis there exist $C_f \longrightarrow^*_{\mathfrak{G}} s_1$ and $C_f \longrightarrow^*_{\mathfrak{G}} s_2$, implying that either $C_f \longrightarrow_{\mathfrak{G}} f(C_f, N_f) \longrightarrow^*_{\mathfrak{G}} f(s_1, N_f)$ or $C_f \longrightarrow_{\mathfrak{G}} f(N_f, C_f) \longrightarrow^*_{\mathfrak{G}} f(N_f, s_2)$. Then we need to prove that N_f produces all $s' \in \uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$. We use induction on the structure of s':

  * If $s' = x \in dom(\sigma)$ with $x\sigma = \varepsilon_f$, we have that $N_f \longrightarrow_{\mathfrak{G}} x$.
  * If $s' = f(s_3, s_4)$ such that $f(s_3\sigma, s_4\sigma) \approx_{\mathfrak{a}} \varepsilon_f$, i.e., where $s_3 \in \uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$ or $s_4 \in \uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$ or both. In any case we have that $N_f \longrightarrow_{\mathfrak{G}} f(N_f, N_f)$. By the induction hypothesis we have that $N_f \longrightarrow^*_{\mathfrak{G}} s_3$ and $N_f \longrightarrow^*_{\mathfrak{G}} s_4$ and by Lemma 2.2.14 we have that $s_3, s_4 \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F}, dom(\sigma))$. Hence, for any case it holds that N_f produces $f(s_3, s_4)$.

  In any case, for all $s \in \uparrow(t, \sigma)$ there exists a derivation from C_f to s using the rules of the grammar $\mathfrak{G}(\varepsilon_f, \sigma)$.

Therefore, from the soundness and completeness we have that for all $s \in \uparrow(\varepsilon_f, \sigma) - \{\varepsilon_f\}$ iff $S_0 \longrightarrow_{\mathfrak{G}} S^{\varepsilon_f} \longrightarrow_{\mathfrak{G}} C_f \longrightarrow^*_{\mathfrak{G}} s$, implying that $\uparrow(\varepsilon_f, \sigma) = \mathcal{L}(\mathfrak{G}(\varepsilon_f, \sigma))$. □

Lemma 2.2.16 (Characterization of $\uparrow(t, \sigma)$)**.** *Let $t \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F})$ such that size$(t) \geq 2$, σ be a substitution, and $max_{\sqsubseteq}(t, \sigma) = \{p_1, \ldots, p_k\}$. Then it holds that:*

$$\uparrow(t, \sigma) = \{t[r_1]_{p_1} \cdots [r_k]_{p_k} \mid r_i \in \uparrow(t|_{p_i}, \sigma), 1 \leq i \leq k\} \cup \{x \mid x\sigma = t\}$$

Proof: Assume that $s \in \{t[r_1]_{p_1} \cdots [r_k]_{p_k} \mid r_i \in \uparrow(t|_{p_i}, \sigma), 1 \leq i \leq k\} \cup \{x \mid x\sigma = t\}$ and we prove that $s \in \uparrow(t, \sigma)$. If $s = x$ by the definition of the set, it holds that $s \in \uparrow(t, \sigma)$. Otherwise, we have that

$$s\sigma = t[r_1]_{p_1} \cdots [r_k]_{p_k}\sigma = t[r_1\sigma]_{p_1} \cdots [r_k\sigma]_{p_k} \approx_{\mathfrak{a}} t[t|_{p_1}]_{p_1} \cdots [t|_{p_k}]_{p_k} = t.$$

Conversely, assume that $s \in \uparrow(t, \sigma)$ and we prove that this s belongs to the following set $\{t[r_1]_{p_1} \cdots [r_k]_{p_k} \mid r_i \in \uparrow(t|_{p_i}, \sigma), 1 \leq i \leq k\} \cup \{x \mid x\sigma = t\}$. We proceed by induction on the structure of s.

Base Case

If $s = x$, then $s\sigma = t$. Hence $s \in \{x \mid x\sigma = t\}$.

Inductive step.

Suppose $s = h(s_1, \ldots, s_m)$ with $m \geq 1$. Then $t = h(t_1, \ldots, t_m)$, where $s_i\sigma \approx_{\mathfrak{a}} t_i$, i.e. $s_i \in \uparrow(t|_i, \sigma)$ for all $1 \leq i \leq m$. By the induction hypothesis, we have

$$\begin{aligned}\uparrow(t|_i, \sigma) &= \uparrow(t_i, \sigma) \\ &= \left\{ t_i[r_{i1}]_{p_{i1}} \cdots [r_{ik(i)}]_{p_{ik(i)}} \;\middle|\; r_{ij} \in \uparrow(t_i|_{p_{ij}}, \sigma),\ 1 \leq j \leq k(i) \right\} \cup \{x \mid x\sigma = t_i\},\end{aligned}$$

where $k(i)$ denotes the respective number of maximal positions of t_i, that is, $\max_{\sqsubseteq}(t|_i, \sigma) = \{p_{i1}, \ldots, p_{ik(i)}\}$. Thus, either $s_i = x$ for some $x \in \uparrow(t_i, \sigma)$, i.e. $x\sigma = t_i$, or $s_i =$

$t_i[r_{i1}]_{p_{i1}} \cdots [r_{ik(i)}]_{p_{ik(i)}}$, with $r_{ij} \in \uparrow(t_i|_{p_{ij}}, \sigma)$, for each $1 \leq i \leq m$. Substituting all the s_i's in the term s, we obtain:

$$\begin{aligned} s &= h\Big(t_1[r_{11}]_{p_{11}} \cdots [r_{1k(1)}]_{p_{1k(1)}}, \ldots, t_m[r_{m1}]_{p_{m1}} \cdots [r_{mk(m)}]_{p_{mk(m)}}\Big) \\ &= t[r_{11}]_{1.p_{11}} \cdots [r_{ij}]_{i.p_{ij}} \cdots [r_{mk(m)}]_{m.p_{mk(m)}}. \end{aligned}$$

Observe that $max_{\sqsubseteq}(t, \sigma) = \biguplus_{i=1}^{m}\{i.q \mid q \in max_{\sqsubseteq}(t_i, \sigma)\}$. Consequently, each position of the form $i.p_{ij}$ corresponds uniquely to a position $p_l \in \{p_1, \ldots, p_k\}$. Moreover, since $r_{ij} \in \uparrow(t_i|_{p_{ij}}, \sigma) = \uparrow(t|_{i.p_{ij}}, \sigma) = \uparrow(t|_{p_l}, \sigma)$, we can relabel and write

$$s = t[r_1]_{p_1} \cdots [r_k]_{p_k}, \quad \text{with } r_\ell = r_{ij}.$$

Hence s belongs to the required set. □

Lemma 2.2.17. *Let $t \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F})$ and σ be a substitution. Then, for all $S^r \in \mathfrak{S}^t_\sigma$, it holds $\mathcal{L}(\mathfrak{G}(t, \sigma), S^r) = \uparrow(r, \sigma)$.*

Proof: We use induction on the size of r.

Base Case

If $size(r) = 1$ then r is a constant and we analyze the following cases:

- If $r = c$, a non-absorbing constant. Since $c \in ran(\sigma) \cap subt(t)$, the abstraction set is $\uparrow(c, \sigma) = \{c\} \cup \{x \mid x\sigma = c\}$. Additionally, using the grammar rules we have that $S^c \longrightarrow_{\mathfrak{G}(t,\sigma)} c$ or $S^c \longrightarrow_{\mathfrak{G}(t,\sigma)} x$, for all $x \in dom(\sigma)$ such that $x\sigma = c$. Hence, we have that $\mathcal{L}(\mathfrak{G}(t, \sigma), S^c) = \uparrow(c, \sigma)$.
- If $r = \varepsilon_f$. Since $\varepsilon_f \in ran(\sigma) \cap subt(t)$, the grammar includes the rules $S^{\varepsilon_f} \longrightarrow \varepsilon_f \mid C_f$, and by Lemma 2.2.15 $\mathcal{L}(\mathfrak{G}(t, \sigma), S^{\varepsilon_f}) = \uparrow(\varepsilon_f, \sigma)$.

Induction Step

We assume that the lemma holds for any term with at most size n, for $n \geq 1$. We prove the statement for terms r such that $size(r) = n + 1 \geq 2$. The grammar includes the rules $S^r \longrightarrow l(r, \sigma) \mid x$, where $x\sigma = r$. Then, for a terminal tree in $\mathcal{L}(\mathfrak{G}(t, \sigma), S^r)$, possible derivations from the non-terminal S^r are either of the form $S^r \longrightarrow_{\mathfrak{G}(t,\sigma)} x$ or $S^r \longrightarrow_{\mathfrak{G}(t,\sigma)} l(r, \sigma) \longrightarrow^*_{\mathfrak{G}(t,\sigma)} s$. Suppose that $max_{\sqsubseteq}(r, \sigma) = \{p_1, \ldots, p_k\}$. Then, we have that $l(r, \sigma) = r[S^{r|_{p_1}}]_{p_1} \cdots [S^{r|_{p_k}}]_{p_k}$. Since $size(r|_{p_i}) \leq n$, by the induction hypothesis, for all $S^{r|_{p_i}}$, $\mathcal{L}(\mathfrak{G}(t, \sigma), S^{r|_{p_i}}) = \uparrow(r|_{p_i}, \sigma)$, for all $1 \leq i \leq k$. Therefore, by Lemma 2.2.16, $\mathcal{L}(\mathfrak{G}(t, \sigma), S^r) = \uparrow(r, \sigma)$. □

Theorem 2.2.18 (Soundness and Completeness of $\mathfrak{G}(t, \sigma)$)**.** *Let $t \in \mathcal{T}_{\mathfrak{a}}(\mathcal{F})$ and σ be any substitution. Then, $\uparrow(t, \sigma) = \mathcal{L}(\mathfrak{G}(t, \sigma))$.*

PROOF: This result is an immediately consequence of Lemmas 2.2.16 and 2.2.17. Indeed, if $t \in ran(\sigma)$ we have that $S_0 \longrightarrow S^t$ and by Lemma 2.2.17 $\mathcal{L}(\mathfrak{G}(t,\sigma)) = \uparrow(t,\sigma)$. Otherwise, we have that $S_0 \longrightarrow l(t,\sigma)$, the non-terminals of $l(t,\sigma)$ are $S^r \in \mathfrak{S}^t_\sigma$, by Lemmas 2.2.17 and 2.2.16 it follows that $\mathcal{L}(\mathfrak{G}(t,\sigma)) = \uparrow(t,\sigma)$. □

The existence of a regular grammar tree that characterizes the abstraction set ensures that these generalizations can be compactly represented and efficiently manipulated. This grammar not only guarantees finiteness in the representation of potentially infinite sets, but also facilitates systematic exploration and reasoning over abstractions within the configuration.

Let us consider a particular configuration $\mathcal{C}$. Observe that all AUTs occurring in the delayed set of $\mathcal{C}$ are of the form $\star \triangleq_x t$, $t \triangleq_x \star$ or $\varepsilon_f \triangleq_x \varepsilon_f$ where t is ground $\star$ is a constant and ε_f is an absorbing constant, indicating that the particular term occurring in the AUT at this position is irrelevant. We produce more specific generalizations by composing *abstraction substitutions* with the computed substitution of $\mathcal{C}$. Each *abstraction substitution* associates any label in the delay set of $\mathcal{C}$ with a generalization between t and an interpretation of $\star$ as a particular term or a collapsible term when $\varepsilon_f \triangleq_x \varepsilon_f$. As $\star$ can be any term, we select generalizations of t restricted to the subterms in the solved problems in $\mathcal{C}$ as in the definition 2.2.19 below.

Definition 2.2.19 (Abstraction Substitutions)**.** Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a configuration. A substitution τ is called an *abstraction substitution* of $\mathcal{C}$ if $dom(\tau) = labels(D)$, and for each $y \in dom(\tau)$ we have $y\tau \in \uparrow_y(D,S)$, where

$$\uparrow_y(D,S) := \begin{cases} \uparrow(s,\sigma^l_S) & \text{if } s \triangleq_y \star \in D, \\ \uparrow(t,\sigma^r_S) & \text{if } \star \triangleq_y t \in D, \\ \uparrow(\varepsilon_f,\sigma^l_S) \cap \uparrow(\varepsilon_f,\sigma^r_S) & \text{if } \varepsilon_f \triangleq_y \varepsilon_f \in D. \end{cases}$$

The set of all possible abstraction substitutions of $\mathcal{C}$ is denoted by $\Psi(D,S)$. If $D = \emptyset$, we define $\Psi(D,S) = \{id\}$.

For each final configuration $\mathcal{C} = \langle A; S; D; \theta\rangle$ and $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$, we can build the respective set of abstraction substitution using the abstraction sets for each $x \in labels(D_f)$. For all $\tau \in \Psi(D_f,S_f)$, the term $label(\mathtt{a})\theta_f\tau$ is called *computed generalization* of the terms $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$ generated by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ and Ψ.

Using the abstraction set and substitutions, we illustrate how to build some of the computed generalizations for the terms in Examples 2.1.16 and 2.1.17.

Example 2.2.20. From Example 2.1.16, consider the terms $g(\varepsilon_f, a)$ and $g(f(h(\varepsilon_f), a), \varepsilon_f)$ and the final configuration $\langle \emptyset; \{\varepsilon_f \triangleq_{y_4} a, a \triangleq_{w_2} \varepsilon_f\}; \{\star \triangleq_{y_3} h(\varepsilon_f)\}; \theta_2\rangle$, where the com-

puted substitution $\theta_2 = \{x \mapsto g(f(y_3, y_4), w_2), w_1 \mapsto f(y_3, y_4)\}$. To compute a more specific generalization than $x\theta_2 = g(f(y_3, y_4), w_2)$ we build an abstraction substitution from the sets $D = \{\star \triangleq_{y_3} h(\varepsilon_f)\}$ and $S = \{\varepsilon_f \triangleq_{y_4} a, a \triangleq_{w_2} \varepsilon_f\}$ of the final configuration. For $y_3 \in \mathit{labels}(D)$: $\uparrow_{y_3}(D, S) = \uparrow(h(\varepsilon_f), \sigma_S^l)$, where $\sigma_S^l = \{y_4 \mapsto a, w_2 \mapsto \varepsilon_f\}$ which is the infinite set $\mathcal{L}(\mathfrak{G}(h(\varepsilon_f), \sigma_S^l))$, we list some of the elements below.

$$\uparrow(h(\varepsilon_f), \{y_4 \mapsto a, w_2 \mapsto \varepsilon_f\}) = \{h(\varepsilon_f), h(w_2), h(f(w_2, a)), h(f(f(w_2, b), y_4)), \cdots\}$$

Notice that $h(f(w_2, a)) \in \mathcal{L}(\mathfrak{G}(h(\varepsilon_f), \sigma_S^l))$, since from the rules presented in Example 2.2.12, the derivation $S_0 \longrightarrow_{\mathfrak{G}} h(S^{\varepsilon_f}) \longrightarrow_{\mathfrak{G}} h(C_f) \longrightarrow_{\mathfrak{G}} h(f(C_f, N_f)) \longrightarrow^2_{\mathfrak{G}} h(f(w_2, a))$ can be reached. The set of abstraction substitutions for this final configuration is the infinite set $\Psi(D, S)$ including the following substitutions:

$$\{\{y_3 \mapsto h(\varepsilon_f)\}, \{y_3 \mapsto h(w_2)\}, \{y_3 \mapsto h(f(w_2, a))\}, \{y_2 \mapsto h(f(f(w_2, b), y_4))\}\}$$

Applying one of these substitutions, e.g., $\{y_3 \mapsto h(w_2)\}$ over $x\theta_2$ we obtain the term $g(f(h(w_2), y_4), w_2)$, which is an `lgg` of the initial terms. Notice that we can apply an infinite number of abstraction substitutions to $x\theta_2$, implying that we have an infinite set of computed solutions. Indeed, this constitutes a set of incomparable generalizations, since all solutions are generated from the same term $g(f(y_3, y_4), w_2)$. The variables in the range of each generalization are y_4 and w_2, which cannot be further instantiated to produce a more specific generalization. ◇

Example 2.2.21. Continuing with Example 2.1.17, consider the terms $g(\varepsilon_f, \varepsilon_f, a)$ and $g(\varepsilon_f, b, \varepsilon_f)$ and the final configuration $\langle \emptyset; S; D; \theta_2 \rangle$, where $S = \{\varepsilon_f \triangleq_{w_2} b, a \triangleq_{w_3} \varepsilon_f\}$, $D = \{\varepsilon_f \triangleq_{w_1} \varepsilon_f\}$, and $\theta_2 = \{x \mapsto g(w_1, w_2, w_3)\}$. As D contains only one AUT, then for the variable w_1, $\uparrow_{w_1}(D, S) = \uparrow(\varepsilon_f, \sigma_S^l) \cap \uparrow(\varepsilon_f, \sigma_S^r)$, where $\sigma_S^l = \{w_2 \mapsto \varepsilon_f, w_3 \mapsto a\}$ and $\sigma_S^r = \{w_2 \mapsto b, w_3 \mapsto \varepsilon_f\}$, we present some of the elements of each abstraction set below.

$$\uparrow(\varepsilon_f, \{w_2 \mapsto \varepsilon_f, w_3 \mapsto a\}) = \{\varepsilon_f, w_2, f(w_2, w_2), f(w_2, w_3), f(w_3, w_2), f(f(w_3, a), w_2), \ldots\}$$
$$\uparrow(\varepsilon_f, \{w_2 \mapsto b, w_3 \mapsto \varepsilon_f\}) = \{\varepsilon_f, w_3, f(w_3, w_3), f(w_3, w_2), f(w_2, w_3), f(f(w_3, a), w_2), \ldots\}$$

The intersection of these sets is the set of collapsible terms at the variables w_2 and w_3 simultaneously.

$$\uparrow_{w_1}(D, S) = \{\varepsilon_f, f(w_2, w_3), f(w_3, w_2), f(f(w_3, a), w_2), f(f(h(w_2), w_3), w_2), \ldots\}$$

Observe that the term $f(f(h(w_2), w_3), w_2) \in \uparrow(\varepsilon_f, \sigma_S^l) \cap \uparrow(\varepsilon_f, \sigma_S^r) = \uparrow_{w_1}(D, S)$, since we have that $f(f(h(w_2), w_3), w_2)\sigma_S^l \approx_{\mathfrak{a}} \varepsilon_f$, and $f(f(h(w_2), w_3), w_2)\sigma_S^r \approx_{\mathfrak{a}} \varepsilon_f$. Then $\Psi(D, S)$

is infinite, it contains, e.g., the substitutions $\{w_1 \mapsto f(f(w_3, a), w_2)\}$, $\{w_1 \mapsto f(w_2, w_3)\}$, and $\{w_1 \mapsto f(f(h(w_2), w_3), w_2)\}$. This leads to infinitely many generalizations, e.g., $g(f(f(w_3, a), w_2), w_2, w_3)$, $g(f(w_2, w_3), w_2, w_3)$, and $g(f(f(h(w_2), w_3), w_2), w_2, w_3)$ of the initial terms. Notice that the generalizations $g(\varepsilon_f, w_2, w_3)$ and $g(f(w_3, w_2), w_2, w_3)$ are not $\mathfrak{a}$-equivalent $\mathtt{lgg}$'s of $g(\varepsilon_f, \varepsilon_f, a)$ and $g(\varepsilon_f, b, \varepsilon_f)$, and that $f(w_3, w_2)$ is a generalization (different from ε_f) of $\varepsilon_f \triangleq \varepsilon_f$ that is needed for completeness. This observation underscores the necessity of the (ExpB) rule. $\diamond$

The previous examples reveal that, under the absorptive setting, the computation of generalizations may give rise to an infinite number of incomparable solutions. As shown in Example 2.2.20, an infinite set of abstraction substitutions can be applied to terms such as $x\theta_2$, producing an infinite collection of computed generalizations. Each of these generalizations is incomparable with the others, since they share the same structural pattern but differ in the specific instantiations introduced at identical positions within that pattern by the abstraction substitutions.

These observations indicate that the absorptive anti-unification problem is at least infinitary. In particular, the generation process gives rise to infinitely many incomparable generalizations, although their production can be finitely represented through the grammar of abstraction substitutions. This property will be revisited in the context of soundness and completeness.

2.3 Soundness of $\mathcal{A}_{\mathfrak{a}\text{-}\mathbf{AUnif}}$ and Abstraction Substitutions

This section treats the soundness property of the $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ algorithm, and the soundness of the computed generalizations, i.e., the soundness of generalizations using the set of abstraction substitutions Ψ. The termination, Theorem 2.1.14, and the preservation configuration, Lemma 2.1.11, are essential to the soundness proof, as these properties enforce consistency concerning the use of the labels.

The following lemma presents some preservation properties of the configurations that will be used for soundness.

Lemma 2.3.1. *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ and $\mathcal{C}' = \langle A'; S'; D'; \theta'\rangle$ be configurations and $\mathcal{C} \Longrightarrow^* \mathcal{C}'$ be a derivation. Then the following properties hold:*

1. *$labels(A \cup S \cup D) \cup dom(\theta) \subseteq labels(A' \cup S' \cup D') \cup dom(\theta')$.*
2. *$D \subseteq D'$*
3. *There exists η such that $\theta' = \theta\eta$.*

4. $\theta\theta' = \theta'$.

PROOF: We proceed by induction over the derivation length.

Base case
If the derivation has length 0, it means that $\langle A; S; D; \theta\rangle$ is a final configuration. Observe that (1) and (2) holds. Moreover, taking η as the identity substitution (3) holds, and from Remark 2.1 θ is idempotent and (4) holds.

Induction Step
Consider the derivation $\mathcal{C} \Longrightarrow \mathcal{C}'' \Longrightarrow^* \mathcal{C}'$ where $\mathcal{C}'' = \langle A''; S''; D''; \theta''\rangle$. By induction hypothesis, the conditions $1 - 4$ hold for $\mathcal{C}''$.

1. $\mathit{labels}(A'' \cup S'' \cup D'') \cup \mathit{dom}(\theta'') \subseteq \mathit{labels}(A' \cup S' \cup D') \cup \mathit{dom}(\theta')$, we need to prove that $\mathit{labels}(A \cup S \cup D) \cup \mathit{dom}(\theta) \subseteq \mathit{labels}(A'' \cup S'' \cup D'') \cup \mathit{dom}(\theta'')$. The proof of the last statement is obtained by analyzing the rules by case:

   - The rules (Dec), (ExpLA1), (ExpLA2), (ExpRA1), and (ExpRA2) move one label of A to $\mathit{dom}(\theta')$ and add new labels in A'. The other labels and variables in the domain remain equal, implying the property needed.
   - The rules (ExpB) move a label of A to D'. The other labels and the domain remain equal, and the property holds.
   - The rule (Sol) moves a label of A to S'', then $\mathit{labels}(A \cup S \cup D) \cup \mathit{dom}(\theta)$ is equal to $\mathit{labels}(A'' \cup S'' \cup D'') \cup \mathit{dom}(\theta'')$.
   - The rule (Mer) moves one label of S to $\mathit{dom}(\theta'')$ and the other labels and variables in the domain remain equal, implying the property needed.

   Hence, it holds that $\mathit{labels}(A \cup S \cup D) \cup \mathit{dom}(\theta) \subseteq \mathit{labels}(A' \cup S' \cup D') \cup \mathit{dom}(\theta')$.

2. $D \subseteq D''$, it remains to prove that $D'' \subseteq D'$. We prove this by case analysis of rule application:

   - The rules (Dec), (Sol), (Mer) do not affect the delayed set, then it holds that $D \subseteq D'' = D'$.
   - The rules (ExpLA1), (ExpLA2), (ExpRA1), (ExpRA2), and (ExpB) add a new AUT to the set D', then $D'' \subseteq D'$. Therefore, we have that $D \subseteq D'$.

3. We have that exists η'' such that $\theta' = \theta''\eta''$. Observe by rule analysis that $\theta'' = \theta\eta'$:

   - All the rules with the exception of the (Sol) and (ExpB) rules add a substitution with one binding to θ, i.e., $\theta'' = \theta\eta'$ where η' depends on the rule.

- Rules (Sol) and (ExpB) do not modify the computed substitution θ, then $\theta = \theta\eta'$ where $\eta' = id$.

Implying that $\theta'' = \theta\eta'$. Hence $\theta' = \theta\eta$, where $\eta = \eta'\eta''$.

4. We have that $\theta' = \theta''\theta'$, we need to prove that $\theta'' = \theta\theta''$. Notice by condition (2) and Remark 2.1 it holds that $\theta\theta'' = \theta(\theta\eta) = \theta''$. Hence, $\theta' = (\theta\theta'')\theta' = \theta\theta''$. □

Lemma 2.3.2 (Auxiliary Soundness of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$)**.** *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a configuration, and $\mathcal{C}_f = \langle\emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$. Then $label(\mathtt{a})\theta_f \in \mathcal{G}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$, for all $\mathtt{a} \in A \cup S$.*

PROOF: We proceed by induction over the derivation length.

Base case

If the derivation has length 0, by Lemma 2.1.13 we have that $\langle A; S; D; \theta\rangle$ is a final configuration. Moreover, for all $\mathtt{a} \in S$, $label(\mathtt{a})\theta_f = label(\mathtt{a}) \in \mathcal{G}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$, since all AUTs in S are solved.

Induction Step

Now, consider a derivation having the following form:

$$\langle A; S; D; \theta\rangle \Longrightarrow \langle A'; S'; D'; \theta'\rangle \Longrightarrow^n \langle\emptyset; S_f; D_f; \theta_f\rangle \tag{2.1}$$

For $n \geq 0$. We assume for the induction hypothesis that for derivations of the form

$$\langle A'; S'; D'; \theta'\rangle \Longrightarrow^n \langle\emptyset; S_f; D_f; \theta_f\rangle,$$

the lemma holds and show that the lemma holds for derivations of the form presented in Derivation 2.1. We continue the proof considering the various options for the transition from $\langle A; S; D; \theta\rangle$ to $\langle A'; S'; D'; \theta'\rangle$.

1. **(Dec)**. Assume that the derivation is of the form:

$$\langle\{f(s_1, \ldots, s_m) \triangleq_y f(t_1, \ldots, t_m)\} \cup A_1; S; D; \theta\rangle \overset{Dec}{\Longrightarrow}$$
$$\langle\{s_1 \triangleq_{x_1} t_1, \ldots, s_m \triangleq_{x_m} t_m\} \cup A_1; S; D; \theta'\rangle \Longrightarrow^n \langle\emptyset; S_f; D_f; \theta_f\rangle$$

where $\theta' = \theta\{y \mapsto f(x_1, \ldots, x_m)\}$. By the induction hypothesis, we know that for all AUT $\mathtt{a} \in A_1 \cup S$, it holds that $label(\mathtt{a})\theta_f \in \mathcal{G}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$ and for all $1 \leq i \leq m$, $x_i\theta_f \in \mathcal{G}_{\mathfrak{a}}(s_i, t_i)$. From Lemma 2.3.1, $y\theta_f = y(\theta\{y \mapsto f(x_1, \ldots, x_m)\})\theta_f$ and from definition of configuration $y\theta = y$. Then $y\theta_f = f(x_1\theta_f, \ldots, x_m\theta_f)$, this implies that $y\theta_f \in \mathcal{G}_{\mathfrak{a}}(f(s_1, \ldots, s_m), f(t_1, \ldots, t_m))$.

2. **(Sol)**. Assume that the derivation is of the form:

$$\langle\{s \triangleq_y t\} \cup A_1; S; D; \theta\rangle \overset{Sol}{\Longrightarrow} \langle A_1; S'; D; \theta\rangle \Longrightarrow^n \langle\emptyset; S_f; D_f; \theta_f\rangle,$$

where $S' = \{s \triangleq_y t\} \cup S$. By the induction hypothesis, for all $\mathtt{a} \in A_1 \cup S'$ we have that $label(\mathtt{a})\theta_f \in \mathcal{G}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$. In particular, it holds that $y\theta_f \in \mathcal{G}_{\mathfrak{a}}(s, t)$.

3. **(Mer)** Assume that the derivation is of the form:
$$\langle \emptyset; \{s \triangleq_y t, s \triangleq_z t\} \cup S_1; D; \theta\rangle \overset{Mer}{\Longrightarrow}$$
$$\langle \emptyset; \{s \triangleq_z t\} \cup S_1; D; \theta\{y \mapsto z\}\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle.$$
By the induction hypothesis, $label(\mathtt{a})\theta_f \in \mathcal{G}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$ for all $\mathtt{a} \in \{s \triangleq_z t\} \cup S_1$, in particular $z \in \mathcal{G}_{\mathfrak{a}}(s,t)$. From Lemma 2.3.1 $y\theta_f = z$, implying that $y\theta_f \in \mathcal{G}_{\mathfrak{a}}(s,t)$.

4. **(ExpB)**. Assume that the derivation is of the form:
$$\langle \{\varepsilon_f \triangleq_y \varepsilon_f\} \cup A_1; S; D; \theta\rangle \overset{ExpB}{\Longrightarrow}$$
$$\langle A_1; S; D'; \theta\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle$$
where $D' = \{\varepsilon_f \triangleq_y \varepsilon_f\} \cup D$. From the definition of configuration, we have that $y\theta_f = y$, this implies that $y\theta_f = y \in \mathcal{G}_{\mathfrak{a}}(\varepsilon_f, \varepsilon_f)$.

5. **(ExpLA1)**. Assume that the derivation is of the form:
$$\langle \{\varepsilon_f \triangleq_y f(s,t)\} \cup A_1; S; D; \theta\rangle \overset{ExpLA1}{\Longrightarrow}$$
$$\langle \{\varepsilon_f \triangleq_{x_1} s\} \cup A_1; S; D'; \theta'\rangle\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle$$
where $D' = \{\star \triangleq_{x_2} t\} \cup D$ and $\theta' = \theta\{y \mapsto f(x_1, x_2)\}$. By the induction hypothesis, all the AUTs in S and $\{\varepsilon_f \triangleq_{x_1} s\} \cup A_1$ are generalized by θ_f, thus, $x_1\theta_f \in \mathcal{G}_{\mathfrak{a}}(\varepsilon_f, s)$. Moreover, since $x_2 \in labels(D_f)$ then $x_2\theta_f = x_2 \preceq_{\mathfrak{a}} t$. Notice that $f(x_1\theta_f, x_2\theta_f) = f(x_1\theta_f, x_2) \in \mathcal{G}_{\mathfrak{a}}(f(\varepsilon_f, t), f(s,t))$ and since $f(\varepsilon_f, t) \approx_{\mathfrak{a}} \varepsilon_f$, we get that $y\theta_f = f(x_1\theta_f, x_2\theta_f) \in \mathcal{G}_{\mathfrak{a}}(\varepsilon_f, f(s,t))$.

6. The analysis of the other lateral expansion rules is analogous to the previous one.□

While the soundness of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ theorem covers the construction of generalizations of AUTs present in a given configuration, it does not consider the abstraction set or the construction of more specific generalizations when generalizing over an absorptive theory. The abstraction set allows us to consider generalizations between a given term and an arbitrary term.

Lemma 2.3.3. *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ and $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-}AUNIF}(\mathcal{C})$. Then for all $\star \triangleq_w t \in D_f$ (resp. for all $s \triangleq_w \star \in D_f$) and $\tau \in \Psi(D_f, S_f)$, there exists a term r such that $w\tau \in \mathcal{G}_{\mathfrak{a}}(r,t)$ (resp. $w\tau \in \mathcal{G}_{\mathfrak{a}}(r,s)$).*

PROOF: Let η be a ground substitution with $dom(\eta) = var(w\tau)$. Then $r = w\tau\eta$. □

Intuitively, Lemma 2.3.3 formalizes the following observation: if $s \triangleq_w \star \in D_f^l$ (resp. $\star \triangleq_w t \in D_f^r$), by Definition 2.2.19, for all τ, it holds that $w\tau \in \uparrow_w(D_f, S_f)$ implying that $w\tau \in \uparrow(s, \sigma^l_{S_f})$ (resp. $w\tau \in \uparrow(t, \sigma^r_{S_f})$). From this, we can deduce that $w\tau \preceq_{\mathfrak{a}} s$ (resp. $w\tau \preceq_{\mathfrak{a}} t$). Thus, for every wild AUT in the set D_f^l (resp. D_f^r), the wild card can be interpreted as $w\tau\sigma^l_{S_f}$ (resp. $w\tau\sigma^r_{S_f}$) and $w\tau \in \mathcal{G}_{\mathfrak{a}}(w\tau\sigma^l_{S_f}, s)$ (resp. $w\tau \in \mathcal{G}_{\mathfrak{a}}(w\tau\sigma^r_{S_f}, t)$).

The following lemmas help us prove that every abstraction substitution in Ψ is sound, as well as the soundness of the computed generalizations.

Lemma 2.3.4 (Nested weight Preserving Rules)**.** *Let $\mathcal{C}$ be a configuration and $\mathcal{C} \Longrightarrow \mathcal{C}'$ be a derivation after an application of any non-lateral expansion rule. Then, it holds that $nes(A) = nes(A')$, where A and A' are the active set of the configurations $\mathcal{C}$ and $\mathcal{C}'$ respectively.*

Proof: We analyze each possible rule application.

- Rule (Dec). This implies that A is of the form $\{f(s_1, \ldots, s_m) \triangleq_x f(t_1, \ldots, t_m)\} \cup A_1$ and $A' = \{s_1 \triangleq_{x_1} t_1, \ldots, s_n \triangleq_{x_n} t_n\} \cup A_1$. We analyze the two possible subcases based on the number of lateral-expansible positions in the terms $f(s_1, \ldots, s_m)$ and $f(t_1, \ldots, t_m)$:
  - No lateral-expansible position. If there is no lateral-expansible position between the terms, then $nes(f(s_1, \ldots, s_m) \triangleq_x f(t_1, \ldots, t_m)) = 0$, and each pair of subterms s_i and t_i also has no lateral-expansible position. This implies that $nes(s_i \triangleq_{x_i} t_i) = 0$, for all $1 \leq i \leq m$. Therefore, we have $nes(A) = nes(A')$.
  - At least one lateral-expansible position. By Definition 1.3.17, for every lateral-expansible position p and every $q \sqsubset p$, we have
    $$head(f(s_1, \ldots, s_m)|_q) = head(f(t_1, \ldots, t_m)|_q).$$
    If $p = i.p'$, then p' is an lateral-expansible position of the subterms s_i and t_i. It follows that
    $$nes(f(s_1, \ldots, s_m) \triangleq_x f(t_1, \ldots, t_m)) = \sum_{i=1}^{m} nes(s_i \triangleq_{x_i} t_i).$$
    Hence, we have $nes(A) = nes(A')$.
- Rule (Sol). This implies that $A = \mathtt{a} \cup A_1$ and $A' = A_1$, where $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$ have different head symbols and they are not absorptive related symbols. We have that $nes(\mathtt{a}) = 0$, which implies that $nes(A) = nes(A')$.
- Rule (Mer). This implies that $A = A' = \emptyset$, and then the lemma holds.
- Rule (ExpB). This implies that A is fo the form $\{\varepsilon_f \triangleq_x \varepsilon_f\} \cup A_1$ and $A' = A_1$. We have that $nes(\varepsilon_f \triangleq_x \varepsilon_f)$, it follows that $nes(A) = nes(A')$. □

Lemma 2.3.5 (Expansible-Solvable Position Reachability)**.** *Let $\mathcal{C} = \langle A; S; D; \theta \rangle$ be a configuration, and let p be an expansible-solvable position of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$, for some $\mathtt{a} \in A$. Then, the following holds:*

- *There exists a derivation $\mathcal{C} \Longrightarrow^* \langle \{(lhs(\mathtt{a}))|_p \triangleq_x (rhs(\mathtt{a}))|_p\} \cup A'; S'; D'; \theta' \rangle$.*

- *Moreover, if this derivation is such that p is the first lateral-expansible position reached, then* $nes(\{(lhs(\mathtt{a}))|_p \triangleq_x (rhs(\mathtt{a}))|_p\} \cup A') = nes(A)$.

PROOF: For the first part, consider an $\mathtt{a} \in A$. By Definition 1.3.17, for all expansible-solvable position p of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$ and for all $q \sqsubset p$, we have that $head((lhs(\mathtt{a}))|_q) = head((rhs(\mathtt{a}))|_q)$.
Let n be the number of function symbols between the root and the symbol at position p. Then, due to the non-determinism of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$, it is possible to apply the (Dec) rule exactly $n+1$ times, each time selecting the adequate AUT that contains the current subterm at position p. This yields the derivation:

$$\mathcal{C} \overset{Dec}{\Longrightarrow}^{n+1} \langle \{lhs(\mathtt{a})|_p \triangleq_x rhs(\mathtt{a})|_p\} \cup A'; S'; D'; \theta' \rangle.$$

For the second part, since it is possible to reach any lateral-expansible position p of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$, for every $\mathtt{a} \in A$. We can consider the following derivation

$$\mathcal{C} \Longrightarrow^* \langle \{lhs(\mathtt{a})|_p \triangleq_x rhs(\mathtt{a})|_p\} \cup A'; S'; D'; \theta' \rangle$$

where p is the first lateral-expansible position reached. In this case, all the rules applied before reaching p are non-lateral expansion rules. Thus, by Lemma 2.3.4, the nested weight is preserved. Therefore, we conclude that $nes(A) = nes(\{lhs(\mathtt{a})|_p \triangleq_x rhs(\mathtt{a})|_p\} \cup A')$. □

Lemma 2.3.6. *Let $\mathcal{C} = \langle A; S; D; \theta \rangle$ be a configuration and $\mathcal{C}_f \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$ of the form $\langle \emptyset; S_f; D_f; \theta_f \rangle$. Then, for all $\mathtt{a} \in A$ and for all lateral-expansible position p of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$ regarding symbol g, there exists a unique $x \in var((label(\mathtt{a})\theta_f)|_p) \cap labels(S_f)$, such that:*

- *$(label(\mathtt{a})\theta_f)|_p$ is x-collapsible,*
- *$\varepsilon_g \triangleq_x u \in S_f$ and $var((label(\mathtt{a})\theta_f)|_p) - \{x\} \subseteq labels(D_f^r)$ with all different variables, if $(lhs(\mathtt{a}))|_p = \varepsilon_g$, or*
- *$u \triangleq_x \varepsilon_g \in S_f$ and $var((label(\mathtt{a})\theta_f)|_p) - \{x\} \subseteq labels(D_f^l))$ with all different variables, if $(rhs(\mathtt{a}))|_p = \varepsilon_g$.*

PROOF: We use induction on $nes(A)$.

Base Case
If $nes(A) = 0$, then the lemma holds vacuously, as there does not exist an lateral-expansible position of $lhs(a)$ and $rhs(a)$ for all $\mathtt{a} \in A$.
Induction Step
We assume that the lemma holds for configurations with $nes(A) \leq n$ and we prove it for $n+1$. By Lemma 2.3.5, we can reach the first lateral-expansible position p for some AUT $\mathtt{a} \in A$. We consider the following cases:

- If we have that $(lhs(\mathtt{a}))|_p = \varepsilon_g$ and $(rhs(\mathtt{a}))|_p = g(s,t)$. We can apply either the rule (ExpLA1) or (ExpLA2). Suppose we have the following derivation (the case (ExpLA2) is analogous):

$$\mathcal{C} \Longrightarrow^* \langle \{\varepsilon_g \triangleq_y g(s,t)\} \cup A'; S'; D'; \theta' \rangle \overset{ExpLA1}{\Longrightarrow}$$
$$\langle \{\varepsilon_g \triangleq_{y_1} s\} \cup A'; S'; \{\star \triangleq_{y_2} t\} \cup D'; \theta'\{y \mapsto g(y_1, y_2)\} \rangle \Longrightarrow^* \mathcal{C}_f,$$

  By Lemma 2.3.5, we get that $nes(A) = nes(\{\varepsilon_g \triangleq_y g(s,t)\} \cup A')$. Also, since $nes_g(g(s,t)) > nes_g(s)$, we can use the induction hypothesis. Consequently, by the induction hypothesis, for all AUT $\mathtt{a}'$ of $\{\varepsilon_g \triangleq_{y_1} s\} \cup A'$ and all p' lateral-expansible position of $\mathtt{a}'$ the lemma holds. We analyze the subcases below.

  - If $head(s) \neq g$ then the rule (Sol) can be applied over $\varepsilon_g \triangleq_{y_1} s$, and then we have that $(label(\mathtt{a})\theta_f)|_p = y\{y \mapsto g(y_1\theta_f, y_2\theta_f)\} = g(x, y_2)$, where x is the label of some AUT of the form $\varepsilon_g \triangleq_x u \in S_f$, since S_f of C_f is merged. Also, it holds that $var((label(\mathtt{a})\theta_f)|_p) - \{x\} = var(g(x, y_2)) - \{x\} \subseteq labels(D_f^r)$.
  - If $head(s) = g$, by induction hypothesis, for $\varepsilon_f \triangleq_{y_1} s$ and $p' = \epsilon$, there exists a unique $x \in var(y_1\theta_f) \cap labels(S_f)$ holding the properties of the lemma. Moreover, as the label associated with position p in $label(\mathtt{a})$ is y, follows that
  $$(label(\mathtt{a})\theta_f)|_p = y\{y \mapsto f(y_1, y_2)\}\theta_f = f(y_1\theta_f, y_2\theta_f) = f(y_1\theta_f, y_2).$$
  This implies that the variable $x \in var((label(\mathtt{a})\theta_f)|_p) \cap labels(S_f)$ is unique, and the term $(label(\mathtt{a})\theta_f)|_p$ is x-collapsible, with $\varepsilon_f \triangleq_x u \in S_f$. Additionally, by Lemma 2.3.1, we have $y_2 \in labels(D_f)$, which implies $y_2 \notin var(y_1\theta_f)$. Hence, it follows that the set
  $$var((label(\mathtt{a})\theta_f)|_p) - \{x\} = (var(y_1\theta_f) \cup \{y_2\}) - \{x\} \subseteq labels(D_f^r),$$
  with all variables different.

- If we have that $(lhs(\mathtt{a}))|_p = g(s,t)$ and $(rhs(\mathtt{a}))|_p = \varepsilon_g$. We can apply either the rule (ExpRA1) or (ExpRA2). In this case, we follow the same reasoning as previous case, and it is possible to conclude that there exists a unique variable $x \in var((label(\mathtt{a})\theta_f)|_p) \cap labels(S_f)$, such that the term $(label(\mathtt{a})\theta_f)|_p$ is x-collapsible, with $u \triangleq_x \varepsilon_f \in labels(S_f)$, and it holds that $var((label(\mathtt{a})\theta_f)|_p) - \{x\} \subseteq labels(D_f^l)$, with all variables different. □

Lemma 2.3.7. *Let $\mathcal{C} = \langle A; S; D; \theta \rangle$ be a configuration, and $\mathcal{C}_f \in \mathcal{A}_{\mathfrak{a}\text{-}AU_{NIF}}(\mathcal{C})$ of the form $\langle \emptyset; S_f; D_f; \theta_f \rangle$. Then, for all $\mathtt{a} \in A$ and p trivially-decomposable position of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$, we have the following:*

- *$(lhs(\mathtt{a}))|_p$ is a constant symbol and $(lhs(\mathtt{a}))|_p = (rhs(\mathtt{a}))|_p$.*
- *$(label(\mathtt{a})\theta_f)|_p = (lhs(\mathtt{a}))|_p$; or $(label(\mathtt{a})\theta_f)|_p \in labels(D_f^n)$.*

PROOF: Assume that p is a trivially-decomposable position of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$. We first prove that $(lhs(\mathtt{a}))|_p$ is a constant symbol. By Lemma 2.3.5, there exists a derivation in which the AUT $(lhs(\mathtt{a}))|_p \triangleq_x (rhs(\mathtt{a}))|_p$ appears in the active set of the resulting configuration. Suppose, for contradiction, that $head((lhs(\mathtt{a}))|_p) = head((rhs(\mathtt{a}))|_p) = g$, with g is a function symbol of arity $m \geq 1$. This implies that the rule (Dec) is applicable again at position p, producing m new AUTs of the form $(lhs(\mathtt{a}))|_{p.i} \triangleq_{x_i} (rhs(\mathtt{a}))|_{p.i}$, for $1 \leq i \leq m$. Since $m \geq 1$, there exists at least one position $p.i$ such that $p \sqsubset p.i$ and $p.i \in pos(lhs(\mathtt{a})) \cap pos(rhs(\mathtt{a}))$, which contradicts the assumption that p is a expansible-solvable position. Therefore, $(lhs(\mathtt{a}))|_p$ must be a constant symbol. The same reasoning applies to $(rhs(\mathtt{a}))|_p$, and then $(lhs(\mathtt{a}))|_p = (rhs(\mathtt{a}))|_p$.
For the second part, we analyze the following cases:

- If $(lhs(\mathtt{a}))|_p$ is any constant, by Lemma 2.3.5 and applying the rule (Dec), we obtain the following derivation:
$$\mathcal{C} \Longrightarrow^* \langle \{(lhs(\mathtt{a}))|_p \triangleq_y (rhs(\mathtt{a}))|_p\} \cup A'; S'; D'; \theta' \rangle \overset{Dec}{\Longrightarrow} \langle A'; S'; D'; \theta'\{y \mapsto (lhs(\mathtt{a}))|_p\} \rangle \Longrightarrow^* \mathcal{C}_f,$$
Then it holds $(label(\mathtt{a})\theta_f)|_p = y\{y \mapsto (lhs(\mathtt{a}))|_p\} = (lhs(\mathtt{a}))|_p$.

- If $(lhs(\mathtt{a}))|_p$ is an absorbing constant, by Lemma 2.3.5 and applying the rule (ExpB), we obtain the following derivation:
$$\mathcal{C} \Longrightarrow^* \langle \{(lhs(\mathtt{a}))|_p \triangleq_y (rhs(\mathtt{a}))|_p \cup A'; S'; D'; \theta' \rangle \overset{ExpB}{\Longrightarrow} \langle A'; S'; \{(lhs(\mathtt{a}))|_p \triangleq_y (rhs(\mathtt{a}))|_p\} \cup D'; \theta' \rangle \Longrightarrow^* \mathcal{C}_f,$$
Since no rule is applied over the delayed set, it holds that $y\theta_f = y$ and thus $(label(\mathtt{a})\theta)|_p = y \in labels(D^n_f)$. □

Lemma 2.3.8. *Let $\mathcal{C} = \langle A; S; D; \theta \rangle$ be a configuration and $\mathcal{C}_f \in \mathcal{A}_{\mathfrak{a}\text{-}AUNIF}(\mathcal{C})$ of the form $\langle \emptyset; S_f; D_f; \theta_f \rangle$. Then, for all $\mathtt{a} \in A$ and for all solvable position p of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$, there exists a unique $x \in labels(S_f)$ such that $(rhs(\mathtt{a}))|_p \triangleq_x (lhs(\mathtt{a}))|_p \in S_f$ and $(label(\mathtt{a})\theta_f)|_p = x$.*

PROOF: Assume that p is a solvable position of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$, for some $\mathtt{a} \in A$. By Lemma 2.3.5 and applying rule (Sol), we have the following derivation:
$$\mathcal{C} \Longrightarrow^* \langle \{(lhs(\mathtt{a}))|_p \triangleq_y (rhs(\mathtt{a}))|_p\} \cup A'; S'; D'; \theta' \rangle \overset{Sol}{\Longrightarrow} \langle A'; \{(lhs(\mathtt{a}))|_p \triangleq_y (rhs(\mathtt{a}))|_p\} \cup S'; D'; \theta' \rangle \Longrightarrow^* \mathcal{C}_f,$$
By the Lemma 2.3.2, we have $(label(\mathtt{a})\theta_f)|_p = y\theta_f \in \mathcal{G}_{\mathfrak{a}}((lhs(\mathtt{a}))|_p, (rhs(\mathtt{a}))|_p)$. Since S_f is merged, we conclude that there exists a unique AUT $(lhs(\mathtt{a}))|_p \triangleq_x (rhs(\mathtt{a}))|_p \in S_f$ with $(label(\mathtt{a})\theta_f)|_p = x$. □

Theorem 2.3.9 (Soundness of Computed Generalizations)**.** *Let* $\mathcal{C} = \langle A; S; D; \theta \rangle$ *be a configuration, and* $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-}AUNIF}(\mathcal{C})$*. Then for all* $\tau \in \Psi(D_f, S_f)$ *we have that the substitution* $(\theta_f \tau)|_{labels(A \cup S \cup D)} \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C})$*.*

PROOF: The proof follows as a consequence of the Soundness of Extended Computed Generalizations (Theorem 3.2.10) and Lemma 3.2.7, presented in Chapter 3. □

Chapter 3

On Completeness of Absorptive Anti-Unification

This section discusses the difficulties in addressing the completeness of $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}$. We begin by analyzing the approach to completeness proposed in [9], which is not correct. We present the errors in both the statement and the proof. Then, we provide a precise formulation of the completeness result, along with a discussion of the main challenges involved in establishing it. One of the key issues concerns the treatment of abstraction substitutions: certain substitutions that are essential for completeness are not generated by the current procedure. We illustrate this problem through concrete examples, highlighting cases where expected generalizations fail to appear, and propose a potential refinement that incorporates the computation of these additional generalizations.

3.1 Completeness Statement

The classical completeness proof in equational anti-unification consists of finding a more specific computed generalization than a given generalization of any pair of terms s and t. We set the following statement as the standard of completeness that we aim to achieve.

Statement 3.1 (Completeness of $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}$ and Ψ)**.** Let $\mathcal{C} = \langle A; S; D; \theta \rangle$ be a configuration and $\gamma \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C})$. Then, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}(\mathcal{C})$ and $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ such that $(\gamma \preceq_{\mathfrak{a}} \theta_f \tau)|_{labels(A \cup S \cup D)}$.

All the approaches in which we attempted to prove completeness in absorptive theory led to introducing new tools for computing more specific $\mathfrak{a}$-generalizations. For example, a more elaborate definition of configuration, the inclusion of the delayed set into the configurations, and the definition of the abstraction substitutions for capturing $\mathfrak{a}$-generalizations in problems where expansions of ε_f were necessary.

On the other hand, note that the statement of completeness presented in [9] is not satisfactory, since it refers separately to generalizations $r \in \mathcal{G}_{\mathfrak{a}}(s,t)$, for $s \triangleq_x t \in A$. This makes the inductive analysis imprecise in the action of the inference rules applied by the anti-unification algorithm to achieve a conclusion $r \preceq_{\mathfrak{a}} x\theta'\tau$. Furthermore, the proof presented in [9] fails during the analysis of the rule (Dec) in the inductive step. The separate treatment of each subproblem obtained after applying this rule does not allow for proper "alignment" of the computed generalizations.

3.2 Issues to Achieve a Completeness Proof

To analyze this in detail, the approach in [9] relies on induction over the size of the active set of a given configuration $\mathcal{C}$, followed by a case analysis of rule application. In particular, for the rule (Dec), the proof considers an $\mathfrak{a}$-generalization r of the form $r = g(r_1, \ldots, r_n)$ of the AUT $g(s_1, \ldots, s_n) \triangleq_x g(t_1, \ldots, t_n)$ in the active set of the configuration $\mathcal{C}$ for which it is necessary to find a computed generalization r' such that $r \preceq_{\mathfrak{a}} r'$. In this case, the rule (Dec) is applied, obtaining a new configuration including in the active set the AUTs $s_i \triangleq_{y_i} t_i$, for $1 \leq i \leq n$. The application of this rule reduces the size of the active set in the derived configuration, and by the induction hypothesis, there exist n final configurations $\mathcal{C}_i = \langle \emptyset; S_i; D_i; \theta_i \rangle$ each with a $\tau_i \in \Psi(D_i, S_i)$ such that $r_i \preceq_{\mathfrak{a}} y_i\theta_i\tau_i$. However, if there exist variables $z \in var(r_i) \cap var(r_j)$, for $i \neq j$, then the term $g(y_1\theta_1\tau_1, \ldots, y_n\theta_n\tau_n)$ may not be comparable to r. The example below illustrates the improper "alignment" between such generalizations.

Example 3.2.1. Consider the AUT $s \triangleq_x t$, where $s = g(\varepsilon_f, f(h(\varepsilon_f), a))$, $t = g(f(a,b), \varepsilon_f)$. The associated initial configuration to this problem is $\mathcal{C} = \langle \{s \triangleq_x t\}; \emptyset; \emptyset; \iota \rangle$. Notice that we can apply rule (Dec) over $\mathcal{C}$ generating the following derivation:

$$\langle \{g(\varepsilon_f, f(h(\varepsilon_f), a)) \triangleq_x g(f(a,b), \varepsilon_f)\}; \emptyset; \emptyset; \iota \rangle \overset{Dec}{\Longrightarrow}$$
$$\langle \{\varepsilon_f \triangleq_{y_1} f(a,b), f(h(\varepsilon_f), a) \triangleq_{y_2} \varepsilon_f\}; \emptyset; \emptyset; \{x \mapsto g(y_1, y_2)\} \rangle,$$

Also, the term r is a $\mathfrak{a}$-generalization of the above AUT:

$$r = g(\overbrace{z}^{r_1}, \overbrace{f(h(z), y)}^{r_2}),$$

which contains a shared variable $z \in var(r_1) \cap var(r_2)$. Consider the final configuration $\mathcal{C}_1 = \langle \emptyset; \{\varepsilon_f \triangleq_{z_1} a, a \triangleq_{z_4} \varepsilon_f\}; \{\star \triangleq_{z_2} b, h(\varepsilon_f) \triangleq_{z_3} \star\}; \theta_1 \rangle$ where $\theta_1 = \{y_1 \mapsto f(z_1, z_2), \ldots\}$ and $\tau_1 = \{z_2 \mapsto b, z_3 \mapsto h(\varepsilon_f)\}$ for the AUT $\varepsilon_f \triangleq_{y_1} f(a,b)$, and the final configuration $\mathcal{C}_2 = \langle \emptyset; \{\varepsilon_f \triangleq_{z_6} b, a \triangleq_{z_8} \varepsilon_f\}; \{\star \triangleq_{z_5} a, h(\varepsilon_f) \triangleq_{z_7} \star\}; \theta_2 \rangle$ where $\theta_2 = \{y_2 \mapsto f(z_7, z_8), \ldots\}$

and $\tau_2 = \{z_5 \mapsto a, z_7 \mapsto h(f(z_6, f(z_6, z_6)))\}$ for the AUT $f(h(\varepsilon_f), a) \triangleq_{y_2} \varepsilon_f$. Observe that the following holds:

$$r_1 \preceq_{\mathfrak{a}} y_1\theta_1\tau_1 = f(z_1, b)$$

$$r_2 \preceq_{\mathfrak{a}} y_2\theta_2\tau_2 = f(h(f(z_6, f(z_6, z_6))), z_8)$$

Finally, the term $g(y_1\theta_1\tau_1, y_2\theta_2\tau_2) = g(f(z_1, b), f(h(f(z_6, f(z_6, z_6))), z_8))$ is not comparable with the given $\mathfrak{a}$-generalization $r = g(z, f(h(z), y))$. ◇

To achieve the required "alignment", it is necessary to ensure that a final configuration $\langle \emptyset; S_f; D_f; \theta_f \rangle$ is reached, from which a substitution $\tau \in \Psi(D_f, S_f)$ is such that $r \preceq_{\mathfrak{a}} x\theta_f\tau$. In this manner, a generalization "aligned" with the arguments of the AUT $s \triangleq_x t$, where $s = g(s_1, \ldots, s_n)$ and $t = g(t_1, \ldots, t_n)$, in the active set of the given configuration $\mathcal{C}$ would be obtained.

To address this issue, when there exists a variable $z \in var(r_i) \cap var(r_j)$ with $i \neq j$, the approach in [9] attempts to handle the "alignment" by distinguishing between the two cases below.

1. There does not exist a position $p \in pos(s) \cap pos(t)$ such that $z \in \mathcal{G}_{\mathfrak{a}}(s|_p, t|_p)$. In other words, z generalizes terms which are absorbed during $\mathfrak{a}$-normalization of $r\sigma$ and $r\rho$, where $r\sigma \approx_{\mathfrak{a}} s$ and $r\rho \approx_{\mathfrak{a}} t$. This implies that replacing occurrences of z by ε_f, for the appropriate absorptive symbol f, within r results in a more specific generalization.

2. There exists a position $p \in pos(s) \cap pos(t)$ such that $z \in \mathcal{G}_{\mathfrak{a}}(s|_p, t|_p)$. Thus, z is structurally smaller than r and, by the induction hypothesis, there exists a final configuration $\langle \emptyset; S^*; D^*; \theta^* \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\langle \{s|_p \triangleq_z t|_p\}; \emptyset; \emptyset; \iota \rangle)$ and $\tau^* \in \Psi(D^*, S^*)$ such that $z \leq x'\theta^*\tau^*$. We will use $\theta^*\tau^*$ to align the generalizations resulting from the induction hypothesis.

But these two cases fail. The Example 3.2.2 below gives a counterexample for the first case, and the induction applied in the second case is wrong since the configuration $\langle \{s|_p \triangleq_z t|_p\}; \emptyset; \emptyset; \iota \rangle$ is not derived from $\mathcal{C}$.

Example 3.2.2 (Counterexample for Case 1)**.** Consider the terms $s = g(\varepsilon_f, \varepsilon_f)$ and $t = g(f(a, b), f(a, b))$, and let $r = g(f(x, z), f(x, z)) \in \mathcal{G}_{\mathfrak{a}}(s, t)$, where $r_1 = r_2 = f(x, z)$. Observe that $z \in var(r_1) \cap var(r_2)$ and applying the substitution proposed for this case, we obtain that $r' = g(f(x, z), f(x, z))\{z \mapsto \varepsilon_f\} \approx_{\mathfrak{a}} g(\varepsilon_f, \varepsilon_f)$, which, however, is not an $\mathfrak{a}$-generalization of s and t. ◇

The issues arising in the case analysis of the rule (Dec) within the completeness proof of [9] are closely related to the appropriate formulation of the induction hypothesis.

This observation motivated the introduction of substitution generalization (see Definition 2.1.3), and led to the formulation of Statement 3.1.

Moreover, achieving completeness requires augmenting the set of abstraction substitutions. Counterexample 3.2.3 shows that the terms generated by the abstraction set do not cover all the terms required to obtain certain `lgg`s.

Example 3.2.3 (`lgg`s Computed from the Store)**.** Consider the terms $g(f(a, g(a, c)), \varepsilon_f)$ and $g(\varepsilon_f, f(g(c, c), b))$. Applying $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ over the associated initial configuration $\mathcal{C} = \langle\{g(f(a, g(a, c)), \varepsilon_f) \triangleq_x g(\varepsilon_f, f(g(c, c), b))\}; \emptyset; \emptyset; \iota\rangle$, we obtain the following four final configurations:

$$\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}) = \left\{ \begin{array}{l} \langle \emptyset; \{a \triangleq_{y_1} \varepsilon_f, \varepsilon_f \triangleq_{z_1} g(c, c)\}; \{g(a, c) \triangleq_{y_2} \star, \star \triangleq_{z_2} b\}; \theta_1 \rangle, \\ \langle \emptyset; \{a \triangleq_{y_1} \varepsilon_f, \varepsilon_f \triangleq_{z_4} b\}; \{g(a, c) \triangleq_{y_2} \star, \star \triangleq_{z_3} g(c, c)\}; \theta_2 \rangle, \\ \langle \emptyset; \{g(a, c) \triangleq_{y_4} \varepsilon_f, \varepsilon_f \triangleq_{z_5} g(c, c)\}; \{a \triangleq_{y_3} \star, \star \triangleq_{z_6} b\}; \theta_3 \rangle, \\ \langle \emptyset; \{g(a, c) \triangleq_{y_4} \varepsilon_f, \varepsilon_f \triangleq_{z_8} b\}; \{a \triangleq_{y_3} \star, \star \triangleq_{z_7} g(c, c)\}; \theta_4 \rangle, \end{array} \right\},$$

where

$$\begin{aligned} \theta_1 &= \{x \mapsto g(f(y_1, y_2), f(z_1, z_2)), \ldots\}, \\ \theta_2 &= \{x \mapsto g(f(y_1, y_2), f(z_3, z_4)), \ldots\}, \\ \theta_3 &= \{x \mapsto g(f(y_3, y_4), f(z_5, z_6)), \ldots\}, \\ \theta_4 &= \{x \mapsto g(f(y_3, y_4), f(z_7, z_8)), \ldots\}. \end{aligned}$$

We denote the final store and delayed sets as S_i and D_i, respectively, for $i = 1, \ldots, 4$. Then the abstraction sets of each variable in the delayed set of the respective final configurations:

$$\begin{array}{ll} \uparrow_{y_2}(D_1, S_1) = \uparrow(g(a, c), \sigma^l_{S_1}) = \{g(a, c), g(y_1, c)\}, & \uparrow_{z_2}(D_1, S_1) = \uparrow(b, \sigma^r_{S_1}) = \{b\}; \\ \uparrow_{y_2}(D_2, S_2) = \uparrow(g(a, c), \sigma^l_{S_2}) = \{g(a, c), g(y_1, c)\}, & \uparrow_{z_3}(D_2, S_2) = \uparrow(g(c, c), \sigma^r_{S_1}) = \{g(c, c)\}; \\ \uparrow_{y_3}(D_3, S_3) = \uparrow(a, \sigma^l_{S_3}) = \{a\}, & \uparrow_{z_6}(D_3, S_3) = \uparrow(b, \sigma^r_{S_3}) = \{b\}; \\ \uparrow_{y_3}(D_4, S_4) = \uparrow(a, \sigma^l_{S_3}) = \{a\}, & \uparrow_{z_7}(D_4, S_4) = \uparrow(g(c, c), \sigma^r_{S_1}) = \{g(c, c)\}; \end{array}$$

The abstraction substitutions obtained are given below.

$$\begin{aligned} \Psi(D_1, S_1) &= \{\{y_2 \mapsto g(a, c), z_2 \mapsto b\}, \{y_2 \mapsto g(y_1, c), z_2 \mapsto b\}\}, \\ \Psi(D_2, S_2) &= \{\{y_2 \mapsto g(a, c), z_3 \mapsto g(c, c)\}, \{y_2 \mapsto g(y_1, c), z_3 \mapsto g(c, c)\}\}, \\ \Psi(D_3, S_3) &= \{\{y_3 \mapsto a, z_6 \mapsto b\}\}, \\ \Psi(D_4, S_4) &= \{\{y_3 \mapsto a, z_7 \mapsto g(c, c)\}\}, \end{aligned}$$

Then, the computed generalizations are the following set of incomparable lggs:

$$\left\{\begin{array}{ll} g(f(y_1, g(a,c)),\ f(z_1,b)), & g(f(y_1, g(y_1,c)),\ f(z_1,b)), \\ g(f(y_1, g(a,c)),\ f(g(c,c),z_4)), & g(f(y_1, g(y_1,c)),\ f(g(c,c),z_4)), \\ g(f(a,y_4),\ f(z_5,b)), & g(f(a,y_4),\ f(g(c,c),z_8)) \end{array}\right\}$$

◇

Example 3.2.4 (Store Completeness Counterexample)**.** Continuing Example 3.2.3. The computed set of generalizations for the AUT $g(f(a, g(a,c)), \varepsilon_f) \triangleq_x g(\varepsilon_f, f(g(c,c), b))$ is incomplete. Indeed, the terms $g(f(y_1, x_1), f(z_1, x_1))$ and $g(f(y_1, g(x_2, c)), f(z_3, x_2))$, among others, are lggs not generated by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ and Ψ.

These lggs were not generated because some abstractions of the terms in the delayed set were missing. For instance, consider the first final configuration derived in Example 3.2.3:

$$\langle \emptyset; \{a \triangleq_{y_1} \varepsilon_f, \varepsilon_f \triangleq_{z_1} g(c,c)\}; \{g(a,c) \triangleq_{y_2} \star, \star \triangleq_{z_2} b\}; \theta_1 \rangle$$

To generate the generalization $g(f(y_1, x_1), f(z_1, x_1))$ from this final configuration, it is necessary to add the substitution $\tau = \{y_2 \mapsto x_1, z_2 \mapsto x_1\}$ to $\Psi(D_1, S_1)$. It is important to observe that the associated substitutions of $g(f(y_1, x_1), f(z_1, x_1))$ are

$$\rho_l = \{y_1 \mapsto a, z_1 \mapsto \varepsilon_f, x_1 \mapsto g(a,c)\} \text{ and } \rho_r = \{y_1 \mapsto \varepsilon_f, z_1 \mapsto g(c,c), x_1 \mapsto b\}$$

which resemble the substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$. This observation suggests that the symbol $\star$ in the AUT $g(a,c) \triangleq_{y_2} \star$ is interpreted as b, whereas $\star$ in $\star \triangleq_{z_2} b$ is interpreted as $g(a,c)$.

By extending the store $S'_1 = \{g(a,c) \triangleq_{x_1} b\} \cup S_1$, we guarantee that $\tau \in \Psi(D_1, S'_1)$.

Now, consider the second non-computed lgg above, $g(f(y_1, g(x_2, c)), f(z_3, x_2))$. In this case, we must add the AUT $a \triangleq_{x_2} b$ to the store to obtain $g(x_2, c)$ in the abstraction set of y_2, and x_2 in the abstraction set of z_2. ◇

More generally, to generate the missed lggs discussed in the previous counterexample, it is necessary that in the generation of the set of abstraction substitution, the operator Ψ considers additional AUTs for every possible combination of subterms of terms s and t, for all AUTs of the form $s \triangleq_x \star$ and $\star \triangleq_y t$ appearing in the final delayed set.

A possible approach to include the missed AUTs in the construction of the set of abstraction substitutions is given in the Definition 3.2.5.

Definition 3.2.5 (Extended Store)**.** Let $\mathcal{C}$ be a configuration and the final configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$. The *extended store* of $\mathcal{C}_f$ is defined as follows:

$$S_{\mathcal{C}_f} = S_f \cup \left\{ s \triangleq_{V(s,t)} t \;\middle|\; \begin{array}{l} s \triangleq_{V(s,t)} t \text{ is solved, } s \in \mathit{subt}(\mathit{lhs}(\mathtt{a})) \text{ for some} \\ \mathtt{a} \in D_f^l,\ t \in \mathit{subt}(\mathit{rhs}(\mathtt{a}')) \text{ for some } \mathtt{a}' \in D_f^r, \\ \text{and } \nexists w \text{ such that } s \triangleq_w t \in S_f. \end{array} \right\}$$

Above, V is a bijection from pairs of terms to fresh variables.

Definition 3.2.6 (Extended Set of Abstraction Substitutions)**.** Let $\mathcal{C}$ be a configuration and $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$. The *extended set of abstraction substitutions* of $\mathcal{C}_f$ is built as $\Psi(D_f, S_{\mathcal{C}_f})$, according to the Definition 2.2.19.

Observe that the previous definition is correct. Indeed, Definition 2.2.19 of $\Psi(D, S)$, for some store S and delayed set D of a configuration, requires the following conditions:

- S must be a valid set of solved AUTs;
- D must consist of wild AUTs, and of AUTs $\mathtt{a}$ such that $\mathit{lhs}(\mathtt{a}) = \mathit{rhs}(\mathtt{a})$ with $\mathit{lhs}(\mathtt{a})$ an absorbing constant;
- the label sets must be disjoint: $\mathit{labels}(S) \cap \mathit{labels}(D) = \emptyset$.

For the configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle$, these conditions are satisfied by the sets D_f and $S_{\mathcal{C}_f}$. Indeed, by the definition of $S_{\mathcal{C}_f}$, all AUTs added to the original store (i.e., those in $S_{\mathcal{C}_f} - S_f$) have distinct fresh labels and are solved. Moreover, the condition $\mathit{labels}(S_{\mathcal{C}_f}) \cap \mathit{labels}(D_f) = \emptyset$ holds due to $\mathit{labels}(S_f) \cap \mathit{labels}(D_f) = \emptyset$ and that the added AUTs have fresh labels.

Lemma 3.2.7 (Preservation of Abstraction Substitutions)**.** *Let $\mathcal{C}_f$ be a final configuration of the form $\langle \emptyset; S_f; D_f; \theta_f \rangle$ and $S_{\mathcal{C}_f}$ the Extended Store of $\mathcal{C}_f$. Then, the extended set of abstraction substitutions is a superset of the set of abstraction substitutions:*

$$\Psi(D_f, S_f) \subseteq \Psi(D_f, S_{\mathcal{C}_f})$$

Proof: Suppose that $\tau \in \Psi(D_f, S_f)$. By Definition 2.2.19, for all $x \in \mathit{labels}(D_f)$ we have that $x\tau \in \uparrow_x(D_f, S_f)$, which means:

- $\uparrow_x(D_f, S_f) = \uparrow(\mathit{lhs}(\mathtt{a}), \sigma^l_{S_f})$ if $\mathtt{a} \in D^l$,
- $\uparrow_x(D_f, S_f) = \uparrow(\mathit{rhs}(\mathtt{a}), \sigma^r_{S_f})$ if $\mathtt{a} \in D^r$, or
- $\uparrow_x(D_f, S_f) = \uparrow(\mathit{lhs}(\mathtt{a}), \sigma^l_{S_f}) \cap \uparrow(\mathit{rhs}(\mathtt{a}), \sigma^r_{S_f})$ if $\mathtt{a} \in D^l$ if $\mathtt{a} \in D^n_f$.

Since $S_f \subseteq S_{\mathcal{C}_f}$, we have that $\sigma^l_{S_f} = (\sigma^l_{S_{\mathcal{C}_f}})|_{labels(S_f)}$ and $\sigma^r_{S_f} = (\sigma^r_{S_{\mathcal{C}_f}})|_{labels(S_f)}$. This implies that $x\tau \in \uparrow_x(D_f, S_{\mathcal{C}_f})$, for all $x \in labels(D_f)$. Then, we conclude that $\tau \in (D_f, S_{\mathcal{C}_f})$. Therefore, it follows that $\Psi(D_f, S_f) \subseteq \Psi(D_f, S_{\mathcal{C}_f})$. □

Definition 3.2.8 (Extended Computed Generalization)**.** Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a configuration, $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}(\mathcal{C})$. For all $\mathtt{a} \in A \cup S$ and $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$, the term $label(\mathtt{a})\theta_f\tau$ is called an *extended computed generalization* of the terms $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$ generated by $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}$ and the extended set of abstractions substitutions Ψ.

Lemma 3.2.9 (Soundness of Extended Set of Abstraction Substitutions for Final Configurations)**.** *Let $\mathcal{C}_f$ be a final configuration of the form $\langle \emptyset; S_f; D_f; \theta_f\rangle$. Then, for all $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$, it holds that $(\theta_f\tau)|_{labels(S_f)\cup labels(D_f)} \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}_f)$.*

Proof: We prove that the substitutions $\sigma^l_{S_{\mathcal{C}_f}}$ and $\sigma^r_{S_{\mathcal{C}_f}}$ are the associated substitutions of $\theta_f\tau$. We analyze the following cases:

1. If $\mathtt{a} \in S_f$, then since $dom(\theta_f) \cap labels(S_f) = \emptyset$ and $dom(\tau) = labels(D_f)$, it follows that $label(\mathtt{a})\theta_f\tau = label(\mathtt{a})$. Moreover, by the definition of the Extended Store, $S_f \subseteq S_{\mathcal{C}_f}$, implying that $label(\mathtt{a})\theta_f\tau\sigma^l_{S_{\mathcal{C}_f}} = lhs(\mathtt{a})$ and $label(\mathtt{a})\theta_f\tau\sigma^r_{S_{\mathcal{C}_f}} = rhs(\mathtt{a})$.
2. If $\mathtt{a} \in D_f$, we get that $dom(\theta_f) \cap labels(D_f) = \emptyset$, and then $label(\mathtt{a})\theta_f\tau = label(\mathtt{a})\tau$. Also, by Definition 2.2.19 for all $\mathtt{a} \in D_f$, $label(\mathtt{a})\theta_f\tau \in \uparrow_{label(\mathtt{a})}(D_f, S_{\mathcal{C}_f})$. Then we have that:
   (a) $label(\mathtt{a})\theta_f\tau\sigma^l_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} lhs(\mathtt{a})$ if $\mathtt{a} \in D^l_f$,
   (b) $label(\mathtt{a})\theta_f\tau\sigma^r_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} rhs(\mathtt{a})$ if $\mathtt{a} \in D^r_f$, or
   (c) $label(\mathtt{a})\theta_f\tau\sigma^l_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} lhs(\mathtt{a})$ and $label(\mathtt{a})\theta_f\tau\sigma^r_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} rhs(\mathtt{a})$ if $\mathtt{a} \in D^n_f$. □

Theorem 3.2.10 (Soundness of Extended Computed Generalizations)**.** *Let $\mathcal{C}$ be a configuration of the form $\langle A; S; D; \theta\rangle$, and $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}(\mathcal{C})$. Then for all $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ we have that the substitution $(\theta_f\tau)|_{labels(A\cup S\cup D)} \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C})$.*

Proof: We prove that the associated substitutions of $\theta_f\tau$ are $\sigma^l_{S_{\mathcal{C}_f}}$ and $\sigma^r_{S_{\mathcal{C}_f}}$. We analyze the following cases:

1. If $\mathtt{a} \in A$. Notice that $var(label(\mathtt{a})\theta_f) \subseteq labels(S_f) \cup labels(D_f)$ and by Lemma 2.3.2, $label(\mathtt{a})\theta_f \in \mathcal{G}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$. We will prove that $label(\mathtt{a})\theta_f\tau\sigma^l_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} lhs(\mathtt{a})$ and $label(\mathtt{a})\theta_f\tau\sigma^r_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} rhs(\mathtt{a})$. By Definition 1.3.15, for all $p \in P(lhs(\mathtt{a}), rhs(\mathtt{a}))$ and $q \sqsubset p$, $head((lhs(\mathtt{a})\theta_f\tau)|_q) = head((rhs(\mathtt{a})\theta_f\tau)|_q)$.

Also, by Lemma 1.3.19, the positions in $ES(s,t)$ are parallel, then it remains to prove that for each $p \in ES(lhs(\mathtt{a}), rhs(\mathtt{a}))$ it holds that $(label(\mathtt{a})\theta_f)|_p \tau \sigma^l_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} (lhs(\mathtt{a}))|_p$ and $(label(\mathtt{a})\theta_f)|_p \tau \sigma^r_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} (rhs(\mathtt{a}))|_p$. We analyze the three cases below depending on the type of position p.

- Assume that p is a *lateral-expansible position* of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$. We consider the following subcases:
  - Suppose $(lhs(\mathtt{a}))|_p = \varepsilon_f$ and $(rhs(\mathtt{a}))|_p = f(s,t)$. by Lemma 2.3.6, for each $\mathtt{a}' \in D^r_f \cup S_f$, if $label(\mathtt{a}') \in var((label(\mathtt{a})\theta_f)|_p)(\subseteq labels(D^r_f) \cup labels(S_f))$ then it occurs only once in $(label(\mathtt{a})\theta_f)|_p$. Therefore, for each $\mathtt{a}' \in D^r_f \cup S_f$, such that $label(\mathtt{a}') \in var((label(\mathtt{a})\theta_f)|_p)$, there exists a unique position $p \sqsubset p'$ such that $(f(s,t))|_{p'} = rhs(\mathtt{a}')$.
    Furthermore, for all $label(\mathtt{a}') \in var((label(\mathtt{a})\theta_f)|_p)$, by Lemma 3.2.9, we have that $label(\mathtt{a}')\tau\sigma^r_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} rhs(\mathtt{a}')$. Thus, $(label(\mathtt{a})\theta_f)|_p \tau \sigma^r_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} f(s,t)$. To conclude, by Lemma 2.3.6, $(label(\mathtt{a})\theta_f))|_p \tau \sigma^l_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} \varepsilon_f$.
  - The case in which $(lhs(\mathtt{a}))|_p = f(s,t)$ and $(rhs(\mathtt{a}))|_p = \varepsilon_f$ is treated analogously.
- Assume that p is a *trivially-decomposable position* of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$. By the first item of Lemma 2.3.7, $(lhs(\mathtt{a}))|_p$ and $(rhs(\mathtt{a}))|_p$ are the same constant. The proof proceeds by case analysis.
  - If $(lhs(\mathtt{a}))|_p$ is a non-absorbing constant. Notice that using the second item of Lemma 2.3.7, we have that $(label(\mathtt{a})\theta_f)|_p\tau = (lhs(\mathtt{a}))|_p$. This implies that $(label(\mathtt{a})\theta_f)|_p \tau \sigma^l_{S_{\mathcal{C}_f}} = (lhs(\mathtt{a}))|_p$ and $(label(\mathtt{a})\theta_f)|_p \tau \sigma^r_{S_{\mathcal{C}_f}} = (rhs(\mathtt{a}))|_p$.
  - If $(lhs(\mathtt{a}))|_p$ is an absorbing constant. Notice that the position p is also a both-expansible position. By the second item of Lemma 2.3.7, either it holds that $(label(\mathtt{a})\theta_f)|_p\tau = (lhs(\mathtt{a}))|_p$ and the analysis is analogous to the previous subcase, or $(label(\mathtt{a})\theta_f)|_p \in labels(D^n_f)$. In the latter case, by Lemma 3.2.9, we have that $(label(\mathtt{a})\theta_f)|_p \tau \sigma^l_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} (lhs(\mathtt{a}))|_p$ and $(label(\mathtt{a}')\theta_f)|_p \tau \sigma^r_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} (rhs(\mathtt{a}))|_p$.
- Assume that p is a *solvable position* of $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$. By Lemma 2.3.8, we have that $(label(\mathtt{a})\theta_f)|_p \in labels(S_f)$. Therefore, by applying Lemma 3.2.9, $(label(\mathtt{a})\theta_f)|_p \tau \sigma^l_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} (lhs(\mathtt{a}))|_p$ and $(label(\mathtt{a}')\theta_f)|_p \tau \sigma^r_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} (rhs(\mathtt{a}))|_p$.

Thus, it holds that $label(\mathtt{a})\theta_f\tau\sigma^l_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} lhs(\mathtt{a})$ and $label(\mathtt{a})\theta_f\tau\sigma^r_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} rhs(\mathtt{a})$.

2. If $\mathtt{a} \in S$, it holds that $label(\mathtt{a})\theta_f\tau = label(\mathtt{a})\theta_f$, by Lemma 2.3.2 and since $S_f \in S_{\mathcal{C}_f}$, we have that $label(\mathtt{a})\theta_f\tau\sigma^l_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} lhs(\mathtt{a})$ and $label(\mathtt{a})\theta_f\tau\sigma^r_{S_{\mathcal{C}_f}} \approx_{\mathfrak{a}} rhs(\mathtt{a})$.

3. If $\mathtt{a} \in D$, by Lemma 2.3.1, we get that $D \subseteq D_f$, and by condition 2 of Lemma 3.2.9, this case holds.

Hence, for all $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$, we have that $(\theta_f \tau)|_{labels(A \cup S \cup D)} \in \mathcal{G}_{\mathtt{a}}(\mathcal{C})$. □

Remark. Notice that the soundness of computed generalizations, Theorem 2.3.9, is an immediate consequence of Theorem 3.2.10 on the soundness of extended computed generalizations and Lemma 3.2.7 on preservation of abstraction substitutions.

Example 3.2.11 (`lgg`s Computed from the Extended Store)**.** Continuing Example 3.2.3. Consider the first final configuration in Example 3.2.3:

$$\begin{aligned}\mathcal{C}_1 = \quad & \langle \emptyset; \{a \triangleq_{y_1} \varepsilon_f, \varepsilon_f \triangleq_{z_1} g(c,c)\}; \{g(a,c) \triangleq_{y_2} \star, \star \triangleq_{z_2} b\}; \theta_1 \rangle, \\ & \text{where } \theta_1 = \{x \mapsto g(f(y_1,y_2), f(z_1,z_2)), \ldots\}.\end{aligned}$$

The extended store is given as:

$$S_{\mathcal{C}_1} = \{a \triangleq_{y_1} \varepsilon_f, \varepsilon_f \triangleq_{z_1} g(c,c)\} \cup \{g(a,c) \triangleq_{x_1} b, a \triangleq_{x_2} b, c \triangleq_{x_3} b\}.$$

Then, the extended abstraction sets for each label in the delayed set of $\mathcal{C}_1$ are given by:

$$\uparrow_{y_2}(D_1, S_{\mathcal{C}_1}) = \uparrow(g(a,c), \sigma^l_{S_{\mathcal{C}_1}}) = \{x_1, g(a,c), g(y_1,c), g(x_2,c), g(x_2,x_3), g(a,x_3), g(y_1,x_3)\},$$
$$\uparrow_{z_2}(D_1, S_{\mathcal{C}_1}) = \uparrow(b, \sigma^r_{S_{\mathcal{C}_1}}) = \{b, x_1, x_2, x_3\}.$$

Thus, the 28 set of extended abstraction substitutions, $\Psi(D_1, S_{\mathcal{C}_1})$, is given below.

$$\left\{\begin{array}{lll}
\{y_2 \mapsto x_1,\ z_2 \mapsto b\}, & \{y_2 \mapsto x_1,\ z_2 \mapsto x_1\}, & \{y_2 \mapsto x_1,\ z_2 \mapsto x_2\}, \\
\{y_2 \mapsto x_1,\ z_2 \mapsto x_3\}, & \{y_2 \mapsto g(a,c),\ z_2 \mapsto b\}, & \{y_2 \mapsto g(a,c),\ z_2 \mapsto x_1\}, \\
\{y_2 \mapsto g(a,c),\ z_2 \mapsto x_2\}, & \{y_2 \mapsto g(a,c),\ z_2 \mapsto x_3\}, & \{y_2 \mapsto g(x_2,c),\ z_2 \mapsto b\}, \\
\{y_2 \mapsto g(x_2,c),\ z_2 \mapsto x_1\}, & \{y_2 \mapsto g(a,x_3),\ z_2 \mapsto b\}, & \{y_2 \mapsto g(a,x_3),\ z_2 \mapsto x_1\}, \\
\{y_2 \mapsto g(y_1,c),\ z_2 \mapsto b\}, & \{y_2 \mapsto g(y_1,c),\ z_2 \mapsto x_1\}, & \{y_2 \mapsto g(x_2,c),\ z_2 \mapsto x_2\}, \\
\{y_2 \mapsto g(x_2,c),\ z_2 \mapsto x_3\}, & \{y_2 \mapsto g(y_1,c),\ z_2 \mapsto x_2\}, & \{y_2 \mapsto g(y_1,c),\ z_2 \mapsto x_3\}, \\
\{y_2 \mapsto g(a,x_3),\ z_2 \mapsto x_2\}, & \{y_2 \mapsto g(a,x_3),\ z_2 \mapsto x_3\}, & \{y_2 \mapsto g(x_2,x_3),\ z_2 \mapsto b\}, \\
\{y_2 \mapsto g(x_2,x_3),\ z_2 \mapsto x_1\}, & \{y_2 \mapsto g(y_1,x_3),\ z_2 \mapsto b\}, & \{y_2 \mapsto g(y_1,x_3),\ z_2 \mapsto x_1\}, \\
\{y_2 \mapsto g(y_1,x_3),\ z_2 \mapsto x_2\}, & \{y_2 \mapsto g(y_1,x_3),\ z_2 \mapsto x_3\}, & \{y_2 \mapsto g(x_2,x_3),\ z_2 \mapsto x_2\}, \\
\{y_2 \mapsto g(x_2,x_3),\ z_2 \mapsto x_3\} & &
\end{array}\right\}$$

This set of extended abstraction substitutions gives rise to 28 extended computed generalizations. By preserving only the least general ones, we obtain the six `lgg`s below.

$$\left\{\begin{array}{ll} g(f(y_1, g(a, c)), f(z_1, b)), & g(f(y_1, x_1), f(z_1, x_1)), \\ g(f(y_1, g(y_1, c)), f(z_1, b)), & g(f(y_1, g(y_1, x_3)), f(z_1, x_3)), \\ g(f(y_1, g(x_2, c)), f(z_1, x_2)), & g(f(y_1, g(a, x_3)), f(z_1, x_3)) \end{array}\right\}$$

A detailed description of all extended computed generalizations generated for all final configurations is provided in Appendix A. ◇

Finally, we present the statement of completeness that we want to achieve, obtained from Statement 3.1 by using the extended set of abstraction substitutions.

Statement 3.2 (Completeness of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ and Extended Ψ)**.** Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a configuration and $\gamma \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C})$. Then, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$ and $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ such that $(\gamma \preceq_{\mathfrak{a}} \theta_f\tau)|_{labels(A\cup S\cup D)}$.

In the following sketch, we outline the current state of a potential proof for Statement 3.2. The main obstacle lies in the base case, which concerns establishing the completeness of the extended set of abstraction substitutions for the AUTs of the delayed set.

SKETCH: We use induction on $(size(A), size(S))$.

Base Case

This case occurs when the configuration $\mathcal{C}$ is a final configuration, as will be stated and analyzed in Statement 3.3.

Induction Step

We assume that the statement holds for all the configurations with measure $(i, j) <_{lex} (k, l)$, and we aim to prove it for configurations with measure (k, l), where $k, l > 0$.

We analyze all possible cases for an AUT of the form $s \triangleq_x t \in A \cup S$. In each case, we aim to show that there exists a rule of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ that transforms the configuration $\mathcal{C} = \langle A; S; D; \theta\rangle$ into a new configuration $\mathcal{C}'$ of the form $\langle A'; S'; D'; \theta'\rangle$. Since $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ is terminating, it holds that $(size(A'), size(S')) <_{lex} (size(A), size(S))$. To apply the induction hypothesis over $\mathcal{C}'$, we construct a substitution generalization $\gamma' \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}')$ based on the given γ and its respective associated substitutions ρ_l and ρ_r. Without loss of generality, we can assume that the variables in the range of γ are different from those used in the algorithm.

Lastly, we use the more specific $\mathfrak{a}$-generalizations provided by the induction hypothesis through γ' to construct a computed $\mathfrak{a}$-generalization of s and t that is more specific than $x\gamma$ and aligned with the other $\mathfrak{a}$-generalizations. We analyze the two main cases below.

1. If $A = \emptyset$ and $s \triangleq_x t \in S$, where S is not merged. As $s \triangleq_x t$ is a solved AUT, then $x\gamma = z$. Without loss of generality, we assume that there exists an AUT $s \triangleq_y t \in S$

such that $y \neq x$ and we can apply rule (Mer) as follows:

$$\langle \emptyset; \{s \triangleq_x t, s \triangleq_y t\} \cup S_1; D; \theta \rangle \overset{Mer}{\Longrightarrow}$$
$$\langle \emptyset; \{s \triangleq_y t\} \cup S_1; D; \theta\{x \mapsto y\} \rangle,$$

Consider $\gamma' = \gamma|_{labels(S' \cup D)}$, this is a generalization substitution of $\mathcal{C}'$. Additionally, since we have that $size(A) = size(A')$ and $size(S) < size(S')$, then by induction hypothesis, there exists $\mathcal{C}_f \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}') \subseteq \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$ and $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ such that $(\gamma' \preceq_{\mathfrak{a}} \theta_f\tau)|_{labels(S' \cup D)}$, i.e., there exists δ, such that $w\gamma\delta \approx_{\mathfrak{a}} w\theta_f\tau$ for all $w \in labels(S' \cup D)$. In particular, it holds that $y\gamma\delta \approx_{\mathfrak{a}} y\theta_f\tau$, and we analyze the subcases for $x\gamma = z$ above.

(a) If $z \in dom(\delta)$ then there exists an AUT in S' of the form $s \triangleq_w t$, such that $w \neq x$ and $w\gamma = z$. Furthermore, since the final configuration is merged, we have that $w\theta_f\tau = y\theta_f\tau$. This implies that $x\gamma\delta \approx_{\mathfrak{a}} y\theta_f\tau = x\{x \mapsto y\}\theta_f\tau = x\theta_f\tau$.

(b) If $z \notin dom(\delta)$ then we can consider $\delta' = \delta\{z \mapsto x\theta_f\tau\}$, this implies that $x\gamma\delta' \approx_{\mathfrak{a}} x\theta_f\tau$.

In both cases we have that $(\gamma \preceq_{\mathfrak{a}} \theta_f\tau)|_{dom(\gamma)}$.

2. If $A \neq \emptyset$ and $s \triangleq_x t \in A$, the case analysis is made over the possibilities for s and t as below. Notice that for any rule application over A in the derivation $\mathcal{C} \Longrightarrow \mathcal{C}'$, we have that $(size(A'), size(S')) <_{lex} (size(A), size(S))$ since $size(A') < size(A)$.

(a) If $head(s) = head(t)$ and $head(s)$ is not an absorbing constant. Say that we have that $s = g(s_1, \ldots, s_n)$ and $t = g(t_1, \ldots, t_n)$. This implies that (i) $x\gamma = z$, or (ii) $x\gamma = g(r_1, \ldots, r_n)$, where r_i is a generalization of s_i and t_i, for $1 \leq i \leq n$. We apply the rule (Dec) in both subcases. Assume that the derivation is of the form:

$$\langle \{g(s_1, \ldots, s_n) \triangleq_x g(t_1, \ldots, t_n)\} \cup A_1; S; D; \theta \rangle \overset{Dec}{\Longrightarrow}$$
$$\langle A'; S; D; \theta\{x \mapsto g(x_1, \ldots, x_n)\} \rangle,$$

where $A' = \{s_1 \triangleq_{x_1} t_1, \ldots, s_n \triangleq_{x_n} t_n\} \cup A_1$ with $n \geq 0$. We analyze both cases mentioned above as follows:

i. If $x\gamma = z$, we can define γ' as

$$\gamma'(y) = \begin{cases} \gamma(y) & \text{if } y \in labels(A_1 \cup S \cup D), \\ z_i & \text{if } y = x_i, \text{ where } z_i \text{ is a fresh variable}, 1 \leq i \leq n. \end{cases}$$

Also, since $x\gamma\rho_l = z\rho_l \approx_{\mathfrak{a}} g(s_1, \ldots, s_m)$, $(z\rho_l)|_i \approx_{\mathfrak{a}} s_i$. Thus, we can define ρ'_l as follows:

$$\rho'_l(y) = \begin{cases} \rho_l(y) & \text{if } y \in \mathcal{V}ran(\gamma) \\ (z\rho_l)|_i & \text{for } z_i, 1 \leq i \leq n. \end{cases}$$

Similarly, for ρ'_r. Observe that $\gamma' \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}')$; since for all the non-wild $\mathtt{a} \in A' \cup S \cup (D - D^r)$, $label(\mathtt{a})\gamma'\rho'_l \approx_{\mathfrak{a}} lhs(\mathtt{a})$. In particular, for $s_i \triangleq_{x_i} t_i$ we have that $x_i\gamma'\rho'_l = z_i\rho'_l = (z\rho_l)|_i \approx_{\mathfrak{a}} s_i$. Similarly, for ρ'_r.

ii. If $x\gamma = g(r_1, \ldots, r_n)$, we can define γ' for $\mathcal{C}'$ as

$$\gamma'(y) = \begin{cases} \gamma(y) & \text{if } y \in labels(A_1 \cup S \cup D), \\ r_i & \text{if } y = x_i, 1 \leq i \leq n. \end{cases}$$

Notice that $\gamma' \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}')$. Indeed, for all $\mathtt{a} \in A_1 \cup S \cup (D - D^r)$, we have that $label(\mathtt{a})\gamma'\rho_l = label(\mathtt{a})\gamma\rho_l \approx_{\mathfrak{a}} lhs(\mathtt{a})$. Similarly, for $\gamma'\rho_r$. Otherwise, $x_i\gamma'\rho_l = r_i\rho_l = (x\gamma\rho_l)|_i \approx_{\mathfrak{a}} s_i$, which is assured since we are working with normal terms. Similarly, for $\gamma'\rho_r$.

Then, by the induction hypothesis, we have that there exist a final configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}') \subseteq \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$ and $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ such that $(\gamma' \preceq_{\mathfrak{a}} \theta_f\tau)|_{dom(\gamma')}$, i.e., there exists a substitution δ such that $y\gamma\delta \approx_{\mathfrak{a}} y\theta_f\tau$ for all $y \in labels(A_1 \cup S \cup D)$. To conclude, in the former case, if $x\gamma$ is headed by the symbol g, observe that

$$x\gamma\delta = g(x_1\gamma'\delta, \ldots, x_n\gamma'\delta) \approx_{\mathfrak{a}} g(x_1\theta_f\tau, \ldots, x_n\theta_f\tau) = x\theta_f\tau,$$

this implies that $(\gamma \preceq_{\mathfrak{a}} \theta_f\tau)|_{dom(\gamma)}$. In the latter case, if $x\gamma = z$ then take $\delta' = \delta\{z \mapsto x\theta_f\tau\}$ and since for all other labels, $y \in labels(A_1 \cup S \cup D)$, $y\gamma\delta' = y\gamma'\delta' \approx_{\mathfrak{a}} y\theta_f\tau$. Therefore, we got that $(\gamma \preceq_{\mathfrak{a}} \theta_f\tau)|_{dom(\gamma)}$.

(b) If $head(s) = head(t)$ and $head(s)$ is an absorbing constant, say $s = t = \varepsilon_f$. This implies that either (i) $x\gamma = z$, (ii) $x\gamma = \varepsilon_f$, or (iii) $x\gamma = f(r_1, r_2)$, where $f(r_1, r_2)$ is a collapsible term. We apply the rules (Dec) and (ExpB). Assume we have the derivations of the form:

$$\begin{array}{ccc} & \langle \{\varepsilon_f \triangleq_x \varepsilon_f\} \cup A_0; S; D; \theta \rangle & \\ \overset{\text{(Dec)}}{\swarrow\!\!\swarrow} & & \overset{\text{(ExpB)}}{\searrow\!\!\searrow} \\ \mathcal{C}_1 = \langle A_0; S; D; \theta\{x \mapsto \varepsilon_f)\} \rangle & & \langle A_0; S; D'; \theta \rangle = \mathcal{C}_2 \end{array}$$

where $D' = \{\varepsilon_f \triangleq_y \varepsilon_f\} \cup D$. The set $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}) = \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}_1) \cup \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}_2)$. We analyze the cases mentioned for $x\gamma$.

i. If $x\gamma = z$. Applying rule (Dec), we obtain the configuration $\mathcal{C}_1$. Notice that $\gamma' = \gamma|_{labels(A_0 \cup S \cup D)} \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}_1)$. By induction hypothesis, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}) \subseteq \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}_1)$ and $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ such that $(\gamma' \preceq_{\mathfrak{a}} \theta_f\tau)|_{dom(\gamma')}$, i.e., there exists δ such that $y\gamma'\delta \approx_{\mathfrak{a}} y\theta_f\tau$ for all $y \in labels(A_0 \cup S \cup D)$. Consider $\delta' = \{z \mapsto \varepsilon_f\}\delta$. Then we have that $x\gamma\delta' \approx_{\mathfrak{a}} \varepsilon_f = x\theta_f\tau$ and for all $y \in labels(A_0 \cup S \cup D)$, it holds that $y\gamma\delta' = y\gamma'\delta \approx_{\mathfrak{a}} y\theta_f\tau$. Hence, we have that $(\gamma \preceq_{\mathfrak{a}} \theta_f\tau)|_{dom(\gamma)}$.

ii. If $x\gamma = \varepsilon_f$. The analysis is analogous to the subcase above.

iii. If $x\gamma = f(r_1, r_2)$, a collapsible term. Applying the rule (ExpB), we obtain the configuration $\mathcal{C}_2$. Notice that $\gamma \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}_2)$, due to the fact that $\mathcal{C}_2$ preservers the same AUTs of $\mathcal{C}$. Then, by the induction hypothesis there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}_2)$ and $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ such that $(\gamma \preceq_{\mathfrak{a}} \theta_f\tau)|_{dom(\gamma)}$, that is that we want to prove in this case.

(c) If $head(s) \neq head(t)$ and they are not related absorptive symbols. Assume that the derivation is of the form:

$$\langle \{s \triangleq_x t\} \cup A_1; S; D; \theta \rangle \stackrel{Sol}{\Longrightarrow} \langle A_1; \{s \triangleq_x t\} \cup S; D; \theta \rangle,$$

Notice that $\gamma \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}')$, since the rule (Sol), move the AUT $s \triangleq_x t$ from the active set of $\mathcal{C}$ to the store of $\mathcal{C}'$, then $A \cup S = A' \cup S'$. By induction hypothesis, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$ and $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ such that $(\gamma \preceq_{\mathfrak{a}} \theta_f\tau)|_{dom(\gamma)}$. Thus, this subcase holds the statement.

(d) If $head(s) \neq head(t)$ such that $head(s) = \varepsilon_f$ and $head(t) = f$. This implies that (i) $x\gamma = y$, or (ii) $x\gamma = f(r_1, r_2)$ is a collapsible term. We apply the rules **(ExpLA1)** and **(ExpLA2)**. Assume we have the derivations of the form:

$$\begin{array}{ccc} & \langle \{\varepsilon_f \triangleq_x f(t_1, t_2)\} \cup A_0; S; D; \theta \rangle & \\ \stackrel{\text{(ExpLA1)}}{\swarrow\!\!\swarrow} & & \searrow\!\!\searrow^{\text{(ExpLA2)}} \\ \mathcal{C}_1 = \langle A_1; S; D_1; \theta\{x \mapsto f(x_1, x_2)\} \rangle & & \langle A_2; S; D_2; \theta\{x \mapsto f(x_3, x_4)\} \rangle = \mathcal{C}_2 \end{array}$$

where $A_1 = \{\varepsilon_f \triangleq_{x_1} t_1\} \cup A_0$, $D_1 = \{\star \triangleq_{x_2} t_2\} \cup D$, $A_2 = \{\varepsilon_f \triangleq_{x_4} t_2\} \cup A_0$, and $D_2 = \{\star \triangleq_{x_3} t_1\} \cup D$. The set $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}) = \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}_1) \cup \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}_2)$. We analyze the cases mentioned for $x\gamma$.

i. Case $x\gamma = z$. Since $\gamma \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C})$, there exist ρ_l and ρ_r such that for all non-wild AUT $\mathtt{a} \in A \cup S \cup D$, $label(\mathtt{a})\rho_l \approx_{\mathfrak{a}} lhs(\mathtt{a})$ and $label(\mathtt{a})\rho_r \approx_{\mathfrak{a}} rhs(\mathtt{a})$. In particular, for $\varepsilon_f \triangleq_x f(t_1, t_2) \in A$, it holds that $x\gamma\rho_l = z\rho_l \approx_{\mathfrak{a}} \varepsilon_f$ and $x\gamma\rho_r = z\rho_l \approx_{\mathfrak{a}} f(t_1, t_2)$. Without loss of generality, we consider the derivation of (ExpLA1) and define γ' as

$$\gamma'(y) = \begin{cases} \gamma(y) & \text{if } y \in \mathit{labels}(A_0 \cup S \cup D); \\ z_i & \text{if } y = x_i, \text{ for } i \in \{1,2\}. \end{cases}$$

To prove that $\gamma' \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}_1)$, consider ρ'_l and ρ'_r defined as below.

$$\rho'_l(y) = \begin{cases} \rho_l(y) & \text{if } y \in \mathcal{V}\mathit{ran}(\gamma) \\ \varepsilon_f & \text{if } y = z_i, \text{ for } i \in \{1,2\}. \end{cases}$$
$$\rho'_r(y) = \begin{cases} \rho_r(y) & \text{if } y \in \mathcal{V}\mathit{ran}(\gamma) \\ t_i & \text{if } y = z_i, \text{ for } i \in \{1,2\}. \end{cases}$$

Moreover, notice that for all $\mathtt{a} \in A_0 \cup S \cup (D_1 - D_1^r)$, we have that $\mathit{label}(\mathtt{a})\gamma'\rho'_l = \mathit{label}(\mathtt{a})\gamma\rho_l \approx_{\mathfrak{a}} \mathit{lhs}(\mathtt{a})$ and for all $\mathtt{a} \in A_0 \cup S \cup (D_1 - D_1^l)$, $\mathit{label}(\mathtt{a})\gamma'\rho'_r = \mathit{label}(\mathtt{a})\gamma\rho_r \approx_{\mathfrak{a}} \mathit{rhs}(\mathtt{a})$. To conclude that $\gamma' \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}_1)$, notice that $x_1\gamma'\rho'_l = z_1\rho'_l = \varepsilon_f$, $x_1\gamma'\rho'_r = z_1\rho'_r = t_1$, and $x_2\gamma'\rho'_r = z_2\rho_r = t_2$.
By induction hypothesis, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$ and $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ such that $(\gamma' \preceq_{\mathfrak{a}} \theta_f\tau)|_{\mathit{dom}(\gamma')}$, i.e., there exists δ such that $y\gamma'\delta \approx_{\mathfrak{a}} y\theta_f\tau$ for all $y \in \mathit{labels}(A_0 \cup S \cup D_1)$. Take $\delta' = \delta\{z \mapsto x\theta_f\tau\}$, then $x\gamma\delta' = z\delta' = x\theta_f\tau$. This implies that $(\gamma \preceq_{\mathfrak{a}} \theta_f\tau)|_{\mathit{labels}(A \cup S \cup D)}$.

ii. Case $x\gamma = f(r_1, r_2)$, a collapsible term. We analyze the following subcases:

A. If r_1 is a collapsible subterm of $x\gamma$ and r_2 is not. This implies that we need to apply the rule (ExpLA1) and define γ' for $\mathcal{C}_1$ as follows:

$$\gamma'(y) = \begin{cases} \gamma(y) & \text{if } y \in \mathit{labels}(A_0 \cup S \cup D); \\ r_i & \text{if } y = x_i, \text{ for } i \in \{1,2\}. \end{cases}$$

We need to prove that $\gamma' \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}_1)$. Observe that the associated substitutions of γ' are the substitutions ρ'_l and ρ'_r defined below.

$$\rho'_l(y) = \begin{cases} \rho_l(y) & \text{if } y \in \mathcal{V}\mathit{ran}(\gamma) \\ \varepsilon_f & \text{if } y = x'_i, \text{ for } i \in \{1,2\}. \end{cases}$$
$$\rho'_r(y) = \begin{cases} \rho_r(y) & \text{if } y \in \mathcal{V}\mathit{ran}(\gamma) \\ t_i & \text{if } y = x'_i, \text{ for } i \in \{1,2\}. \end{cases}$$

By induction hypothesis, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$ and $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ such that $(\gamma' \preceq_{\mathfrak{a}} \theta_f\tau)|_{\mathit{dom}(\gamma')}$, i.e., there exists δ such that $y\gamma'\delta \approx_{\mathfrak{a}} y\theta_f\tau$ for all $y \in \mathit{labels}(A_1 \cup S \cup D_1)$. Moreover, notice that $x\gamma\delta = f(x_1\gamma'\delta, x_2\gamma'\delta) \approx_{\mathfrak{a}} f(x_1\theta_f\tau, x_2\theta_f\tau) = x\theta_f\tau$. Thus, it holds that $(\gamma \preceq_{\mathfrak{a}} \theta_f\tau)|_{\mathit{labels}(A \cup S \cup D)}$.

B. If r_2 is a collapsible subterm of $x\gamma$ and r_2 is not. This case is analogous to the previous case using the rule (ExpLA2).

C. If r_1 and r_2 are both collapsible subterms of $x\gamma$. This case is analogous to the two cases above; it is possible to use either rule (ExpLA1) or (ExpLA2).

(e) If $head(s) \neq head(t)$ such that $head(s) = f$ and $head(t) = \varepsilon_f$. This implies that (i) $x\gamma = y$, or (ii) $x\gamma = f(r_1, r_2)$ is a collapsible term. This case is analogous to the case above using the rules (ExpRA1) and (ExpRA2). □

Statement 3.3 (Completeness of Set of Extended Abstraction Substitutions)**.** Let $\mathcal{C}_f$ be a final configuration of the form $\langle \emptyset; S_f; D_f; \theta_f \rangle$ and $\gamma \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}_f)$ with the associated substitutions ρ_l and ρ_r. Then, there exist $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ and δ such that:

- For all $s \triangleq_x t \in S_f$, $x\gamma\delta \approx_{\mathfrak{a}} x$.
- For all $s \triangleq_x t \in D_f$, $x\gamma\delta \approx_{\mathfrak{a}} x\tau \in \uparrow_x(D_f, S_{\mathcal{C}_f})$.

Sketch: The idea of the proof is to construct a substitution $\tau \in \Psi(D_f, S_{\mathcal{C}_f})$ satisfying the conditions of the statement. This construction relies on the terms in the range of ρ_l and ρ_r. Since the domain of these substitutions is $var(label(\mathtt{a})\gamma)$, we must find a substitution δ that instantiates the variables of all terms $label(\mathtt{a})\gamma$ in such a way that each $label(\mathtt{a})\gamma\delta$ can be generated by its respective abstraction set. The substitution δ is obtained from the following set of valid AUTs:

$$\mathcal{R} = \{w\rho_l \triangleq_w w\rho_r \mid w \in \mathcal{V}ran(\gamma)\}$$

We partition the set into the following subsets, depending on the relation between $w\rho_l$ and $w\rho_r$:

- $\mathcal{R}_s = \left\{ w\rho_l \triangleq_w w\rho_r \in \mathcal{R} \;\middle|\; \begin{array}{l} w\rho_l = lhs(\mathtt{a}) \text{ and } w\rho_r = rhs(\mathtt{a}) \\ \text{for some } \mathtt{a} \in S_{\mathcal{C}_f} \end{array} \right\}$
- $\mathcal{R}_u = \left\{ w\rho_l \triangleq_w w\rho_r \in \mathcal{R} - \mathcal{R}_s \;\middle|\; \begin{array}{l} w\rho_l \in subt(lhs(\mathtt{a})),\ w\rho_r \in subt(rhs(\mathtt{a}')) \text{ for} \\ \mathtt{a} \in D_f^l \text{ and } \mathtt{a}' \in D_f^r \end{array} \right\}$
- $\mathcal{R}_l = \left\{ w\rho_l \triangleq_w w\rho_r \in \mathcal{R} \;\middle|\; \begin{array}{l} w\rho_l \in subt(lhs(\mathtt{a})),\ w\rho_r \notin subt(rhs(\mathtt{a}')) \text{ for} \\ \mathtt{a} \in D_f^l \text{ and } \mathtt{a}' \in D_f^r \end{array} \right\}$
- $\mathcal{R}_r = \left\{ w\rho_l \triangleq_w w\rho_r \in \mathcal{R} \;\middle|\; \begin{array}{l} w\rho_l \notin subt(lhs(\mathtt{a})),\ w\rho_r \in subt(rhs(\mathtt{a}')) \text{ for} \\ \mathtt{a} \in D_f^l \text{ and } \mathtt{a}' \in D_f^r \end{array} \right\}$
- $\mathcal{R}_e = \left\{ w\rho_l \triangleq_w w\rho_r \in \mathcal{R} \;\middle|\; \begin{array}{l} w\rho_l \notin subt(lhs(\mathtt{a})),\ w\rho_r \notin subt(rhs(\mathtt{a}')) \text{ for} \\ \mathtt{a}' \in D_f^l \text{ and } \mathtt{a} \in D_f^r \end{array} \right\}$

Consider the following substitutions:

- $\delta_s = \{w \mapsto \mathit{label}(\mathtt{a}) \mid w \in \mathit{labels}(\mathcal{R}_s), x\rho_l = \mathit{lhs}(\mathtt{a}), w\rho_r = \mathit{rhs}(\mathtt{a}),\ \text{for some } \mathtt{a} \in S_{\mathcal{C}_f}\}$.
- $\delta_e = \{w \mapsto \mathit{lhs}(\mathtt{a}) \mid w \in \mathit{labels}(\mathcal{R}_e),\}$

Moreover, for all $w\rho_l \triangleq_w w\rho_r \in \mathcal{R}_u$ we have $w\rho_l \in \mathit{subt}(\mathit{lhs}(\mathtt{a}))$ for some $\mathtt{a} \in D_j^l$ and $w\rho_r \in \mathit{subt}(\mathit{rhs}(\mathtt{a}'))$ for some $\mathtt{a}' \in D_j^r$, but $w\rho_l \triangleq_w w\rho_r \notin \mathcal{R}_s$, i.e., there does not exist an AUT $\mathtt{a}'' \in S_{\mathcal{C}_f}$ such that $\mathit{lhs}(\mathtt{a}'') = w\rho_l$ and $\mathit{rhs}(\mathtt{a}'') = w\rho_r$. This implies that $w\rho_l \triangleq_w w\rho_r$ is not solved, and there exists at least one AUT $(w\rho_l)|_p \triangleq_z (w\rho_r)|_q \in S_{\mathcal{C}_f}$ such that $p \in \mathit{pos}(w\rho_l) - \{\epsilon\}$, $q \in \mathit{pos}(w\rho_r) - \{\epsilon\}$, $\mathit{head}((w\rho_l)|_p) \neq \mathit{head}((w\rho_r)|_q)$ (non-absorptive related symbols), and $z = V((w\rho_l)|_p, (w\rho_r)|_q)$. This holds since $S_{\mathcal{C}_f}$ contains all combinations of subterms of the wild AUTs, and such that the pair of subterms form a solved AUT. Finally, this implies that it is possible to generate a more specific term, say r_w, which is a generalization of $w\rho_l$ and $w\rho_r$ and is generated by the subterms of the abstraction substitutions of $\uparrow(\mathit{lhs}(\mathtt{a}), \sigma^l_{S_{\mathcal{C}_f}})$ and $\uparrow(\mathit{rhs}(\mathtt{a}'), \sigma^r_{S_{\mathcal{C}_f}})$. Finally, we define the substitution $\delta_u = \{\, w \mapsto r_w \mid w \in \mathit{labels}(\mathcal{R}_u) \,\}$ and the substitution $\delta = \delta_s\delta_e\delta_u(\rho_l)|_{\mathit{labels}(\mathcal{R}_l)}(\rho_r)|_{\mathit{labels}(\mathcal{R}_r)}$. It remains to prove the following:

1. All the sets $\mathcal{R}_s, \mathcal{R}_u, \mathcal{R}_l, \mathcal{R}_r, \mathcal{R}_e$ are pairwise disjoint and the union is the set $\mathcal{R}$.
2. For all $s \triangleq_x t \in S_f$, we have that $x\gamma\delta_s = x$.
3. $\tau = \{x \mapsto x\gamma\delta \mid x \in \mathit{labels}(D_f)\} \in \Psi(D_f, S_{\mathcal{C}_f})$, i.e., for all $s \triangleq_x t \in D_f$, it holds that $x\gamma\delta \in \uparrow(D_f, S_{\mathcal{C}_f})$.

□

Example 3.2.12. This example illustrates the intuition of the idea of sketch presented above. Consider the following final configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle$ where

$$S_f = \{a \triangleq_{x_1} \varepsilon_f, \varepsilon_f \triangleq_{x_2} b, a \triangleq_{x_3} h(a)\}$$

$$D_f = \{\varepsilon_f \triangleq_{x_4} \varepsilon_f, h(\varepsilon_f) \triangleq_{x_5} \star, \star \triangleq_{x_6} g(b, h(a)), \star \triangleq_{x_7} h(f(a, a))\}$$

We build the τ for the following substitutions generalization γ of $\mathcal{C}_f$:

$$\gamma = \left\{ \begin{array}{l} x_1 \mapsto\ y_1,\quad x_2 \mapsto y_2,\quad x_3 \mapsto\ y_3,\quad x_4 \mapsto\ f(f(y_1, y_2), h(y_4)), \\ x_5 \mapsto h(f(a, f(w_1, w_2))), x_6 \mapsto g(y_2, w_4), x_7 \mapsto h(f(w_3, y_4)) \end{array} \right\}$$

Consider the extended store set for this case.

$$S_{C_f} = S_f \cup \{h(\varepsilon_f) \triangleq_{x_8} g(b, h(a)), \varepsilon_f \triangleq_{x_9} g(b, h(a)), \varepsilon_f \triangleq_{x_{10}} h(a), \varepsilon_f \triangleq_{x_{11}} a\}$$

The associated substitutions of γ are given by:

$$\begin{aligned}\rho_l = &\ \{y_1 \mapsto a, y_2 \mapsto \varepsilon_f, y_3 \mapsto a, y_4 \mapsto \varepsilon_f, w_1 \mapsto \varepsilon_f, w_2 \mapsto g(a,b), w_3 \mapsto b, w_4 \mapsto h(\varepsilon_f)\}\\ \rho_r = &\ \{y_1 \mapsto \varepsilon_f, y_2 \mapsto b, y_3 \mapsto h(a), y_4 \mapsto a, w_1 \mapsto h(b), w_2 \mapsto c, w_3 \mapsto a, w_4 \mapsto h(a)\}\end{aligned}$$

We build the set $\mathcal{R}$ of all of the following AUTs:

$$\mathcal{R} = \left\{ \begin{array}{l} a \triangleq_{y_1} \ \varepsilon_f,\ \varepsilon_f \triangleq_{y_2} b,\ a \triangleq_{y_3} \ h(a),\ \varepsilon_f \triangleq_{y_4} a,\ \varepsilon_f \triangleq_{w_1} h(b), \\ g(a,b) \triangleq_{w_2} c,\ b \triangleq_{w_3} a,\ h(\varepsilon_f) \triangleq_{w_4} h(a) \end{array} \right\}$$

The partition of the set $\mathcal{R}$:

- $\mathcal{R}_s = \{a \triangleq_{y_1} \varepsilon_f, \varepsilon_f \triangleq_{y_2} b, a \triangleq_{y_3} h(a), \varepsilon_f \triangleq_{y_4} a\}$
- $\mathcal{R}_u = \{h(\varepsilon_f) \triangleq_{w_4} h(a)\}$
- $\mathcal{R}_l = \{\varepsilon_f \triangleq_{w_1} h(b)\}$
- $\mathcal{R}_r = \{b \triangleq_{w_3} a\}$
- $\mathcal{R}_e = \{g(a,b) \triangleq_{w_2} c\}$

We calculate the four substitutions mentioned in the sketch of the Statement 3.3.

$$\begin{aligned}\delta_s = &\ \{y_1 \mapsto x_1, y_2 \mapsto x_2, y_3 \mapsto x_3, y_4 \mapsto x_{11}\}\\ \delta_e = &\ \{w_2 \mapsto g(a,b)\}\\ \delta_u = &\ \{w_4 \mapsto h(x_{11})\}\\ (\rho_l)|_{labels(\mathcal{R}_l)} = &\ \{w_1 \mapsto \varepsilon_f\}\\ (\rho_r)|_{labels(\mathcal{R}_r)} = &\ \{w_3 \mapsto a\}\\ \delta = &\ \delta_s\delta_e\delta_u(\rho_l)|_{labels(\mathcal{R}_l)}(\rho_r)|_{labels(\mathcal{R}_r)}\end{aligned}$$

Notice that for the AUT $h(\varepsilon_f) \triangleq_{w_4} h(a) \in \mathcal{R}_u$, there exists $\varepsilon_f \triangleq_{x_{11}} a \in S_{\mathcal{C}_f}$, then $h(x_{11})$ is a generalization of $h(\varepsilon_f)$ and $h(a)$, and such that $h(x_{11}) \in subt(g(x_2, h(x_{11})))$, where $g(x_2, h(x_{11})) \in \uparrow(g(b, h(a)), \sigma^r_{S_{\mathcal{C}_f}})$. Applying substitution δ to each term $x_i\gamma$, for $1 \leq i \leq 7$, we get the following set

$$\{y_1,\ y_2,\ y_3,\ f(f(x_1,x_2),h(x_{11})),\ h(f(a,f(\varepsilon_f,g(a,b)))),\ g(x_2,w_4), h(f(a,x_{11}))\}$$

Observe that these terms are holding:

$$
\begin{aligned}
f(f(x_1, x_2), h(x_{11})) \in &\ \uparrow(\varepsilon_f, \sigma^l_{S_{\mathcal{C}_f}}) \cap \uparrow(\varepsilon_f, \sigma^r_{S_{\mathcal{C}_f}}) \\
h(f(a, f(\varepsilon_f, g(a, b)))) \approx_{\mathfrak{a}} h(\varepsilon_f) \in &\ \uparrow(h(\varepsilon_f), \sigma^l_{S_{\mathcal{C}_f}}) \\
g(x_2, h(x_{11})) \in &\ \uparrow(g(b, h(a)), \sigma^r_{S_{\mathcal{C}_f}}) \\
h(f(a, x_{11})) \in &\ \uparrow(h(f(a, a)), \sigma^r_{S_{\mathcal{C}_f}})
\end{aligned}
$$

It means, $\tau = \{x_i \mapsto x_i\gamma\delta_s\delta_e\delta_u(\rho_l)|_{labels(\mathcal{R}_l)}(\rho_r)|_{labels(\mathcal{R}_r)} \mid 4 \leq i \leq 7\} \in \Psi(D_f, S_{\mathcal{C}_f})$. $\diamond$

Even though the extended store and the extended abstraction substitutions produce additional, and in some cases comparable, generalizations, the situation illustrated in Example 2.2.20 remains unaffected. The extended store corresponds to the same final store, generating an infinite set of incomparable generalizations, and the absorptive anti-unification problem continues to be at least infinitary. Moreover, if completeness is achieved, the absorptive anti-unification problem will be infinitary. In this case, the existence of a procedure for absorptive matching would allow the sets of computed generalizations to be minimized, enabling the determination of a minimal complete set of generalizations for any pair of terms.

Chapter 4

Linear Absorptive Anti-unification

This Chapter treats an important variant of the absorptive anti-unification: the linear absorptive anti-unification. This variant consists of finding linear generalizations for any pair of terms modulo $\mathfrak{a}$. The set of linear $\mathfrak{a}$-generalizations of the terms s and t is denoted as $\mathcal{G}^{\ell}_{\mathfrak{a}}(s,t)$. This chapter aims to present the $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$ algorithm and to prove that, together with the *linear abstraction substitution*, also introduced in this chapter, it produces the minimal complete set of linear $\mathfrak{a}$-generalizations for any pair of terms s and t.

4.1 Generalization Algorithm for Linear Variant of $\mathfrak{a}$-Theories

The example below illustrates how treating the linear variant can reduce the number of solutions. The goal is computing linear least general $\mathfrak{a}$-generalizations.

Example 4.1.1. Consider the infinite set of $\mathfrak{a}$-generalizations of the terms $g(\varepsilon_f, a)$ and $g(f(h(\varepsilon_f), a), \varepsilon_f)$ in Example 2.2.20. The linear $\mathfrak{a}$-generalizations of these terms are only the terms $g(f(y_1, a), w_2)$ and $g(f(h(\varepsilon_f), y_4), w_2)$. $\diamond$

Restricting attention to linear solutions simplifies the completeness analysis; however, this restriction necessitates the introduction of new notions specific to linearity.

Firstly, every initial configuration (see Remark 2.1) admits linear generalizations, although these generalizations are not necessarily computed by the $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ procedure.

Example 4.1.2. Consider the configuration $\langle\{g(a,a) \triangleq_x g(b,b)\}; \emptyset; \emptyset; \iota\rangle$. The $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ algorithm computes the generalization $g(y,y)$; however, the only linear `lgg` is $g(y,z)$. $\diamond$

Definition 4.1.3 (Linear Configurations)**.** Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a configuration, $\mathcal{C}$ is a *linear configuration* if θ is a linear substitution.

Example 4.1.4. Consider the configuration $\langle\{\varepsilon_f \triangleq_{y_1} h(\varepsilon_f), a \triangleq_{w_2} \varepsilon_f\}; \emptyset; \{\star \triangleq_{y_2} a\}; \theta\rangle$, where $\theta = \{x \mapsto g(f(y_1, y_2), w_2), w_1 \mapsto f(y_1, y_2)\}$. This intermediate configuration of Derivation 1 in Example 2.1.16 is a linear configuration since the terms $g(f(y_1, y_2), w_2)$ and $f(y_1, y_2)$ in the range of θ are linear terms. $\diamond$

Notice that an initial configuration is also a linear configuration; indeed, this holds since the substitution ι is a renaming, then each term in the range is a variable.

Definition 4.1.5 (Linear Substitution Generalization)**.** Let $\mathcal{C}$ be a linear configuration with active set A. A *linear substitution generalization* is a substitution $\gamma \in \mathcal{G}_{\mathfrak{a}}(A)$ such that the following conditions are satisfied:

1. γ is a linear substitution;
2. for all different AUTs $\mathtt{a}, \mathtt{a}' \in A$ we have that $var(label(\mathtt{a})\gamma) \cap var(label(\mathtt{a}')\gamma) = \emptyset$.

For short, we will write "γ is a linear generalization of $\mathcal{C}$". The set of linear substitution generalizations of $\mathcal{C}$ is denoted as $\mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C})$.

For proving that $\gamma \in \mathcal{G}_{\mathfrak{a}}(A)$ for some non-wild valid set of AUTs A, we must provide substitutions ρ_l and ρ_r that instantiate the variables in the generalizations of all AUTs in A to their respective left-hand sides and right-hand sides. In the linear variant, it suffices to prove that, for all $\mathtt{a} \in A$, the term $label(\mathtt{a})\gamma$ belongs to $\mathcal{G}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$.

Lemma 4.1.6 (Characterization of Linear Substitutions Generalizations)**.** *Let A be a valid set of non-wild AUTs, and let γ be a substitution such that $dom(\gamma) = labels(A)$, and such that for all distinct AUTs $\mathtt{a}, \mathtt{a}' \in A$, we have that $var(label(\mathtt{a})\gamma) \cap var(label(\mathtt{a}')\gamma) = \emptyset$. Then, $label(\mathtt{a})\gamma \in \mathcal{G}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$ for all $\mathtt{a} \in A$ if and only if $\gamma \in \mathcal{G}_{\mathfrak{a}}(A)$.*

Proof: Assume that for all $\mathtt{a} \in A$ it holds that $label(\mathtt{a})\gamma \in \mathcal{G}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$. Then, for each $\mathtt{a} \in A$, there exist substitutions $\rho_l(\mathtt{a})$ and $\rho_r(\mathtt{a})$ such that $label(\mathtt{a})\gamma\rho_l(\mathtt{a}) \approx_{\mathfrak{a}} lhs(\mathtt{a})$ and $label(\mathtt{a})\gamma\rho_r(\mathtt{a}) \approx_{\mathfrak{a}} rhs(\mathtt{a})$.
Since $var(label(\mathtt{a})\gamma) \cap var(label(\mathtt{a}')\gamma) = \emptyset$ for all distinct $\mathtt{a}, \mathtt{a}' \in A$, it follows that the domains of the substitutions $\rho_l(\mathtt{a})$ and $\rho_l(\mathtt{a}')$ (similarly for ρ_r) are disjoint. Let $A = \{\mathtt{a}_1, \ldots, \mathtt{a}_n\}$. Therefore, the compositions substitution $\rho_l = \rho_l(\mathtt{a}_1) \cdots \rho_l(\mathtt{a}_n)$, and $\rho_r = \rho_r(\mathtt{a}_1) \cdots \rho_r(\mathtt{a}_n)$ are well-defined, and they form associated substitutions for γ with $dom(\gamma\rho_l) = dom(\gamma\rho_r) = labels(A)$. Hence, it holds that $\gamma \in \mathcal{G}_{\mathfrak{a}}(A)$.
Conversely, suppose that $\gamma \in \mathcal{G}_{\mathfrak{a}}(A)$. By Definition 1.3.33, there exist substitutions ρ_l and ρ_r such that for all $\mathtt{a} \in A$, it holds that $label(\mathtt{a})\gamma\rho_l \approx_{\mathfrak{a}} lhs(\mathtt{a})$ and $label(\mathtt{a})\gamma\rho_r \approx_{\mathfrak{a}} rhs(\mathtt{a})$, i.e., $label(\mathtt{a})\gamma \in \mathcal{G}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$. $\square$

Corollary 4.1.7. *Let A be a valid set of non-wild AUTs, and let γ be a substitution such that $dom(\gamma) = labels(A)$, and such that for all distinct AUTs $\mathtt{a}, \mathtt{a}' \in A$, we have that $var(label(\mathtt{a})\gamma) \cap var(label(\mathtt{a}')\gamma) = \emptyset$. Then, $label(\mathtt{a})\gamma \in \mathcal{G}^{\ell}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$ for all $\mathtt{a} \in A$ if and only if $\gamma \in \mathcal{G}^{\ell}_{\mathfrak{a}}(A)$.*

Definition 4.1.8 (Algorithm $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$)**.** The algorithm for the linear variant of absorptive anti-unification, denoted by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$, consists of the exhaustive application of the inference rules of the $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ algorithm (see Table 2.1), except for the rules (Mer) and (ExpB).

These rules are dropped since rule (Mer) duplicates generalization variables, and the rule (ExpB) leads to non-linear generalizations.

Example 4.1.9. Consider the configuration $\langle\{g(a,a) \triangleq_x g(b,b)\}; \emptyset; \emptyset; \iota\rangle$ given in Example 4.1.2, the procedure $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$ computes only the $\mathtt{lgg}\ g(y,z)$. Note that the non-linear $\mathtt{lgg}\ g(y,y)$ computed by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ is not derived by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$, since the latter excludes the rule (Mer). ◇

Example 4.1.10. Consider the terms $g(\varepsilon_f, \varepsilon_f, a)$ and $g(\varepsilon_f, b, \varepsilon_f)$ of Example 2.2.21. In this example, the generalizations $g(f(w_3, w_2), w_2, w_3)$ and $g(f(f(w_3, a), w_2), w_2, w_3)$ were obtained after application of the rule (ExpB) and the use of the abstraction substitutions. Note that these generalizations are not linear.

Moreover, considering a linear version of these generalizations, i.e., $g(f(w_1, w_4), w_2, w_3)$ and $g(f(f(w_1, a), w_4), w_2, w_3)$, both of them are more general than $g(\varepsilon_f, w_2, w_3)$, which is a generalization computed by (Dec) rule. ◇

The Examples 4.1.9 and 4.1.10 illustrate why these rules are not considered in the algorithm for computing linear generalizations, and the next lemmas prove this fact.

Lemma 4.1.11. *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a configuration and $\mathcal{C}_f \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C})$ of the form $\mathcal{C}_f = \langle\emptyset; S_f; D_f; \theta_f\rangle$ and $\tau \in \Psi(D_f, S_f)$ where at least one application of (Mer) was made in the derivation, then $(\theta_f\tau)|_{labels(A)}$ is a non-linear substitution generalization of $\mathcal{C}$.*

Proof: As rule (Mer) was applied at least once, then we have the following derivation:

$$\mathcal{C} \Longrightarrow^{*} \langle\{\emptyset; \{s' \triangleq_x t', s' \triangleq_y t'\} \cup S; D; \theta'\rangle \overset{Mer}{\Longrightarrow}$$
$$\langle\emptyset; \{s' \triangleq_y t'\} \cup S; D; \theta'\{x \mapsto y\}\rangle \Longrightarrow^{*} \mathcal{C}_f$$

From definition of configuration $x, y \in \mathcal{V}ran(\theta')$ and we analyze the following two cases:

1. If $x, y \in var(y_1\theta')$ for some $y_1 \in labels(A)$. This implies that the variable y occurs twice in $y_1\theta'\{x \mapsto y\}$ and we have the two subcases:

(a) If there does not exist an AUT $s' \triangleq_z t' \in S$ with $z \neq y$ then y remains in the range of θ_f and as $y\tau = y$ for all $\tau \in \Psi(D_f, S_f)$, then $y_1\theta_f\tau$ is a non-linear term.

(b) Otherwise, assume that there exist m AUTs $s' \triangleq_{z_m} t' \in S$ with $m \geq 1$, and $z_i \neq z_j$ for $i \neq j$. Without loss of generality, assume that the configuration after m applications of rule (Mer) has the form $\langle \emptyset; \{s' \triangleq_z t'\} \cup S'; D; \theta'\{x \mapsto y\}\nu \rangle$, where $\nu = \{y \mapsto z, z_j \mapsto z\}$ with $z = z_i$ for some $1 \leq i \leq m$ and for $1 \leq j \leq n$ and $j \neq i$, then we have that z occurs at least two times in $y_1\theta_f$, implying that $y_1\theta_f\tau$ is a not linear term, for all $\tau \in \Psi(D_f, S_f)$.

In both cases, $(\theta_f\tau)|_{labels(A)}$ is not a linear substitution and the lemma holds.

2. If $x \in var(y_1\theta')$ and $y \in var(y_2\theta')$ for some $y_1, y_2 \in labels(A)$ with $y_1 \neq y_2$. We analyze the following two subcases:

(a) If there does not exist an AUT $s' \triangleq_z t' \in S$ with $z \neq y$ then $y \in var(y_1\theta_f)$ and $y \in var(y_2\theta_f)$, implying that $var(y_1\theta_f\tau) \cap var(y_2\theta_f\tau) \neq \emptyset$ for all $\tau \in \Psi(D_f, S_f)$.

(b) Otherwise, assume that there exist m AUTs $s' \triangleq_{z_m} t' \in S$ with $m \geq 1$, and $z_i \neq z_j$ for $i \neq j$. Without loss of generality, assume that the configuration after m applications of rule (Mer) has the form $\langle \emptyset; \{s' \triangleq_z t'\} \cup S'; D; \theta'\{x \mapsto y\}\nu \rangle$, where $\nu = \{y \mapsto z, z_j \mapsto z\}$ with $z = z_i$, for some $1 \leq i \leq m$ and for $1 \leq j \leq n$ and $j \neq i$, then $z \in var(y_1\theta_f)$ and $z \in var(y_2\theta_f)$, implying that $var(y_1\theta_f\tau) \cap var(y_2\theta_f\tau) \neq \emptyset$ for all $\tau \in \Psi(D_f, S_f)$.

In both cases, $(\theta_f\tau)|_{labels(A)}$ is not a linear substitution generalization of $\mathcal{C}$, as condition 2 of the definition does not hold. □

Lemma 4.1.12. *Let $\mathcal{C} = \langle A; \emptyset; \emptyset; \iota \rangle$ be an initial configuration and $\mathcal{C}_f \in \mathcal{A}_{\mathfrak{a}\text{-}AUNIF}(\mathcal{C})$ of the form $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle$ and $\tau \in \Psi(D_f, S_f)$, where at least one application of (ExpB) was made in the derivation. Then $(\theta_f\tau)|_{labels(A)}$ is a non-linear substitution generalization of $\mathcal{C}$ or there exists $\mathcal{C}' = \langle \emptyset; S'; D'; \theta' \rangle \in \mathcal{A}_{\mathfrak{a}\text{-}AUNIF}(\mathcal{C})$ and $\tau' \in \Psi(D', S')$ without applications of any of these rules such that $label(\mathtt{a})\theta_f\tau \simeq_{\mathfrak{a}} label(\mathtt{a})\theta'\tau'$ for all $\mathtt{a} \in A$.*

Proof: We use induction over the number $n \geq 0$ of applications of both expansion rules in the derivation $\mathcal{C} \Longrightarrow^* \mathcal{C}_f$.

Base Case

If there is no application of rule (ExpB), then the lemma holds vacuously.

Induction Step

We assume that the lemma holds for n applications of rule (ExpB) and we prove it when we have $n+1$ applications. By Lemma 2.3.5, it is possible to reach the first application of the rule (ExpB) and we have the following derivation:

$$\mathcal{C} \Longrightarrow^* \langle \{\varepsilon_f \triangleq_x \varepsilon_f\} \cup A''; S''; D''; \theta'' \rangle \overset{ExpB}{\Longrightarrow}$$
$$\langle A''; S''; \{\varepsilon_f \triangleq_x \varepsilon_f\} \cup D''; \theta'' \rangle \Longrightarrow^* \mathcal{C}_f$$

We analyze the following two cases base on the set $\uparrow_x(D_f, S_f) = \uparrow(\varepsilon_f, \sigma^l_{S_f}) \cap \uparrow(\varepsilon_f, \sigma^r_{S_f})$:

- If $\uparrow_x(D_f, S_f) = \{\varepsilon_f\}$, then for all $\tau \in \Psi(D_f, S_f)$, $\{x \mapsto \varepsilon_f\} \subseteq \tau$. Thus $x\theta_f\tau = \varepsilon_f$. Consider the configuration $\mathcal{C}'' \in \mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}}}(\mathcal{C})$ where instead of this application of (ExpB) we apply (Dec) in the derivation. We have that $x\theta_f\tau \approx_{\mathfrak{a}} \varepsilon_f = x\theta'\tau'$ and this implies $\mathit{label}(\mathtt{a})\theta_f\tau \approx_{\mathfrak{a}} \mathit{label}(\mathtt{a})\theta'\tau'$ for all $\mathtt{a} \in A$. In the derivation $\mathcal{C} \Longrightarrow^* \mathcal{C}'' \Longrightarrow^* \mathcal{C}_f$, there are n applications of the expansion rules. Hence, by the induction hypothesis, the lemma is established.

- If $\uparrow_x(D_f, S_f) = \{\varepsilon_f\} \cup U$, where U is a non-empty set, then by definition of the abstraction substitutions, U is a set of collapsible terms with at least two distinct variables, say x_1 and x_2, that collapse the term and such that $x_1, x_2 \in \mathit{labels}(S_f)$. Now, if $x\tau \in U$, by the definition of configuration and item 1 of Lemma 2.3.1, it follows that $x \in \mathit{var}(y\theta'')$ for some $y \in \mathit{labels}(A)$. Since there does not exist a rule that affects the delayed set, we further obtain $x \in \mathit{var}(y\theta_f)$, which implies that $x_1, x_2 \in \mathit{var}(y\theta_f\tau)$. Moreover, by the definition of configuration and item 1 of Lemma 2.3.1, for $x_1 \in \mathit{labels}(S_f)$ there exists $w \in \mathit{labels}(A)$ such that $x_1 \in \mathit{var}(w\theta_f\tau)$. If $w = y$, then x_1 occurs twice in $y\theta_f\tau$. Otherwise, it holds that $x_1 \in \mathit{var}(w\theta_f\tau) \cap \mathit{var}(y\theta_f\tau)$. Therefore, we conclude that $(\theta_f\tau)|_{\mathit{labels}(A)}$ is not a linear substitution of $\mathcal{C}$. □

Lemma 4.1.13 (Preservation of Linear Configurations)**.** *Let $\mathcal{C} = \langle A; S; D; \theta \rangle$ be a linear configuration, any configuration $\mathcal{C}' = \langle A'; S'; D'; \theta' \rangle$ derived from $\mathcal{C}$ by $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^\ell}}$ is also a linear configuration.*

Proof: We analyze each rule application in the derivation $\mathcal{C} \Longrightarrow \mathcal{C}'$:

- Rule (Dec) introduces at most n fresh variables ($n \geq 0$), say $y_1, \ldots, y_n$, in $\mathcal{V}ran(\theta')$ and $\theta' = \theta\{\mathit{label}(\mathtt{a}) \mapsto t\}$, where $t = \mathit{head}(\mathit{lhs}(\mathtt{a}))(y_1, \ldots, y_m)$ with $\mathit{head}(\mathit{lhs}(\mathtt{a})) = \mathit{head}(\mathit{rhs}(\mathtt{a}))$ for some $\mathtt{a} \in A$. Note that each y_i occurs once in t and for all $t' \in \mathit{ran}(\theta') - \{t\}$, we have two cases to analyze:

  1. If $t' \in \mathit{ran}(\theta)$ the lemma holds trivially.

2. If $t' \notin ran(\theta)$, i.e., $t' = t''\{label(\mathtt{a}) \mapsto t\}$ for some $t'' \in ran(\theta)$ and $label(\mathtt{a}) \in var(t'')$. Observe that any $x \in var(t') = (var(t'') - \{label(\mathtt{a})\}) \uplus var(t)$ occurs once in t' and the lemma holds.

- Rules (ExpLA1), (ExpLA2), (ExpRA1) and (ExpRA2) introduce two fresh variables, say $y_1$and y_2, in $\mathcal{V}ran(\theta')$ and $\theta' = \theta\{label(\mathtt{a}) \mapsto t\}$ where $t = f(y_1, y_2)$ with f an $\mathfrak{a}$-symbol. Note that y_1 and y_2 occur once in t and for all term $t' \in ran(\theta') - \{t\}$ each variable in t' occurs at most once since if the term is affected by $\{label(\mathtt{a}) \mapsto t\}$ the only introduced variables are fresh. Therefore, the lemma holds for this case.
- Rule (Sol) does not change the computed substitution of $\mathcal{C}'$, then $\theta' = \theta$, and as $\mathcal{C}$ is linear then the lemma holds. □

The termination of the general algorithm $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ (Theorem 2.1.14) implies the termination for the linear variant algorithm.

Corollary 4.1.14 (Termination of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^\ell}$)**.** *$\mathcal{A}_{\mathfrak{a}\text{-}AUNIF^\ell}$ terminates for any linear configuration $\mathcal{C}$ and it outputs a finite set of linear configurations.*

As for $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$, the analysis of algorithm $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^\ell}$ assumes only configurations derived from initial configurations.

A *linear final configuration* is a linear configuration with its active set empty. The set of linear final configurations of $\mathcal{C}$ (see Definition 2.1.15) is denoted by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^\ell}(\mathcal{C})$.

Lemma 4.1.15 (Auxiliary Soundness of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^\ell}$)**.** *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration and $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-}AUNIF^\ell}(\mathcal{C})$. Then $label(\mathtt{a})\theta_f \in \mathcal{G}^\ell_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$, for all $\mathtt{a} \in A$.*

Proof: We proceed by induction over the derivation length.

Base case. If the derivation has length 0, $\mathcal{C}$ is a final configuration, and the lemma holds vacuously.

Inductive Step. Now, consider a derivation having the following form:

$$\langle A; S; D; \theta\rangle \Longrightarrow \langle A'; S'; D'; \theta'\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle \tag{4.1}$$

For $n \geq 0$. We assume for the induction hypothesis that for derivations of the form

$$\langle A'; S'; D'; \theta'\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle,$$

the lemma holds and show that the theorem holds for derivations of the form presented in Derivation 4.1. We continue the proof considering the various options for the transition from $\langle A; S; D; \theta\rangle$ to $\langle A'; S'; D'; \theta'\rangle$.

1. **(Dec)**. Assume that the derivation is of the form:

$$\langle \{g(s_1, \ldots, s_m) \triangleq_y g(t_1, \ldots, t_m)\} \cup A_1; S; D; \theta\rangle \overset{Dec}{\Longrightarrow}$$

$$\langle \{s_1 \triangleq_{x_1} t_1, \ldots, s_m \triangleq_{x_m} t_m\} \cup A_1; S; D; \theta' \rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f \rangle$$

where $\theta' = \theta\{y \mapsto g(x_1, \ldots, x_m)\}$. Observe that by the induction hypothesis, for all $\mathtt{a} \in A_1$ we get that $label(\mathtt{a})\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$ and for all $1 \leq i \leq m$, $x_i\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}}(s_i, t_i)$. From Lemma 2.3.1, $y\theta_f = y(\theta\{y \mapsto g(x_1, \ldots, x_m)\})\theta_f$ and from definition of configuration $y\theta = y$. Then $y\theta_f = g(x_1, \ldots, x_m)\theta_f = g(x_1\theta_f, \ldots, x_m\theta_f)$, implying that $y\theta_f \in \mathcal{G}_{\mathfrak{a}}(g(s_1, \ldots, s_m), g(t_1, \ldots, t_m))$. In addition, from Lemma 4.1.13 we have that θ_f is a linear substitution, which implies that $y\theta_f$ is a linear term. Therefore, it holds that $y\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}}(g(s_1, \ldots, s_m), g(t_1, \ldots, t_m))$.

2. **(Sol)**. Assume that the derivation is of the form:

$$\langle \{s \triangleq_y t\} \cup A_1; S; D; \theta \rangle \overset{Sol}{\Longrightarrow} \langle A_1; S'; D; \theta \rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f \rangle,$$

where $S' = \{s \triangleq_y t\} \cup S$. By induction hypothesis, for all $\mathtt{a} \in A_1$ we have that $label(\mathtt{a})\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$. Moreover, notice that $y\theta = y$ and this variable is a generalization of $s \triangleq_y t$, since this AUT is solved. Hence, we get that $y\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}}(s, t)$.

3. **(ExpLA1)**. Assume that the derivation is of the form:

$$\langle \{\varepsilon_f \triangleq_y f(s, t)\} \cup A_1; S; D; \theta \rangle \overset{ExpLA1}{\Longrightarrow}$$
$$\langle \{\varepsilon_f \triangleq_{x_1} s\} \cup A_1; S; D'; \theta' \rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f \rangle$$

where $D' = \{\star \triangleq_{x_2} t\} \cup D$ and $\theta' = \theta\{y \mapsto f(x_1, x_2)\}$. By the induction hypothesis, all the AUTs in $\{\varepsilon_f \triangleq_{x_1} s\} \cup A_1$ are generalized by θ_f and these generalizations are linear, in particular, $x_1\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\varepsilon_f, s)$. Furthermore, since $x_2 \in labels(D_f)$ then $x_2\theta_f = x_2$ and $x_2 \preceq_{\mathfrak{a}} t$. From Lemma 2.3.1 we get that $y\theta_f = f(x_1\theta_f, x_2\theta_f)$. Observe that $f(x_1\theta_f, x_2\theta_f) = f(x_1\theta_f, x_2) \in \mathcal{G}_{\mathfrak{a}}(f(\varepsilon_f, t), f(s, t))$ and since $f(\varepsilon_f, t) \approx_{\mathfrak{a}} \varepsilon_f$ and θ_f is linear substitution, we get that $y\theta_f$ belongs to $\mathcal{G}^{\ell}_{\mathfrak{a}}(\varepsilon_f, f(s, t))$.

4. The analysis of the other lateral expansion rules is analogous to the previous one.

□

Lemma 4.1.16. *Let $\mathcal{C} = \langle A; S; D; \theta \rangle$ be a linear configuration and a final linear configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-}AUNIF^{\ell}}(\mathcal{C})$, then for all different $\mathtt{a}, \mathtt{a}' \in A$ we have that $var(label(\mathtt{a})\theta_f) \cap var(label(\mathtt{a}')\theta_f) = \emptyset$.*

Proof: We use induction over length derivation.

Base case

If the derivation has length 0, $\mathcal{C}$ is a final configuration, and the lemma holds vacuously.

Induction Step

We assume that the lemma holds for derivations of length less than or equal to $n \geq 0$, and we prove it for derivations with length $n + 1$. Consider the derivation $\mathcal{C} \Longrightarrow \mathcal{C}' \Longrightarrow^n \mathcal{C}_f$, where $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle$. We analyze each rule application case to obtain $\mathcal{C}'$:

1. **(Dec)**. Assume that the derivation is of the form:
$$\langle \{g(s_1,\ldots,s_m) \triangleq_x g(t_1,\ldots,t_m)\} \cup A_1; S; D; \theta\rangle \overset{Dec}{\Longrightarrow} \langle A'; S; D; \theta\{x \mapsto g(x_1,\ldots,x_m)\}\rangle$$
where $A' = \{s_1 \triangleq_{x_1} t_1,\ldots,s_m \triangleq_{x_m} t_m\} \cup A_1$. By the induction hypothesis, for all different $\mathtt{a},\mathtt{a}' \in A'$ the lemma holds, in particular for all $\mathtt{a}'' \in A'$ with $\mathtt{a}'' \neq x_i$, $var(label(\mathtt{a}'')\theta_f) \cap var(x_i\theta_f) = \emptyset$, for $1 \leq i \leq m$. It remains to prove that for all $\mathtt{a}'' \in A_1$, $var(label(\mathtt{a}'')\theta_f) \cap var(x\theta_f) = \emptyset$. Observe that $x\theta_f = g(x_1\theta_f,\ldots,x_m\theta_f)$, then we have that $var(x\theta_f) = \uplus_{i=1}^{m} var(x_i\theta_f)$ which implies
$$var(label(\mathtt{a}'')\theta_f) \cap var(x\theta_f) = \biguplus_{i=1}^{m}(var(label(\mathtt{a}'')\theta_f) \cap var(x_i\theta_f)) = \biguplus_{i=1}^{m} \emptyset = \emptyset$$
Therefore, the lemma holds for this case.

2. **(Sol)**. Assume that the derivation is of the form:
$$\langle \{s \triangleq_x t\} \cup A'; S; D; \theta\rangle \overset{Sol}{\Longrightarrow} \langle A'; \{s \triangleq_x t\} \cup S; D; \theta\rangle,$$
By induction hypothesis for all different $\mathtt{a},\mathtt{a}' \in A'$ we have that $var(label(\mathtt{a})\theta_f) \cap var(label(\mathtt{a}')\theta_f) = \emptyset$. Observe that $x\theta_f = x$ since $x \in labels(S_f)$. Moreover, as after the application of the rule (Sol) all the introduced variables in the terms in the range related with any label of A' are fresh, then $x \notin var(label(\mathtt{a}')\theta_f)$ for all $\mathtt{a}' \in A'$ and the lemma holds.

3. **(ExpLA1)**. Assume that the derivation is of the form:
$$\langle \{\varepsilon_f \triangleq_x f(s,t)\} \cup A_1; S; D; \theta\rangle \overset{ExpLA1}{\Longrightarrow} \langle A'; S; \{\star \triangleq_{x_2} t\} \cup D; \theta\{x \mapsto f(x_1,x_2)\}\rangle$$
where $A' = \{\varepsilon_f \triangleq_{x_1} s\} \cup A_1$. By induction hypothesis for all different $\mathtt{a},\mathtt{a}' \in A'$ we have that $var(label(\mathtt{a})\theta_f) \cap var(label(\mathtt{a}')\theta_f) = \emptyset$. Notice that $x\theta_f = f(x_1\theta_f, x_2\theta_f)$ and $x_2\theta_f = x_2$ since $x_2 \notin dom(\theta_f)$, implying that $var(x\theta_f) = var(x_1\theta_f) \uplus \{x_2\}$. Additionally, $x_2 \notin var(label(\mathtt{a})\theta_f)$ for all $\mathtt{a} \in A_1$, as all the introduced variables in the range of θ_f after the rule (ExpLA1) application are fresh. Hence for all $\mathtt{a} \in A_1$ we have that $var(label(\mathtt{a})\theta_f) \cap var(x\theta_f) = \emptyset$ and the lemma holds.

4. The analysis of the other lateral expansion rules is analogous to the previous one.

□

Theorem 4.1.17 (Soundness of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^\ell}$)**.** *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration, and $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-}AUNIF^\ell}(\mathcal{C})$. Then we have that $\theta_f|_{labels(A)} \in \mathcal{G}^\ell_{\mathfrak{a}}(\mathcal{C})$.*

PROOF: By Lemma 4.1.16 we know that $var(label(\mathtt{a})\gamma) \cap var(label(\mathtt{a}')\gamma) = \emptyset$, for all distinct AUTs $\mathtt{a},\mathtt{a}' \in A$. Moreover, by Lemma 4.1.15 for all $\mathtt{a} \in A$, we have that

$label(\mathtt{a})\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$. Finally, by applying Corollary 4.1.7, we conclude that the restriction $\theta_f|_{labels(A)} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C})$. □

As we need to avoid the duplication of variables in the generalizations, the abstraction set of Definition 2.2.1 and the abstraction substitutions of Definition 2.2.19 need to be replaced with the following definition.

Definition 4.1.18 (Linear Abstraction Substitution)**.** Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration, the *linear abstraction substitution* μ_D of $\mathcal{C}$, for short we denote it as `las`, with $dom(\mu_D) = labels(D)$ is defined as $\mu_D = \{x \mapsto s \mid \star \triangleq_x s \in D \text{ or } s \triangleq_x \star \in D\}$ when $D \neq \emptyset$ and $\mu_D = id$, otherwise.

Definition 4.1.19 (Algorithm $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}_{\mu}}$)**.** The algorithm $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}_{\mu}}$ takes as input a set of valid AUTs A and performs the next steps:

1. Associate the respective initial configuration $\mathcal{C}$ with active set A and computed substitution ι and apply $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$ to $\mathcal{C}$ generating the set $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$.
2. Build the `las` μ_{D_f} for each $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$.
3. Apply the substitution $\theta_f\mu_{D_f}$ to $label(\mathtt{a})$ for each $\mathtt{a} \in A$ and any final linear configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$.

It is easy to notice that $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}_{\mu}}$ is terminating since $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$ is terminating, and from the definition of the `las` μ, for each final configuration, there exists a unique μ.

Lemma 4.1.20. *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration, a final linear configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$, and μ_{D_f} the* `las` *of $\mathcal{C}_f$. Then for all different $\mathtt{a}, \mathtt{a}' \in A$ we have that $var(label(\mathtt{a})\theta_f\mu_{D_f}) \cap var(label(\mathtt{a}')\theta_f\mu_{D_f}) = \emptyset$.*

Proof: By Lemma 4.1.16 for all different $\mathtt{a}, \mathtt{a}' \in A$ we have that $var(label(\mathtt{a})\theta_f)$ and $var(label(\mathtt{a}')\theta_f)$ are disjoint sets. In addition, since the terms in the range of μ_{D_f} are ground, $\mathcal{V}ran(\mu_{D_f}) = \emptyset$, then $var(label(\mathtt{a})\theta_f\mu_{D_f}) \cap var(label(\mathtt{a}')\theta_f\mu_{D_f}) = \emptyset$. □

Lemma 4.1.21 (Auxiliary Soundness of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}_{\mu}}$)**.** *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration, $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$, and μ_{D_f} the* `las` *of $\mathcal{C}_f$. Then for all $\mathtt{a} \in A$, $label(\mathtt{a})\theta_f\mu_{D_f} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$.*

Proof: We proceed by induction over the derivation length.

Base case. If the derivation has length 0, $\mathcal{C}$ is a final configuration, and the lemma holds vacuously.

Inductive Step. Now, consider a derivation having the following form:

$$\langle A; S; D; \theta\rangle \Longrightarrow \langle A'; S'; D'; \theta'\rangle \Longrightarrow^{n} \langle \emptyset; S_f; D_f; \theta_f\rangle \tag{4.2}$$

For $n \geq 0$. We assume for the induction hypothesis that for derivations of the form

$$\langle A'; S'; D'; \theta'\rangle \Longrightarrow^{n} \langle \emptyset; S_f; D_f; \theta_f\rangle,$$

the theorem holds and show that the theorem holds for derivations of the form presented in Derivation 4.2. We continue the proof considering the various options for the transition from $\langle A; S; D; \theta\rangle$ to $\langle A'; S'; D'; \theta'\rangle$.

1. **(Dec)**. Assume that the derivation is of the form:

$$\langle \{g(s_1, \ldots, s_m) \triangleq_y g(t_1, \ldots, t_m)\} \cup A_1; S; D; \theta\rangle \overset{Dec}{\Longrightarrow}$$
$$\langle \{s_1 \triangleq_{x_1} t_1, \ldots, s_m \triangleq_{x_m} t_m\} \cup A_1; S; D; \theta'\rangle \Longrightarrow^{n} \langle \emptyset; S_f; D_f; \theta_f\rangle$$

   where $\theta' = \theta\{y \mapsto g(x_1, \ldots, x_m)\}$. By the induction hypothesis, for all $\mathtt{a} \in A_1$ we get that $label(\mathtt{a})\theta_f\mu_{D_f} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$ and $x_i\theta_f\mu_{D_f} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(s_i, t_i)$, for all $1 \leq i \leq m$. From Lemma 2.3.1, $y\theta_f\mu_{D_f} = g(x_1\theta_f\mu_{D_f}, \ldots, x_m\theta_f\mu_{D_f})$, implying that $y\theta_f\mu_{D_f} \in \mathcal{G}_{\mathfrak{a}}(g(s_1, \ldots, s_m), g(t_1, \ldots, t_m))$. In addition, from Lemma 4.1.20 we have that $\theta_f\mu_{D_f}$ is a linear substitution, then $y\theta_f\mu_{D_f}$ is a linear term. Therefore, we have that $y\theta_f\mu_{D_f} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(g(s_1, \ldots, s_m), g(t_1, \ldots, t_m))$.

2. **(Sol)**. Assume that the derivation is of the form:

$$\langle \{s \triangleq_y t\} \cup A_1; S; D; \theta\rangle \overset{Sol}{\Longrightarrow} \langle A_1; S'; D; \theta\rangle \Longrightarrow^{n} \langle \emptyset; S_f; D_f; \theta_f\rangle,$$

   where $S' = \{s \triangleq_y t\} \cup S$. By induction hypothesis, for all $\mathtt{a} \in A_1$ we have that $label(\mathtt{a})\theta_f\mu_{D_f} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$. Notice that $y\theta\mu_{D_f} = y$ and is a generalization of $s \triangleq_y t$ since this AUT is solved. Hence, we get that $y\theta_f\mu_{D_f} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(s, t)$.

3. **(ExpLA1)**. Assume that the derivation is of the form:

$$\langle \{\varepsilon_f \triangleq_y f(s, t)\} \cup A_1; S; D; \theta\rangle \overset{ExpLA1}{\Longrightarrow}$$
$$\langle \{\varepsilon_f \triangleq_{x_1} s\} \cup A_1; S; D'; \theta'\rangle \Longrightarrow^{n} \langle \emptyset; S_f; D_f; \theta_f\rangle$$

   where $D' = \{\star \triangleq_{x_2} t\} \cup D$ and $\theta' = \theta\{y \mapsto f(x_1, x_2)\}$. By the induction hypothesis, all the AUTs in $\{\varepsilon_f \triangleq_{x_1} s\} \cup A_1$ are generalized by $\theta_f\mu_{D_f}$, in particular, $x_1\theta_f\mu_{D_f} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\varepsilon_f, s)$. Furthermore, since $x_2 \in labels(D_f)$ then $x_2\theta_f = x_2$ and $x_2\mu_{D_f} = t$. From Lemma 2.3.1 we get that $y\theta_f\mu_{D_f} = f(x_1\theta_f\mu_{D_f}, x_2\theta_f\mu_{D_f})$. Observe that $f(x_1\theta_f\mu_{D_f}, x_2\theta_f\mu_{D_f}) \in \mathcal{G}_{\mathfrak{a}}(f(\varepsilon_f, t), f(s, t))$ and since $f(\varepsilon_f, t) \approx_{\mathfrak{a}} \varepsilon_f$ and $\theta_f\mu_{D_f}$ is linear substitution, we get that $y\theta_f\mu_{D_f} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\varepsilon_f, f(s, t))$.

4. The analysis of the remaining lateral expansion rules is analogous. □

Theorem 4.1.22 (Soundness of $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^{\ell}_{\mu}}}$)**.** *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration, $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-}AU_{NIF^{\ell}}}(\mathcal{C})$, and μ_{D_f} the* `las` *of $\mathcal{C}_f$. Then we have that the substitution $(\theta_f\mu_{D_f})|_{labels(A)} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C})$.*

PROOF: By Lemma 4.1.20 then it holds that $var(label(\mathtt{a})\theta_f\mu_{D_f}) \cap var(label(\mathtt{a}')\theta_f\mu_{D_f}) = \emptyset$, for all different $\mathtt{a}, \mathtt{a}' \in A$. Moreover, by Lemma 4.1.21, we get that for all $\mathtt{a} \in A$, $label(\mathtt{a})\theta_f\mu_{D_f} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$. Therefore, it follows that $(\theta_f\mu_{D_f})|_{labels(A)} \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C})$. □

Notice that for any linear configuration $\mathcal{C} = \langle A; S; D; \theta\rangle$ and any final linear configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$, there exits a unique μ_{D_f}. Then for each AUT $\mathtt{a} \in A$ the term $label(\mathtt{a})\theta_f\mu_{D_f}$ is a *computed linear* $\mathfrak{a}$*-generalization* by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}_{\mu}}$. Indeed, from the Lemmas 4.1.21 and 4.1.15, the term $label(\mathtt{a})\theta_f\mu_{D_f}$ is a linear generalization of the terms $lhs(\mathtt{a})$ and $rhs(\mathtt{a})$.

Example 4.1.23. To compute linear $\mathfrak{a}$-generalizations for $g(\varepsilon_f, a)$ and $g(f(h(\varepsilon_f), a), \varepsilon_f)$ with $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}_{\mu}}$, we follow the same derivations described in Example 2.1.16 without applying the (Mer) and (ExpB) rules. Then, instead of computing the set of abstraction substitutions, we use the `las` μ for each final configuration. From final configurations computed in the example:

$$\langle \emptyset; \{\varepsilon_f \triangleq_{y_1} h(\varepsilon_f), a \triangleq_{w_2} \varepsilon_f\}; D_1; \theta_1\rangle,$$
$$\langle \emptyset; \{\varepsilon_f \triangleq_{y_4} a, a \triangleq_{w_2} \varepsilon_f\}; D_2; \theta_2\rangle$$

where $D_1 = \{\star \triangleq_{y_2} a\}$, $D_2 = \{\star \triangleq_{y_3} h(\varepsilon_f)\}$, $\theta_1 = \{x \mapsto g(f(y_1, y_2), w_2), w_1 \mapsto f(y_1, y_2)\}$, and $\theta_2 = \{x \mapsto g(f(y_3, y_4), w_2), w_1 \mapsto f(y_3, y_4)\}$, we obtain the `las`'s $\mu_{D_1} = \{y_2 \mapsto a\}$ and $\mu_{D_2} = \{y_3 \mapsto h(\varepsilon_f)\}$ respectively. Each final configuration produces a single linear $\mathfrak{a}$-generalization:

$$x\theta_1\mu_{D_1} = g(f(y_1, a), w_2) \qquad \text{(from derivation 1)}$$
$$x\theta_2\mu_{D_2} = g(f(h(\varepsilon_f), y_4), w_2) \qquad \text{(from derivation 2)}$$

Notice that the terms above are the computed linear $\mathfrak{a}$-generalizations of $g(\varepsilon_f, a)$ and $g(f(h(\varepsilon_f), a), \varepsilon_f)$ generated by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}_{\mu}}$. ◇

4.2 The Minimal Complete Set of Linear $\mathfrak{a}$-Generalizations

To show that the set of all computed linear $\mathfrak{a}$-generalizations generated by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}_{\mu}}$ is the minimal complete set of linear generalizations, we must first prove that it captures a set of generalizations containing linear $\mathfrak{a}$-generalizations more specific than any linear $\mathfrak{a}$-generalization of any pair of terms.

Lemma 4.2.1. *Let A be a valid set of AUTs and $\gamma \in \mathcal{G}^{\ell}_{\mathfrak{a}}(A)$. Then, for all $x \in labels(A)$ it holds that $x\gamma \preceq_{\mathfrak{a}} x\theta_f\mu_{D_f}$ if and only if $(\gamma \preceq_{\mathfrak{a}} \theta_f\mu_{D_f})|_{labels(A)}$.*

PROOF: Assume first that for all $x \in \mathit{labels}(A)$ we have $x\gamma \preceq_{\mathfrak{a}} x\theta_f\mu_{D_f}$. Then, for each x, there exists δ_x with $x\gamma\delta_x \approx_{\mathfrak{a}} x\theta_f\mu_{D_f}$. Let $\mathit{labels}(A) = \{x_1, \ldots, x_n\}$. Since γ is linear, the sets $\mathit{var}(x_i\gamma)$ are pairwise disjoint, hence $\delta = \delta_{x_1} \cdots \delta_{x_n}$ is well-defined. Thus, $x\gamma\delta \approx_{\mathfrak{a}} x\theta_f\mu_{D_f}$ for all $x \in \mathit{labels}(A)$, implying that $(\gamma \preceq_{\mathfrak{a}} \theta_f\mu_{D_f})|_{\mathit{labels}(A)}$.
Conversely, if $(\gamma \preceq_{\mathfrak{a}} \theta_f\mu_{D_f})|_{\mathit{labels}(A)}$, then some substitution δ satisfies $x\gamma\delta \approx_{\mathfrak{a}} x\theta_f\mu_{D_f}$ for all $x \in \mathit{labels}(A)$, i.e. $x\gamma \preceq_{\mathfrak{a}} x\theta_f\mu_{D_f}$. □

Theorem 4.2.2 (Completeness of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}_{\mu}}$)**.** *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration and $\gamma \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C})$. Then, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$ and μ_{D_f} the* las *of $\mathcal{C}_f$ such that $(\gamma \preceq_{\mathfrak{a}} \theta_f\mu_{D_f})|_{\mathit{labels}(A)}$.*

PROOF: We use induction on $\mathit{size}(A)$.

Base Case

If $\mathit{size}(A) = 0$, $\mathcal{C}$ is a final linear configuration, and the theorem holds vacuously.

Induction Step

The statement is assumed for configurations with $\mathit{size}(A) < n$, $n > 2$, and we show that it holds for configurations with $\mathit{size}(A) = n$. The proof is obtained through case analysis over the term structure of the terms s and t of an AUT $s \triangleq_x t \in A$. Let $\mathcal{C} = \langle\{s \triangleq_x t\} \cup A_1; S; D; \theta\rangle \Longrightarrow \mathcal{C}'$ be a derivation after an application of any rule over $s \triangleq_x t$. Without loss of generality, we assume that $\gamma \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C})$ is such that variables in the range of γ are different from the variables used by the algorithm. Also, let ρ_l and ρ_r be the associated substitutions of γ as a linear substitution generalization of A. We analyze each case of transition from $\mathcal{C}$ to $\mathcal{C}'$. For each case, Lemma 4.2.1 is used to establish that $x\gamma \preceq_{\mathfrak{a}} x\theta_f\mu_{D_f}$ holds for all $x \in \mathit{labels}(A)$, in place of verifying $(\gamma \preceq_{\mathfrak{a}} \theta_f\mu_{D_f})|_{\mathit{labels}(A)}$.

1. If $\mathit{head}(s) = \mathit{head}(t)$, then we have two subcases to consider and are covered by the rule (Dec).

   (a) If $\mathit{head}(s) = g \neq \varepsilon_f$. Assume that the derivation is of the form:
   $$\langle\{g(s_1, \ldots, s_m) \triangleq_x g(t_1, \ldots, t_m)\} \cup A_1; S; D; \theta\rangle \overset{Dec}{\Longrightarrow} \langle A'; S; D; \theta\{x \mapsto g(x_1, \ldots, x_m)\}\rangle,$$
   where $A' = \{s_1 \triangleq_{x_1} t_1, \ldots, s_m \triangleq_{x_m} t_m\} \cup A_1$ with $m \geq 0$. We have two cases: either $x\gamma$ is headed by the symbol g or is a variable. In both cases, we will build a linear substitution generalization γ' for $\mathcal{C}'$, then apply the induction hypothesis to $\mathcal{C}'$ and γ', and infer that γ is more general than the substitution computed by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}_{\mu}}$ restricted to the labels of A.

      i. If $x\gamma$ is headed by the symbol g, say $x\gamma = g(r_1, \ldots, r_m)$, we can define γ' for $\mathcal{C}''$ as
      $$\gamma'(y) = \begin{cases} \gamma(y) & \text{if } y \in \mathit{labels}(A_1), \\ r_i & \text{if } y = x_i, 1 \leq i \leq m. \end{cases}$$

Notice that $\gamma' \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C}')$. Indeed, $dom(\gamma')$ is the set of active labels in $\mathcal{C}'$. Also, if $y \in labels(A_1)$ is the label of an AUT $\mathtt{a}$, $y\gamma'\rho_l = y\gamma\rho_l \approx_{\mathfrak{a}} lhs(\mathtt{a})$. Similarly, for $\gamma'\rho_r$. Otherwise, $x_i\gamma'\rho_l = r_i\rho_l = (x\gamma\rho_l)|_i \approx_{\mathfrak{a}} s_i$, which is assured since we are working with normal terms. Similarly, for $\gamma'\rho_r$. Additionally, since $\gamma \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C})$, we have that $var(x\gamma) \cap var(label(\mathtt{a})\gamma') = \emptyset$ for all $\mathtt{a} \in A_1$ and as $x\gamma = g(r_1, \ldots, r_m)$ then $var(r_i\gamma') \cap var(label(\mathtt{a})\gamma') = \emptyset$ for all $\mathtt{a} \in A_1$. Therefore $\gamma' \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C}')$.

ii. If $x\gamma$ is a variable, say z, we can define γ' as

$$\gamma'(y) = \begin{cases} \gamma(y) & \text{if } y \in labels(A_1), \\ z_i & \text{if } y = x_i, \text{ where } z_i \text{ fresh variable}, 1 \leq i \leq m. \end{cases}$$

Also, since $x\gamma\rho_l = z\rho_l \approx_{\mathfrak{a}} g(u_1, \ldots, u_m)$, $(z\rho_l)|_i \approx_{\mathfrak{a}} s_i$. Thus, we can define ρ'_l as $\rho_l|_{(dom(\rho_l)-\{z\})}$ extending its domain with the set of fresh variables $\{z_1, \ldots, z_m\}$ as

$$\rho'_l(y) = \begin{cases} \rho_l(y) & \text{if } y \in dom(\rho_l) - \{z\} \\ (z\rho_l)|_i & \text{for } z_i, 1 \leq i \leq m. \end{cases}$$

Similarly, for ρ'_r. Observe that γ' is a linear substitution since γ' is based on γ and the introduced variables are fresh. Then, we can conclude that $\gamma' \in \mathcal{G}_{\mathfrak{a}}(\mathcal{C}')$; since for all $\mathtt{a} \in A'$, $labels(\mathtt{a})\gamma'\rho'_l \approx_{\mathfrak{a}} lhs(\mathtt{a})$. In particular, for AUTs of the form $s_i \triangleq_{x_i} t_i$ we have that $x_i\gamma'\rho'_l = z_i\rho'_l = (z\rho_l)|_i \approx_{\mathfrak{a}} s_i$. Similarly, for ρ'_r. Moreover, for $1 \leq i \leq m$, since z_i are new variables, then $var(x_i\gamma') \cap var(label(\mathtt{a})\gamma) = \emptyset$ for all $\mathtt{a} \in A_1$. Therefore $\gamma' \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C}')$.

Then, by the induction hypothesis, we have that there exist a final linear configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^{\ell}}}(\mathcal{C}') \subseteq \mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^{\ell}}}(\mathcal{C})$ and the $\mathtt{las}$ μ_{D_f} such that for all $y \in labels(A')$ we have that $y\gamma' \preceq_{\mathfrak{a}} y\theta_f\mu_{D_f}$.

To conclude, in the former case, if $x\gamma$ is headed by the symbol g, observe that $x\gamma = g(x_1\gamma', \ldots, x_m\gamma') \preceq_{\mathfrak{a}} g(x_1\theta_f\mu_{D_f}, \ldots, x_m\theta_f\mu_{D_f}) = x\theta_f\mu_{D_f}$. In the latter case, if $x\gamma = z$ then $x\gamma \preceq_{\mathfrak{a}} x\theta_f\mu_{D_f}$. For all other labels, $y \in labels(A_1)$, $y\gamma = y\gamma' \preceq_{\mathfrak{a}} y\theta_f\mu_{D_f}$, then $\mathcal{C}_f$ is the desired final configuration.

(b) If $head(s) = \varepsilon_f$. Assume that the derivation is of the form:

$$\langle \{\varepsilon_f \triangleq_x \varepsilon_f\} \cup A'; S; D; \theta \rangle \overset{Dec}{\Longrightarrow} \langle A'; S; D; \theta\{x \mapsto \varepsilon_f\} \rangle$$

We have three subcases to consider: either $x\gamma$ is headed by the symbol f, is ε_f, or is a variable. In all cases, we build a linear substitution generalization γ' for $\mathcal{C}'$, then apply the induction hypothesis to $\mathcal{C}'$ and γ', and we infer that γ is more general than the computed generalization by $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^{\ell}_{\mu}}}$.

i. If $x\gamma$ is headed by the symbol f, say $x\gamma = f(r_1, r_2)$, we can consider $\gamma' = \gamma|_{labels(A')}$. Observe that this is the desired substitution since for all

$\mathtt{a} \in A'$, $label(\mathtt{a})\gamma'\rho_l = label(\mathtt{a})\gamma\rho_l \approx_{\mathfrak{a}} lhs(\mathtt{a})$. Similarly, for ρ_r and $rhs(\mathtt{a})$. By induction hypothesis, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^\ell}(\mathcal{C})$ and μ_{D_f} the abstraction substitution of $\mathcal{C}'$ such that $y \in labels(A')$, we have that $y\gamma' \preceq_{\mathfrak{a}} y\theta_f\mu_{D_f}$. It remains to prove it for $x\gamma = f(r_1, r_2)$. Notice that $x\theta_f\mu_{D_f} = \varepsilon_f$, and as $f(r_1, r_2)$ is a linear $\mathfrak{a}$-generalization of $\varepsilon_f \triangleq_x \varepsilon_f$ then $f(r_1, r_2)$ is collapsible, i.e., there exists at least $z \in var(f(r_1, r_2))$ that collapse the term. All of these variables are different and occur at most once in the term. Then without loss of generality consider $\sigma = \{z \mapsto \varepsilon_f\}$, we get that $x\gamma\sigma \approx_{\mathfrak{a}} \varepsilon_f = x\theta_f\mu_{D_f}$. It is important to note that this instantiation is possible because each variable in $f(r_1, r_2)$ is unique and distinct from the variables in the range of γ. Hence $x\gamma \preceq_{\mathfrak{a}} x\theta_f\mu_{D_f}$, and the theorem for this subcase holds.

   ii. If $x\gamma = \varepsilon_f$. We can consider the same γ' of case (i) since we have the same A'. We need to prove that $x\gamma \preceq_{\mathfrak{a}} x\theta_f\mu_{D_f}$. Indeed, we have that $x\gamma = \varepsilon_f \preceq_{\mathfrak{a}} \varepsilon_f = x\theta_f\mu_{D_f}$ and the lemma holds.

   iii. If $x\gamma$ is a variable, say $x\gamma = z$. This case is analogous to case $1.(a)ii$.

   In all these three subcases, it holds that for all $y \in labels(A)$, $y\gamma' \preceq_{\mathfrak{a}} y\theta_f\mu_{D_f}$.

2. If $head(s) \neq head(t)$ and they are not related absorptive symbols. We apply the rule **(Sol)**. Assume that the derivation is of the form:

$$\langle \{s \triangleq_x t\} \cup A'; S; D; \theta \rangle \overset{Sol}{\Longrightarrow} \langle A'; \{s \triangleq_x t\} \cup S; D; \theta \rangle,$$

   Let $\gamma' = \gamma|_{labels(A')}$. Then, $\gamma' \in \mathcal{G}^\ell_{\mathfrak{a}}(\mathcal{C}')$. In fact, for any $\mathtt{a} \in A'$, $label(\mathtt{a})\gamma'\rho_l = label(\mathtt{a})\gamma\rho_l \approx_{\mathfrak{a}} lhs(\mathtt{a})$. Similarly, for ρ_r and $rhs(\mathtt{a})$.

   By induction hypothesis, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^\ell}(\mathcal{C})$ and μ_{D_f} the $\mathtt{las}$ of $\mathcal{C}_f$ such that for all $y \in labels(A')$ we have $y\gamma' \preceq_{\mathfrak{a}} y\theta_f\mu_{D_f}$. It remains to prove that $x\gamma \preceq_{\mathfrak{a}} x\theta_f\mu_{D_f}$. Notice that $x\gamma$ should be a variable, say x', since it labels a solved equation, and solved equations allow only AUT with pairs of terms with different head symbols, which also cannot be related absorptive symbols. Without loss of generality, we assume that this variable does not belong to the variables in the final configuration $\mathcal{C}_f$ and then $x\gamma = x' \preceq_{\mathfrak{a}} x = x\theta_f\mu_{D_f}$, implying that the theorem holds for this case.

3. If $head(s) \neq head(t)$ with $head(s) = \varepsilon_f$ and $head(t) = f$. We apply the rules **(ExpLA1)** and **(ExpLA2)**. Assume we have the derivations of the form:

$$\langle \{\varepsilon_f \triangleq_x f(s,t)\} \cup A'; S; D; \theta \rangle$$
$$\overset{\text{(ExpLA1)}}{\swarrow} \qquad \overset{\text{(ExpLA2)}}{\searrow}$$
$$\mathcal{C}_1 = \langle A_1; S; D_1; \theta\{x \mapsto f(x_1, x_2)\} \rangle \quad \langle A_2; S; D_2; \theta\{x \mapsto f(x_3, x_4)\} \rangle = \mathcal{C}_2$$

where $A_1 = \{\varepsilon_f \triangleq_{x_1} s\} \cup A'$, $D_1 = \{\star \triangleq_{x_2} t\} \cup D$, $A_2 = \{\varepsilon_f \triangleq_{x_4} t\} \cup A'$, and $D_2 = \{\star \triangleq_{x_3} s\} \cup D$. The set $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^\ell}}(\mathcal{C}) = \mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^\ell}}(\mathcal{C}_1) \cup \mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^\ell}}(\mathcal{C}_2)$. Also, $x\gamma$ is either a term of the form $f(u, v)$ or a variable, say x'. We need to consider both cases.

(a) Case $x\gamma = f(u, v)$. Since $x\gamma$ is a generalization of ε_f, this term is collapsible, and we get that either u or v are collapsible. We analyze the three following subcases:

i. u is a collapsible subterm of $f(u, v)$ and v is not. For this case we need to consider the derivation of (ExpLA1) and define γ for $\mathcal{C}_1$ as follows

$$\gamma'(z) = \begin{cases} \gamma(z) & \text{if } z \in \mathit{labels}(A'); \\ u & \text{if } z = x_1, \end{cases}$$

Moreover, if $y \in \mathit{labels}(A_1)$ is a label of an AUT $\mathtt{a}$, $y\gamma'\rho_l = y\gamma\rho_l \approx_{\mathfrak{a}} \mathit{lhs}(\mathtt{a})$. Similarly, for $\gamma'\rho_r$. Otherwise, $x_1\gamma'\rho_l = u\rho_l = (x\gamma\rho_l)|_1 \approx_{\mathfrak{a}} \varepsilon_f$, which is assured for the assumption of this case. On the other hand, $x_1\gamma'\rho_r = u\rho_r = (x\gamma\rho_r)|_1 \approx_{\mathfrak{a}} s$. Furthermore, since $\gamma \in \mathcal{G}^\ell_{\mathfrak{a}}(\mathcal{C})$, we have that $\mathit{var}(x\gamma) \cap \mathit{var}(\mathit{label}(\mathtt{a})\gamma') = \emptyset$ for all $\mathtt{a} \in A_1$ and as $x\gamma = f(u, v)$ then we get that $\mathit{var}(x_1\gamma') \cap \mathit{var}(\mathit{label}(\mathtt{a})\gamma') = \emptyset$. Therefore $\gamma' \in \mathcal{G}^\ell_{\mathfrak{a}}(\mathcal{C}_1)$. By induction hypothesis, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^\ell}}(\mathcal{C})$ and μ_{D_f} the abstraction substitution of $\mathcal{C}_f$ such that for all $y \in \mathit{labels}(A_1)$ we have that $y\gamma' \preceq_{\mathfrak{a}} y\theta_f\mu_{D_f}$. In particular, $u = x_1\gamma' \preceq_{\mathfrak{a}} x_1\theta_f\mu_{D_f}$, and as v is a generalization of t, then it holds

$$x\gamma = f(x_1\gamma', v) \preceq_{\mathfrak{a}} f(x_1\theta_f\mu_{D_f}, t) = f(x_1\theta_f\mu_{D_f}, x_2\theta_f\mu_{D_f}) = x\theta_f\mu_{D_f}.$$

Hence for all $y \in \mathit{labels}(A)$ we have that $y\gamma \preceq_{\mathfrak{a}} y\theta_f\mu_{D_f}$.

ii. v is collapsible subterm of $f(u, v)$ and u is not. This case is analogous to the case (i) but considering the derivation of (ExpLA2).

iii. u and v are both collapsible subterms of $f(u, v)$. It is possible to choose any derivation generated by an expansion rule to get a more specific generalization generated by $\mathcal{A}_{\mathfrak{a}\text{-AU}_{\text{NIF}^\ell}}$ as in case (i).

(b) Case $x\gamma = x'$. Since $\gamma \in \mathcal{G}^\ell_{\mathfrak{a}}(\mathcal{C})$, there exist ρ_l and ρ_r such that for all AUT $\mathtt{a} \in A$ and, in particular for $\varepsilon_f \triangleq_x f(s, t)$, $x\gamma\rho_l = x'\rho_l \approx_{\mathfrak{a}} \varepsilon_f$ and $x\gamma\rho_r = x'\rho_l \approx_{\mathfrak{a}} f(s, t)$. Without loss of generality, we consider the derivation of (ExpLA1) and define γ' as

$$\gamma'(z) = \begin{cases} \gamma(z) & \text{if } z \in \mathit{labels}(A'); \\ x'_1 & \text{if } z = x_1. \end{cases}$$

To prove that $\gamma' \in \mathcal{G}^\ell_{\mathfrak{a}}(\mathcal{C}_1)$, firstly, notice that $\mathit{dom}(\gamma') = \{x_1\} \cup \mathit{labels}(A')$, the labels of the active set of $\mathcal{C}_1$. Secondly, consider ρ'_l and ρ'_r defined as below.

$$\rho'_l(z) = \begin{cases} \rho_l(z) & \text{if } z \in \mathcal{V}ran(\gamma) \\ \varepsilon_f & \text{if } z = x'_1. \end{cases}$$
$$\rho'_r(z) = \begin{cases} \rho_r(z) & \text{if } z \in \mathcal{V}ran(\gamma) \\ s & \text{if } z = x'_1. \end{cases}$$

Notice that for all $\mathtt{a} \in A'$, we have that $label(\mathtt{a})\gamma'\rho'_l = label(\mathtt{a})\gamma\rho_l \approx_{\mathfrak{a}} lhs(\mathtt{a})$ and $label(\mathtt{a})\gamma'\rho'_r = label(\mathtt{a})\gamma\rho_r \approx_{\mathfrak{a}} rhs(\mathtt{a})$. To conclude that $\gamma' \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C}_1)$, notice that $x_1\gamma'\rho'_l = x'_1\rho'_l = \varepsilon_f$, and $x_1\gamma'\rho'_r = x'_1\rho'_r = s$.

Furthermore, since $\gamma \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C})$, we have that $var(x\gamma) \cap var(label(\mathtt{a})\gamma') = \emptyset$ for all $\mathtt{a} \in A'$ and as x'_1 is a fresh variable then $var(x_1\gamma') \cap var(label(\mathtt{a})\gamma') = \emptyset$ for all $\mathtt{a} \in A'$. Therefore $\gamma' \in \mathcal{G}^{\ell}_{\mathfrak{a}}(\mathcal{C}_1)$.

By induction hypothesis, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$ and μ_{D_f} the `las` of $\mathcal{C}_f$ such that for all $y \in labels(A_1)$ we have that $y\gamma' \preceq_{\mathfrak{a}} y\theta_f\mu_{D_f}$. Additionally, $x\gamma = x' \preceq_{\mathfrak{a}} x\theta_f\mu_{D_f}$ implying that for all $y \in labels(A)$ we have that $y\gamma \preceq_{\mathfrak{a}} y\theta_f\mu_{D_f}$.

4. If $head(s) \neq head(t)$ with $head(s) = f$ and $head(t) = \varepsilon_f$, then we can apply the rules **(ExpRA1)** and **(ExpRA2)**. This case is analogous to the previous one. □

Corollary 4.2.3 (Type of Linear Absorptive Anti-Unification)**.** *The linear variant of anti-unification modulo* $\mathfrak{a}$ *is of type finitary.*

Definition 4.2.4 (Merged configurations)**.** Let $\mathcal{C}$ be a configuration. We say that the set $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$ is *merged* if for all $\langle \emptyset; S; D; \theta \rangle, \langle \emptyset; S'; D'; \theta' \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$ and $u \triangleq_y v \in S$, $u \triangleq_{y'} v \in S'$ if $y = y'$.

The Definition 4.2.4 is a tool for avoiding equivalent linear $\mathfrak{a}$-generalizations over renaming, this merging does not affect the linearity, because it is applied between different configurations.

Definition 4.2.5. Let s and t be terms, $\mathcal{C} = \langle \{s \triangleq_x t\}; \emptyset; \emptyset; \iota \rangle$ and $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$ merged. We define the merged set of computed linear $\mathfrak{a}$-generalizations, denoted $\mathfrak{C}_{\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}}(s,t)$ as

$$\mathfrak{C}_{\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}}(s,t) = \{x\theta\mu_D \mid \langle \emptyset; S; D; \theta \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C}) \text{ and } \mu_D \texttt{ las of } \langle \emptyset; S; D; \theta \rangle\}.$$

It remains to prove that the set $\mathfrak{C}_{\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}}(s,t)$ for any pair of terms s and t is the minimal complete set of linear generalizations for which some lemmas need to be proved.

Lemma 4.2.6. *Let* $\mathcal{C} = \langle A; S; D; \theta \rangle$ *be a linear configuration such that* $\varepsilon_f \triangleq_x f(u,v) \in A$ *(resp.* $f(u,v) \triangleq_x \varepsilon_f \in A$*),* $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$ *and* μ_{D_f} *the* `las` *of* $\mathcal{C}_f$*. Then* $|var(x\theta_f\mu_{D_f})| = 1$ *and there exists* $\{\varepsilon_f \triangleq_y t\} \subseteq S_f$ *(resp.* $\{t \triangleq_y \varepsilon_f\} \subseteq S_f$*) where* $y \in var(x\theta_f\mu_{D_f})$ *and* t *is a subterm of* $f(u,v)$*.*

PROOF: We use induction over the derivation length.

Base case

If the derivation has length 0, $\mathcal{C}$ is a final configuration, and the lemma holds vacuously.

Induction Step

We assume that the lemma holds for derivations of length less than n, and we prove it for derivations of length n. Without loss of generality, assume that we have the derivation:

$$\langle \{\varepsilon_f \triangleq_x f(u,v)\} \cup A_1; S; D; \theta\rangle \overset{ExpLA1}{\Longrightarrow}$$

$$\langle \{\varepsilon_f \triangleq_{x_1} u\} \cup A_1; S; \{\star \triangleq_{x_2} v\} \cup D; \theta\{x \mapsto f(x_1,x_2)\}\rangle \Longrightarrow^{n-1} \mathcal{C}_f$$

where $\mathcal{C}_f \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^\ell}(\mathcal{C})$ is the final linear configuration of the form $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle$. By the induction hypothesis we have that $|var(x_1\theta_f\mu_{D_f})| = 1$ and there exists $\{\varepsilon_f \triangleq_y t\} \subseteq S'$ where $y \in var(x_1\theta_f\mu_{D_f})$ and t a subterm of u. Since $x\theta_f\mu_{D_f} = f(x_1\theta_f\mu_{D_f}, x_2\theta_f\mu_{D_f})$ and $\mathcal{C}_f$ is linear, we have that

$$|var(x\theta_f\mu_{D_f})| = |var(x_1\theta_f\mu_{D_f})| + |var(x_2\theta_f\mu_{D_f})| = 1 + var(v) = 1.$$

Moreover, as $y \in var(x\theta_f\mu_{D_f})$ and t is a subterm of $f(u,v)$, it follows that $\{\varepsilon_f \triangleq_y t\}$ is the desired AUT in S_f. □

Lemma 4.2.7. *Let s and t be terms, $\mathcal{C} = \langle \{s \triangleq_x t\}; \emptyset; \emptyset; \iota\rangle$ be the associated initial configuration, and $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^\ell}(\mathcal{C})$. Then, for all $y \in labels(S_f)$ and any non-variable term r, $x\theta_f\mu_{D_f}\{y \mapsto r\} \notin \mathcal{G}^\ell_{\mathfrak{a}}(s,t)$*

PROOF: Note that $var(x\theta_f\mu_{D_f}) = labels(S_f)$. Let $y \in var(x\theta_f\mu_{D_f})$ and $p \in pos(x\theta_f\mu_{D_f})$ such that $(x\theta_f\mu_{D_f})|_p = y$; we analyze the cases below.

1. If $p \in pos(s) \cap pos(t)$ is because y is a generalization of terms that are not related to an expansion. As $y \in labels(S_f)$, there exists a solved AUT $s' \triangleq_y t'$ in S_f, i.e., $head(s') \neq head(t')$ and these symbols are not related absorptive symbols. In $x\theta_f\mu_{D_f}\{y \mapsto r\}$, the non-variable term r replaces y which was a generalization of s' and t', but by this replacement, $head(r)$ will clash with $head(s')$, $head(t')$, or both. Hence, $x\theta_f\mu_{D_f}\{y \mapsto r\}$ cannot be a generalization of s' and t', which implies $x\theta_f\mu_{D_f}\{y \mapsto r\} \notin \mathcal{G}^\ell_{\mathfrak{a}}(s,t)$.

2. If $p \notin pos(s)$ and $p \in pos(t)$ is because y is generalization of some terms in the expansion, i.e., there exists an expansible position q where $q \sqsubset p$, $s|_q = \varepsilon_f$ and $t|_q = f(u,v)$ and $(x\theta_f\mu_{D_f})|_q$ is a y-collapsible subterm of the generalization $x\theta\mu_{D_f}$. Notice as $s|_p$ and $t|_p$ are respective subterms of s and t, by Lemma 2.3.5, there exists $\mathcal{C}' = \langle \{s|_q \triangleq_z t|_q\} \cup A'; S'; D'; \theta'\rangle$ such that $\langle \{s \triangleq_x t\}; \emptyset; \emptyset; \iota\rangle \Longrightarrow^* \mathcal{C}' \Longrightarrow^* \mathcal{C}_f$. Additionally, from Lemma 4.2.6 we have that $|var((x\theta\mu_D)|_q)| = 1$ implying that y is the only variable that collapses $(x\theta_f\mu_{D_f})|_q$ and is a generalization of the AUT $\varepsilon_f \triangleq_y t'$ in S_f and t' is a subterm of $f(u,v)$. Then consider the following cases for r in the term $x\theta_f\mu_{D_f}\{y \mapsto r\}$.

(a) If $r = \varepsilon_f$, then y collapses the term and $(x\theta_f\mu_{D_f}\{y \mapsto r\})|_q \approx_{\mathfrak{a}} \varepsilon_f$ implying that $(x\theta_f\mu_{D_f}\{y \mapsto r\})|_q$ is not a generalization of $f(u, v)$. Therefore, we have that $x\theta_f\mu_{D_f}\{y \mapsto r\} \notin \mathcal{G}^{\ell}_{\mathfrak{a}}(s, t)$.

(b) If $r \neq \varepsilon_f$ and $head(r) = f$. Then the term $(x\theta\mu_D\{y \mapsto r\})|_p = f(u', v')$ is not a generalization of $\varepsilon_f \triangleq_y t'$, since $head(t') \neq f$. Hence $x\theta_f\mu_{D_f}\{y \mapsto r\} \notin \mathcal{G}^{\ell}_{\mathfrak{a}}(s, t)$.

(c) If $r \neq \varepsilon_f$ and $head(r) \neq f$. The subterm $(x\theta_f\mu_{D_f}\{y \mapsto r\})|_q$ is not collapsible anymore and then $(x\theta_f\mu_{D_f}\{y \mapsto r\})|_q$ is not a generalization of $\varepsilon_f \triangleq_y t'$, implying that $x\theta_f\mu_{D_f}\{y \mapsto r\} \notin \mathcal{G}^{\ell}_{\mathfrak{a}}(s, t)$. □

Lemma 4.2.8 (Minimality of $\mathfrak{C}_{\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}}(s, t)$)**.** *Let s and t be ground terms and $r_1, r_2 \in \mathfrak{C}_{\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}}(s, t)$, if $r_1 \neq r_2$ then neither $r_1 \preceq_{\mathfrak{a}} r_2$ nor $r_2 \preceq_{\mathfrak{a}} r_1$ holds.*

Proof: Without loss of generality, consider that $r_1 = x\theta_1\mu_{D_1}$ and $r_2 = x\theta_2\mu_{D_2}$ are computed from merged configurations $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\langle\{s \triangleq_x t\}; \emptyset; \emptyset; \iota\rangle)$ where $\mathcal{C}_1 = \langle\emptyset; S_1; D_1; \theta_1\rangle$ and $\mathcal{C}_2 = \langle\emptyset; S_2; D_2; \theta_2\rangle$ respectively. By Lemma 4.2.7, for all $y_1 \in labels(S_1)$ we have $r_1\{y_1 \mapsto r\} \notin \mathcal{G}^{\ell}_{\mathfrak{a}}(s, t)$ and for all $y_2 \in labels(S_2)$ we have $r_2\{y_2 \mapsto r\} \notin \mathcal{G}^{\ell}_{\mathfrak{a}}(s, t)$. If r is a variable and $r \in labels(S_1) \cup labels(S_2)$, then $r_1\{x \mapsto r\} \notin \mathcal{G}^{\ell}_{\mathfrak{a}}(s, t)$ because labels in $labels(S_1) \cup labels(S_2)$ are assigned to unique AUTs (due to merging of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$) and then x and r generalize different terms. Thus, $r \notin labels(S_1) \cup labels(S_2)$ implying that $r_1 \preceq_{\mathfrak{a}} r_2$ nor $r_2 \preceq_{\mathfrak{a}} r_1$ hold. □

Theorem 4.2.9. *For all terms s, t, $\mathfrak{C}_{\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}}(s, t)$ is the minimal complete set of linear generalizations of s and t.*

By Theorem 4.2.2, $\mathfrak{C}_{\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}}(s, t)$ is complete. Minimality follows from Lemma 4.2.8. Implying that $\mathfrak{C}_{\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}}(s, t) = mcsg^{\ell}_{\mathfrak{a}}(s, t)$. □

Example 4.2.10. Following Example 4.1.23 the minimal complete set of linear $\mathfrak{a}$-generalizations of the terms $g(\varepsilon_f, a)$ and $g(f(h(\varepsilon_f), a), \varepsilon_f)$ is giving by:

$$\mathfrak{C}_{\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}}(g(\varepsilon_f, a), g(f(h(\varepsilon_f), a), \varepsilon_f)) = \{g(f(y_1, a), w_2), g(f(h(\varepsilon_f), y_4), w_2)\}$$

Indeed, these terms are not comparable, since there does not exist substitutions σ and ρ such that $g(f(y_1, a), w_2)\sigma \preceq_{\mathfrak{a}} g(f(h(\varepsilon_f), y_4), w_2)$ or $g(f(h(\varepsilon_f), y_4), w_2)\rho \preceq_f g(f(y_1, a), w_2)$ respectively. ◇

Chapter 5

Absorptive Commutative Anti-unification

Several algebras that have the absorptive property, such as Semirings and Boolean Algebras, include the commutative property for one or both operations. This chapter focuses on anti-unification modulo equational theories, including $\mathfrak{a}$-symbols, C-symbols, and $\mathfrak{a}$C-symbols, i.e., we treat the $(\mathfrak{a})(\mathrm{C})(\mathfrak{a}\mathrm{C})$-theory, for short, we denote it as $\mathfrak{a}$C-theory. The algorithms $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$ and $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}{}^{\ell}}$ are introduced, and properties as termination, soundness, and completeness for the linear variant are provided in this chapter. The goal of $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$ and $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}{}^{\ell}}$ is to compute generalizations and linear generalizations for terms in $\mathfrak{a}$C-theory, respectively. As now C-symbols and $\mathfrak{a}$C-symbols are allowed, new definitions are needed.

Example 5.0.1. Let $g(g(a,c),a)$ and $g(b,g(c,b))$ be terms, where g is a C-symbol. The term $g(g(x,c),x)$ is an $\mathfrak{a}$C-generalization since, $g(g(x,c),x)\{x \mapsto a\} = g(g(a,c),a)$ and $g(g(x,c),x)\{x \mapsto b\} = g(g(b,c),b) \approx_{\mathfrak{a}\mathrm{C}} g(b,g(c,b))$. ◇

Notice that the obtained generalization $g(g(x,c),x)$ of the example above is an `lgg` since there does not exist a more specific $\mathfrak{a}$C-generalization, and that this term would not be a generalization if g were a syntactic symbol.

5.1 A Sound Algorithm for Absorptive Commutative Anti-Unification

The algorithm for absorptive commutative anti-unification uses some of the same notions that were introduced in Chapters 1 and 2, such as AUT, configuration, and initial configuration, but definitions such as merged set of AUTs and the abstraction set need to be modified.

Definition 5.1.1 (Merged Set of AUTs over $\mathfrak{a}$C)**.** Let A be a valid set of AUTs, A is *merged* if there do not exist diferent AUTs in A, $\mathtt{a}_1$ and $\mathtt{a}_2$, such that $lhs(\mathtt{a}_1) \approx_{\mathfrak{a}\mathrm{C}} lhs(\mathtt{a}_2)$ and $rhs(\mathtt{a}_1) \approx_{\mathfrak{a}\mathrm{C}} rhs(\mathtt{a}_2)$.

A configuration $\mathcal{C}$ is a *final configuration* if the active set is empty and the store is merged.

As we want to compute $\mathfrak{a}$C-generalizations and linear $\mathfrak{a}$C-generalizations for a valid set of AUTs, then we work with substitutions generalizations over $\mathfrak{a}$C, i.e., with substitutions $\gamma \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(A)$ and $\gamma \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(A)$, respectively.

Definition 5.1.2 (Algorithm $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$)**.** The algorithm $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$ works over configurations and is defined as an exhaustive application of the inference rules in Table 5.1.

Notice that the inference rules of this algorithm are the set of rules of $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ including the rule (Com) and rule (Mer) modified. The rule (Com) is included with the intuition of capturing generalizations for terms that are headed with a C-symbol, and the rule (Mer) is modified since now we can have C-equivalent terms. The explanation of the commutative rule is presented below.

(Com): **Commutative**

$$\langle \{f(s_1, s_2) \triangleq_x f(t_1, t_2)\} \uplus A; S; D; \theta \rangle \overset{Com}{\Longrightarrow}$$
$$\langle \{s_1 \triangleq_{y_1} t_2, s_2 \triangleq_{y_2} t_1\} \cup A; S; D; \theta\{x \mapsto f(y_1, y_2)\} \rangle,$$

where f is an C-symbol and $y_1, \ldots, y_n$ are fresh variables.

Explanation: The commutative rule is applied when terms in the selected AUT have the same C-symbol as head. This rule is a shorthand for two steps. The first step is to consider the right-hand side term as its C-equivalent term $f(t_2, t_1)$. The second step is to decompose the obtained AUT: $f(s_1, s_2) \triangleq_x f(t_2, t_1)$. Then we associate the label x to more specific generalization $f(y_1, y_2)$, where y_1 and y_2 are the respective labels of the new AUTs: $s_1 \triangleq_{y_1} t_2$ and $s_1 \triangleq_{y_2} t_2$ in the active set.

The termination of $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$ follows as an immediate consequence of Theorem 2.1.14, since the proposed measure works for the rule (Com) and the algorithm remains finitely branching.

Corollary 5.1.3 (Termination of $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$)**.** *$\mathcal{A}_{\mathfrak{a}C\text{-}AUNIF}$ terminates for any linear configuration $\mathcal{C}$ and it outputs a finite set of configurations.*

We present the soundness for $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$. As was mentioned, this algorithm is $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ modified, and some of the case rules that are analogous to the cases in Lemma 2.3.2 are omitted in the proof.

$(\overset{Dec}{\Longrightarrow})$ $$\frac{\langle \{f(s_1,\ldots,s_n) \triangleq_x f(t_1,\ldots,t_n)\} \uplus A; S; D; \theta\rangle}{\langle \{s_1 \triangleq_{y_1} t_1,\ldots,s_n \triangleq_{y_n} t_n\} \cup A; S; D; \theta\{x \mapsto f(y_1,\ldots,y_n)\}\rangle}$$
where f is an n-ary symbol, $n \geq 0$, and $y_1,\ldots,y_n$ are fresh variables.

$(\overset{Com}{\Longrightarrow})$ $$\frac{\langle \{f(s_1,s_2) \triangleq_x f(t_1,t_2)\} \uplus A; S; D; \theta\rangle}{\langle \{s_1 \triangleq_{y_1} t_2, s_2 \triangleq_{y_2} t_1\} \cup A; S; D; \theta\{x \mapsto f(y_1,y_2)\}\rangle}$$
for f a C- or $\mathfrak{a}$C-symbol and y_1, y_2 fresh variables.

$(\overset{Sol}{\Longrightarrow})$ $$\frac{\langle \{s \triangleq_x t\} \uplus A; S; D; \theta\rangle}{\langle A; \{s \triangleq_x t\} \cup S; D; \theta\rangle}$$
where $s \triangleq_x t$ is a solved AUT.

$(\overset{Mer}{\Longrightarrow})$ $$\frac{\langle \emptyset; \{s_1 \triangleq_x t_1, s_2 \triangleq_y t_2\} \uplus S; D; \theta\rangle}{\langle \emptyset; \{s_2 \triangleq_y t_2\} \cup S; D; \theta\{x \mapsto y\}\rangle}$$
where $s_1 \approx_{\mathfrak{a}\mathrm{C}} t_1$, $s_2 \approx_{\mathfrak{a}\mathrm{C}} t_2$, and $x \neq y$.

$(\overset{ExpB}{\Longrightarrow})$ $$\frac{\langle \{\varepsilon_f \triangleq_x \varepsilon_f\} \uplus A; S; D; \theta\rangle}{\langle A; S; \{\varepsilon_f \triangleq_x \varepsilon_f\} \cup D; \theta\rangle}$$
where ε_f is an absorbing constant.

In the following rules, f is an $\mathfrak{a}$- or $\mathfrak{a}$C-symbol and y_1, y_2 are fresh variables:

$(\overset{ExpLA1}{\Longrightarrow})$ $$\frac{\langle \{\varepsilon_f \triangleq_x f(t_1,t_2)\} \uplus A; S; D; \theta\rangle}{\langle \{\varepsilon_f \triangleq_{y_1} t_1\} \cup A; S; \{\star \triangleq_{y_2} t_2\} \cup D; \theta\{x \mapsto f(y_1,y_2)\}\rangle}$$

$(\overset{ExpLA2}{\Longrightarrow})$ $$\frac{\langle \{\varepsilon_f \triangleq_x f(t_1,t_2)\} \uplus A; S; D; \theta\rangle}{\langle \{\varepsilon_f \triangleq_{y_2} t_2\} \cup A; S; \{\star \triangleq_{y_1} t_1\} \cup D; \theta\{x \mapsto f(y_1,y_2)\}\rangle}$$

$(\overset{ExpRA1}{\Longrightarrow})$ $$\frac{\langle \{f(s_1,s_2) \triangleq_x \varepsilon_f\} \uplus A; S; D; \theta\rangle}{\langle \{s_1 \triangleq_{y_1} \varepsilon_f\} \cup A; S; \{s_2 \triangleq_{y_2} \star\} \cup D; \theta\{x \mapsto f(y_1,y_2)\}\rangle}$$

$(\overset{ExpRA2}{\Longrightarrow})$ $$\frac{\langle \{f(s_1,s_2) \triangleq_x \varepsilon_f\} \uplus A; S; D; \theta\rangle}{\langle \{s_2 \triangleq_{y_2} \varepsilon_f\} \cup A; S; \{s_1 \triangleq_{y_1} \star\} \cup D; \theta\{x \mapsto f(y_1,y_2)\}\rangle}$$

Table 5.1: Generalization $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$ rules for $(\mathfrak{a})$(C)$(\mathfrak{a}$C)-theory.

Theorem 5.1.4 (Auxiliary Soundness of $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$)**.** *Let* $\mathcal{C} = \langle A; S; D; \theta\rangle$ *be a configuration and* $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}C\text{-}AUNIF}(\mathcal{C})$*. Then* $label(\mathtt{a})\theta_f \in \mathcal{G}_{\mathfrak{a}C}(lhs(\mathtt{a}), rhs(\mathtt{a}))$*, for all* $\mathtt{a} \in A \cup S$*.*

PROOF: We proceed by induction over the derivation length.

Base case. If the derivation has length 0, then $\mathcal{C}$ is a final configuration of the form $\langle \emptyset; S; D; \theta\rangle$. By the definition of configuration, for all $\mathtt{a} \in S$ we have $label(\mathtt{a})\theta = label(\mathtt{a})$. Moreover, since each $\mathtt{a}$ is solved, it follows that $label(\mathtt{a})\theta_f \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$.

Inductive Step. Now, consider a derivation having the following form:

$$\langle A; S; D; \theta\rangle \Longrightarrow \langle A'; S'; D'; \theta'\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle \tag{5.1}$$

For $n \geq 0$. We assume for the induction hypothesis that for derivations of the form

$$\langle A'; S'; D'; \theta'\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle,$$

the theorem holds and show that the theorem holds for derivations of the form presented in Derivation 5.1. We continue the proof considering the various options for the transition from $\langle A; S; D; \theta\rangle$ to $\langle A'; S'; D'; \theta'\rangle$.

1. **(Com)**. Assume that the derivation is of the form:

$$\langle \{f(s_1, s_2) \triangleq_y f(t_1, t_2)\} \cup A_1; S; D; \theta\rangle \overset{Com}{\Longrightarrow}$$
$$\langle \{s_1 \triangleq_{x_1} t_2, s_2 \triangleq_{x_2} t_1\} \cup A_1; S; D; \theta'\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle$$

   where $\theta' = \theta\{y \mapsto f(x_1, x_2)\}$. By the induction hypothesis, we know that for all AUT $\mathtt{a}$ of the active set $\{s_1 \triangleq_{x_1} t_2, s_2 \triangleq_{x_2} t_1\} \cup A_1$, $label(\mathtt{a})\theta_f \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$, in particular we have that $x_1\theta_f \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(s_1, t_2)$ and $x_2\theta_f \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(s_2, t_1)$, implying that $y\theta_f = f(x_1, x_2)\theta_f \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(f(s_1, s_2), f(t_2, t_1))$. Finally, since $f(t_2, t_1) \approx_{\mathfrak{a}\mathrm{C}} f(t_1, t_2)$, it holds that $y\theta_f \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(f(s_1, s_2), f(t_1, t_2))$.

2. **(Mer)** Assume that the derivation is of the form:

$$\langle \emptyset; \{s_1 \triangleq_y t_1, s_2 \triangleq_z t_2\} \cup S_1; D; \theta\rangle \overset{Mer}{\Longrightarrow}$$
$$\langle \emptyset; \{s_2 \triangleq_z t_2\} \cup S_1; D; \theta'\rangle; \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle.$$

   where $\theta' = \theta\{y \mapsto z\}$. By induction hypothesis, we have that $z\theta_f \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(s_2, t_2)$. This implies that $y\theta_f = y\{y \mapsto z\}\theta_f \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(s_2, t_2)$. Since $s_1 \approx_{\mathfrak{a}\mathrm{C}} s_2$ and $t_1 \approx_{\mathfrak{a}\mathrm{C}} t_2$, it follows that $y\theta_f \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(s_1, t_1)$. □

Example 5.1.5. Let $g(g(a, c), a) \triangleq_x g(b, g(c, b))$ be an AUT, where g is a C-symbol. Two rules can be applied, the rules (Dec) and (Com) that leads to three derivations:

Derivation 1:
$$\langle \{g(g(a, c), a) \triangleq_x g(b, g(c, b))\}; \emptyset; \emptyset; \iota\rangle \overset{Dec}{\Longrightarrow}$$
$$\langle \{g(a, c) \triangleq_{y_1} b, a \triangleq_{y_2} g(c, b)\}; \emptyset; \emptyset; \{x \mapsto g(y_1, y_2)\}\rangle \overset{Sol \times 2}{\Longrightarrow}$$

$$\langle \emptyset; \{g(a,c) \triangleq_{y_1} b, a \triangleq_{y_2} g(c,b)\}; \emptyset; \{x \mapsto g(y_1,y_2)\}\rangle$$

Derivation 2:
$$\langle \{g(g(a,c),a) \triangleq_x g(b,g(c,b))\}; \emptyset; \emptyset; \iota\rangle \overset{Com}{\Longrightarrow}$$
$$\langle \{g(a,c) \triangleq_y g(c,b), a \triangleq_z b\}; \emptyset; \emptyset; \{x \mapsto g(y,z)\}\rangle \overset{Dec}{\Longrightarrow}$$
$$\langle \{a \triangleq_{w_1} c, c \triangleq_{w_2} b, a \triangleq_z b\}; \emptyset; \emptyset; \{x \mapsto g(g(w_1,w_2),z), y \mapsto g(w_1,w_2)\}\rangle \overset{Sol \times 3}{\Longrightarrow}$$
$$\langle \emptyset; \{a \triangleq_{w_1} c, c \triangleq_{w_2} b, a \triangleq_z b\}; \emptyset; \{x \mapsto g(g(w_1,w_2),z), y \mapsto g(w_1,w_2)\}\rangle$$

Derivation 3:
$$\langle \{g(g(a,c),a) \triangleq_x g(b,g(c,b))\}; \emptyset; \emptyset; \iota\rangle \overset{Com}{\Longrightarrow}$$
$$\langle \{g(a,c) \triangleq_y g(c,b), a \triangleq_z b\}; \emptyset; \emptyset; \{x \mapsto g(y,z)\}\rangle \overset{Com}{\Longrightarrow}$$
$$\langle \{a \triangleq_{w_3} b, c \triangleq_{w_4} c, a \triangleq_z b\}; \emptyset; \emptyset; \{x \mapsto g(g(w_3,w_4),z), y \mapsto g(w_3,w_4)\}\rangle \overset{Dec}{\Longrightarrow}$$
$$\langle \{a \triangleq_{w_3} b, a \triangleq_z b\}; \emptyset; \emptyset; \{x \mapsto g(g(w_3,c),z), y \mapsto g(w_3,c), w_4 \mapsto c\}\rangle \overset{Sol \times 2}{\Longrightarrow}$$
$$\langle \emptyset; \{a \triangleq_{w_3} b, a \triangleq_z b\}; \emptyset; \{x \mapsto g(g(w_3,c),z), y \mapsto g(w_3,c), w_4 \mapsto c\}\rangle \overset{Mer}{\Longrightarrow}$$
$$\langle \emptyset; \{a \triangleq_z b\}; \emptyset; \{x \mapsto g(g(z,c),z), y \mapsto g(z,c), w_4 \mapsto c, w_3 \mapsto z\}\rangle$$

The final configuration of Derivation 3 computes the generalization $g(g(z,c),z)$ that is not computed by $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$, and that is more specific than the other generalizations computed in Derivations 1 and 2. $\diamond$

5.2 Abstraction Computation and Completeness

In this section, we construct the *abstraction set* and *substitutions* for absorptive commutative theories. Since C-symbols and $\mathfrak{a}$C-symbols are included, the set is defined using the relation induced by the axioms of $\mathfrak{a}$C-theory.

Definition 5.2.1 (Abstraction set)**.** Let t be a term in $\mathfrak{a}$-normal form, and σ be a substitution whose range is in $\mathfrak{a}$-normal form. The set defined below is the abstraction set of t with respect to σ over $\mathfrak{a}$C.

$$\uparrow(t,\sigma) := \{r \mid r\sigma \approx_{\mathfrak{a}\mathrm{C}} t,\ r \text{ is in an } \mathfrak{a}\text{-normal form and } var(r) \subseteq dom(\sigma)\}.$$

In words, $\uparrow(t,\sigma)$ is the set of all those $\mathfrak{a}$C-generalizations of t, whose σ-instances equal t, and that may contain only variables from $dom(\sigma)$.

Example 5.2.2. Consider the term $f(a, h(a))$ with f an $\mathfrak{a}$C-symbol, h a syntactic symbol, and the substitution $\sigma = \{x \mapsto a\}$. Then the abstraction set of $f(a, h(a))$ with respect to σ over $\mathfrak{a}$C is:

$$\uparrow(f(a, h(a)), \sigma) = \left\{ \begin{array}{l} f(a, x), f(x, a), f(a, h(a)), f(x, h(x)), \\ f(h(x), x), f(x, h(a)), f(x, h(a)) \end{array} \right\},$$

$\diamond$

In this approach, the set of abstraction substitutions is treated as in Definition 2.2.19 but using the abstraction set of Definition 5.2.1. The set $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}(\mathcal{C})$ is the set of all final configurations derived from the configuration $\mathcal{C}$. Moreover, for all $\mathtt{a} \in A$, the active set of some configuration $\mathcal{C}$, a *computed generalization* of $\mathtt{a}$ generated by $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$ and the set of abstractions substitutions, is the term $label(\mathtt{a})\theta_f\tau$, where θ_f is the computed substitution for some $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}(\mathcal{C})$ and $\tau \in \Psi(D_f, S_f)$.

From Lemma 5.1.4, for all $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}(\mathcal{C})$, where $\mathcal{C}$ is of the form $\langle \{s \triangleq_x t\}; \emptyset; \emptyset; \iota \rangle$, we get that $x\theta_f$ is a $\mathfrak{a}$C-generalization for the terms s and t. It remains to prove that using an abstraction substitution τ over $x\theta_f$ produces a $\mathfrak{a}$C-generalization. Furthermore, we prove that $\theta_f\tau$ is a substitution generalization for all AUTs in any configuration.

Lemma 5.2.3 (Soundness of Set of Abstraction Substitutions for Final Configurations)**.** *Let $\mathcal{C}_f$ be a final configuration of the form $\langle \emptyset; S_f; D_f; \theta_f \rangle$. Then, for all $\tau \in \Psi(D_f, S_f)$ we have that the substitution $(\theta_f\tau)|_{labels(S_f \cup D_f)} \in \mathcal{G}_{\mathfrak{a}C}(\mathcal{C}_f)$.*

Proof: We prove that the substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$ are the associated substitutions of $\theta_f\tau$. We analyze the following cases:

1. If $\mathtt{a} \in S_f$, then since $dom(\theta_f) \cap labels(S_f) = \emptyset$ and $dom(\tau) = labels(D_f)$, it follows that $label(\mathtt{a})\theta_f\tau = label(\mathtt{a})$. This implies that $label(\mathtt{a})\theta_f\tau\sigma^l_{S_f} = lhs(\mathtt{a})$ and $label(\mathtt{a})\theta_f\tau\sigma^r_{S_f} = rhs(\mathtt{a})$.

2. If $\mathtt{a} \in D_f$, we get that $dom(\theta_f) \cap labels(D_f) = \emptyset$, and then $label(\mathtt{a})\theta_f\tau = label(\mathtt{a})\tau$. Also, by Definitions 2.2.19 and 5.2.1, for all $\mathtt{a} \in D_f$, $label(\mathtt{a})\theta_f\tau \in \uparrow_{label(\mathtt{a})}(D_f, S_f)$. Then, we have that:

   (a) $label(\mathtt{a})\theta_f\tau\sigma^l_{S_f} \approx_{\mathfrak{a}\text{C}} lhs(\mathtt{a})$ if $\mathtt{a} \in D^l_f$,

   (b) $label(\mathtt{a})\theta_f\tau\sigma^r_{S_f} \approx_{\mathfrak{a}\text{C}} rhs(\mathtt{a})$ if $\mathtt{a} \in D^r_f$, or

   (c) $label(\mathtt{a})\theta_f\tau\sigma^l_{S_f} \approx_{\mathfrak{a}\text{C}} lhs(\mathtt{a})$ and $label(\mathtt{a})\theta_f\tau\sigma^r_{S_f} \approx_{\mathfrak{a}\text{C}} rhs(\mathtt{a})$ if $\mathtt{a} \in D^n_f$.

$\square$

Theorem 5.2.4 (Soundness of Computed Generalizations)**.** *Let* $\mathcal{C} = \langle A; S; D; \theta\rangle$ *be a configuration and* $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}C\text{-}AUNIF}(\mathcal{C})$*. Then, for all* $\tau \in \Psi(D_f, S_f)$ *it holds that* $(\theta_f\tau)|_{label(A\cup S\cup D)} \in \mathcal{G}_{\mathfrak{a}C}(\mathcal{C})$ *with associated substitutions* $\sigma^l_{S_f}$ *and* $\sigma^r_{S_f}$*.*

Proof: We proceed by induction over the derivation length.

Base case. If the derivation has length 0, $\mathcal{C}$ is a final configuration and the theorem holds by Lemma 5.2.3.

Inductive Step. Now, consider $\mathcal{C} = \langle A; S; D; \theta\rangle$ and a derivation having the following form:

$$\mathcal{C} \Longrightarrow \mathcal{C}' \Longrightarrow^n \mathcal{C}_f \tag{5.2}$$

for $n \geq 0$, and where $\mathcal{C}' = \langle A'; S'; D'; \theta'\rangle$ and $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle$. We assume for the induction hypothesis that for derivations of the form $\mathcal{C}' \Longrightarrow^n \mathcal{C}_f$, the theorem holds, and show that the theorem holds for derivations of the form presented in Derivation 5.2. We continue the proof considering the various options for the transition from $\mathcal{C}$ to $\mathcal{C}'$.

1. **(Dec)**. Assume that the derivation is of the form:
$$\langle \{f(s_1, \ldots, s_m) \triangleq_y f(t_1, \ldots, t_m)\} \cup A_1; S; D; \theta\rangle \overset{Dec}{\Longrightarrow}$$
$$\langle A'; S; D; \theta'\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle$$
where $A' = \{s_1 \triangleq_{x_1} t_1, \ldots, s_m \triangleq_{x_m} t_m\} \cup A_1$ and $\theta' = \theta\{y \mapsto f(x_1, \ldots, x_m)\}$. By induction hypothesis, we have that $(\theta_f\tau)|_{label(A'\cup S\cup D)} \in \mathcal{G}_{\mathfrak{a}C}(\mathcal{C}')$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$. In particular, for all $\mathtt{a}$ of $\{s_i \triangleq_{x_i} t_i\}_{1\leq i\leq m}$, it holds that $x_i\theta_f\tau\sigma^l_{S_f} \approx_{\mathfrak{a}C} s_i$ and $x_i\theta_f\tau\sigma^r_{S_f} \approx_{\mathfrak{a}C} t_i$, for $1 \leq i \leq m$. From Lemma 2.3.1, it holds that $y\theta_f = f(x_1\theta_f, \ldots, x_m\theta_f)$. Applying substitutions $\tau\sigma^l_{S_f}$ and $\tau\sigma^r_{S_f}$ respectively to $y\theta_f$, we obtain that $y\theta_f\tau\sigma^l_{S_f} = f(x_1\theta_f\tau\sigma^l_{S_f}, \ldots, , x_m\theta_f\tau\sigma^l_{S_f}) \approx_{\mathfrak{a}C} f(s_1, \ldots, s_m)$ and $y\theta_f\tau\sigma^r_{S_f} = f(x_1\theta_f\tau\sigma^r_{S_f}, \ldots, x_m\theta_f\tau\sigma^r_{S_f}) \approx_{\mathfrak{a}C} f(t_1, \ldots, t_m)$. Hence, we have that $(\theta_f\tau)|_{label(A\cup S\cup D)} \in \mathcal{G}_{\mathfrak{a}C}(\mathcal{C})$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$.

2. **(Com)**. Assume that the derivation is of the form:
$$\langle \{f(s_1, s_2) \triangleq_y f(t_1, t_2)\} \cup A_1; S; D; \theta\rangle \overset{Com}{\Longrightarrow}$$
$$\langle \{s_1 \triangleq_{x_1} t_2, s_2 \triangleq_{x_2} t_1\} \cup A_1; S; D; \theta'\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle$$
where $\theta' = \theta\{y \mapsto f(x_1, x_2)\}$. By induction hypothesis, $(\theta_f\tau)|_{label(A'\cup S\cup D)} \in \mathcal{G}_{\mathfrak{a}C}(\mathcal{C}')$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$. In particular, for $1 \leq i \leq 2$, it holds that $x_i\theta_f\tau\sigma^l_{S_f} \approx_{\mathfrak{a}C} s_i$ and $x_i\theta_f\tau\sigma^r_{S_f} \approx_{\mathfrak{a}C} t_{i \bmod 2+1}$. Notice that $y\theta_f\tau = f(x_1\theta_f\tau, x_2\theta_f\tau)$ then applying $\sigma^l_{s_f}$, we have that $y\theta_f\tau\sigma^l_{s_f} = f(x_1\theta_f\tau\sigma^l_{s_f}, x_2\theta_f\tau\sigma^l_{s_f}) \approx_{\mathfrak{a}C} f(s_1, s_2)$ and applying $\sigma^r_{s_f}$, we have that $y\theta_f\tau\sigma^r_{s_f} = f(x_1\theta_f\tau\sigma^r_{s_f}, x_2\theta_f\tau\sigma^r_{s_f}) \approx_{\mathfrak{a}C} f(t_1, t_2)$. Hence, we have that $(\theta_f\tau)|_{label(A\cup S\cup D)}$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$.

3. **(Sol)**. Assume that the derivation is of the form:
$$\langle \{s \triangleq_y t\} \cup A_1; S; D; \theta\rangle \overset{Sol}{\Longrightarrow} \langle A_1; S'; D; \theta\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle,$$

where $S' = \{s \triangleq_y t\} \cup S$. By induction hypothesis, $(\theta_f\tau)|_{label(A' \cup S \cup D)} \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(\mathcal{C}')$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$. In particular, since $s \triangleq_y t \in S'$, it holds that $y\theta_f\tau\sigma^l_{S_f} \approx_{\mathfrak{a}\mathrm{C}} s$ and $y\theta_f\tau\sigma^r_{S_f} \approx_{\mathfrak{a}\mathrm{C}} t$. Therefore, we have that $(\theta_f\tau)|_{label(A \cup S \cup D)} \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(\mathcal{C})$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$.

4. **(Mer)** Assume that the derivation is of the form:
$$\langle \emptyset; \{s \triangleq_y t, s \triangleq_z t\} \cup S_1; D; \theta\rangle \overset{Mer}{\Longrightarrow}$$
$$\langle \emptyset; S'; D; \theta\{y \mapsto z\}\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle.$$
Where $S' = \{s \triangleq_z t\} \cup S_1$. By induction hypothesis, $(\theta_f\tau)|_{label(S' \cup D)} \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(\mathcal{C}')$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$. In particular, since $s \triangleq_z t \in S'$, it holds that $z\theta_f\tau\sigma^l_{S_f} \approx_{\mathfrak{a}\mathrm{C}} s$ and $z\theta_f\tau\sigma^r_{S_f} \approx_{\mathfrak{a}\mathrm{C}} t$. Finally, from Lemma 2.3.1, it holds that $y\theta_f = z$. Therefore, we have that $(\theta_f\tau)|_{label(S \cup D)} \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(\mathcal{C})$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$.

5. **(ExpB)**. Assume that the derivation is of the form:
$$\langle \{\varepsilon_f \triangleq_y \varepsilon_f\} \cup A_1; S; D; \theta\rangle \overset{ExpB}{\Longrightarrow}$$
$$\langle A_1; S; D'; \theta\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle$$
where $D' = \{\varepsilon_f \triangleq_y \varepsilon_f\} \cup D$. By induction hypothesis, $(\theta_f\tau)|_{label(A_1 \cup S \cup D')} \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(\mathcal{C}')$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$. In particular, since $\varepsilon_f \triangleq_y \varepsilon_f \in D'$, it holds that $y\theta_f\tau\sigma^l_{S_f} \approx_{\mathfrak{a}\mathrm{C}} \varepsilon_f$ and $y\theta_f\tau\sigma^r_{S_f} \approx_{\mathfrak{a}\mathrm{C}} \varepsilon_f$. Thus, $(\theta_f\tau)|_{label(A \cup S \cup D)} \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(\mathcal{C})$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$.

6. **(ExpLA1)**. Assume that the derivation is of the form:
$$\langle \{\varepsilon_f \triangleq_y f(s,t)\} \cup A_1; S; D; \theta\rangle \overset{ExpLA1}{\Longrightarrow}$$
$$\langle A'; S; D'; \theta'\rangle\rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f\rangle$$
where $A' = \{\varepsilon_f \triangleq_{x_1} s\} \cup A_1$, $D' = \{\star \triangleq_{x_2} t\} \cup D$, and $\theta' = \theta\{y \mapsto f(x_1, x_2)\}$. By induction hypothesis, we have that $(\theta_f\tau)|_{label(A' \cup S \cup D')} \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(\mathcal{C}')$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$. In particular,since $\varepsilon_f \triangleq_{x_1} s \in A'$, it holds that $x_1\theta_f\tau\sigma^l_{S_f} \approx_{\mathfrak{a}\mathrm{C}} \varepsilon_f$ and $x_1\theta_f\tau\sigma^r_{S_f} \approx_{\mathfrak{a}\mathrm{C}} s$. Also, $\star \triangleq_{x_2} t \in D'$, it holds that $x_2\theta_f\tau\sigma^r_{S_f} \approx_{\mathfrak{a}\mathrm{C}} t$. From Lemma 2.3.1, it holds that $y\theta_f = f(x_1\theta_f, x_2\theta_f)$. Applying substitutions $\tau\sigma^l_{S_f}$ and $\tau\sigma^l_{S_f}$ to $y\theta_f$, we get that $y\theta_f\tau\sigma^l_{S_f} = f(x_1\theta_f\tau\sigma^l_{S_f}, , x_2\theta_f\tau\sigma^l_{S_f}) \approx_{\mathfrak{a}\mathrm{C}} f(\varepsilon_f, x_2\theta_f\tau\sigma^l_{S_f}) \approx_{\mathfrak{a}\mathrm{C}} \varepsilon_f$ and $y\theta_f\tau\sigma^r_{S_f} = f(x_1\theta_f\tau\sigma^r_{S_f}, x_2\theta_f\tau\sigma^r_{S_f}) \approx_{\mathfrak{a}\mathrm{C}} f(s,t)$, respectively. Hence, we have that $(\theta_f\tau)|_{label(A \cup S \cup D)} \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(\mathcal{C})$ with associated substitutions $\sigma^l_{S_f}$ and $\sigma^r_{S_f}$.

7. The analysis of the other lateral expansion rules is analogous to the previous one.

□

The following example illustrates one of the computed generalizations generated by $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$ and the set of abstraction substitutions.

Example 5.2.5. Let $g(\varepsilon_f, f(a,a))$ and $g(\varepsilon_f, f(g(a,a),a))$ be terms with g a C-symbol and f an $\mathfrak{a}$C-symbol. One of the derivations of $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$:

Derivation 1:

$$\langle \{g(\varepsilon_f, f(a,a)) \triangleq_x g(\varepsilon_f, f(g(a,a),a))\}; \emptyset; \emptyset; \iota\rangle \overset{Com}{\Longrightarrow}$$
$$\langle \{\varepsilon_f \triangleq_{y_1} f(g(a,a),a), f(a,a) \triangleq_{y_2} \varepsilon_f\}; \emptyset; \emptyset; \{x \mapsto g(y_1,y_2)\}\rangle \overset{ExpLA2}{\Longrightarrow}$$
$$\langle \{\varepsilon_f \triangleq_{z_2} a, f(a,a) \triangleq_{y_2} \varepsilon_f\}; \emptyset; \{\star \triangleq_{z_1} g(a,a)\};$$
$$\{x \mapsto g(f(z_1,z_2),y_2), y_1 \mapsto f(z_1,z_2)\}\rangle \overset{ExpRA2}{\Longrightarrow}$$
$$\langle \{\varepsilon_f \triangleq_{z_2} a, a \triangleq_{z_4} \varepsilon_f\}; \emptyset; \{\star \triangleq_{z_1} g(a,a), a \triangleq_{z_3} \star\};$$
$$\{x \mapsto g(f(z_1,z_2),f(z_3,z_4)), y_1 \mapsto f(z_1,z_2), y_2 \mapsto f(z_3,z_4)\}\rangle \overset{Sol \times 2}{\Longrightarrow}$$
$$\langle \emptyset; \{\varepsilon_f \triangleq_{z_2} a, a \triangleq_{z_4} \varepsilon_f\}; \{\star \triangleq_{z_1} g(a,a), a \triangleq_{z_3} \star\};$$
$$\{x \mapsto g(f(z_1,z_2),f(z_3,z_4)), y_1 \mapsto f(z_1,z_2), y_2 \mapsto f(z_3,z_4)\}\rangle$$

For this final configuration, it is possible to find the abstraction set for variables z_1 and z_3 in the final delayed set respectively, using the substitutions $\sigma^l = \{z_2 \mapsto \varepsilon_f, z_4 \mapsto a\}$ and $\sigma^r = \{z_2 \mapsto a, z_4 \mapsto \varepsilon_f\}$.

$$\uparrow_{z_1}(g(a,a),\sigma^r) = \{g(a,a), g(a,z_2), g(z_2,a), g(z_2,z_2)\} \text{ and } \uparrow_{z_3}(a,\sigma^l) = \{a, z_4\}$$

Then, the term $g(f(g(z_2,a),z_2), f(z_4,z_4))$ is a $\mathfrak{a}$C-generalization of the initial terms is obtained by the substitution $\{z_1 \mapsto g(z_2,a), z_3 \mapsto z_4\} \in \Psi(D_f, S_f)$, where D_f and S_f are the final delayed and store sets respectively. ◇

5.3 Linear Absorptive Commutative Anti-unification

This section discusses the linear variant of the anti-unification on absorptive commutative theories. The goal is to find an algorithm that can compute the minimal complete set of generalizations for any pair of terms modulo $\mathfrak{a}$C. As in Chapter 4, some restrictions on the definitions should be made. Additionally, the algorithm $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}$ computes linear $\mathfrak{a}$C-generalizations for any pair of terms, and the proofs of soundness and completeness are presented.

Definition 5.3.1 (Algorithm $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}$)**.** The algorithm for absorptive commutative linear anti-unification, denoted as $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}$, consists of the exhaustive application of the rules of $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$ except the rules (Mer) and (ExpB).

The notions of linearity as linear configuration and linear `lgg` are the same as those presented in Chapter 4. In this section, we work with the linear substitution generalization

modulo $\mathfrak{a}$C, i.e., with linear substitutions γ such that $\gamma \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(\mathcal{C})$ as in Definitions 4.1.5 and 1.3.33. In this approach, the characterization of the linear substitutions, given in Corollary 4.1.7 of Chapter 4, is preserved. The following corollaries and lemmas are extended for the theory treated in this section.

Lemma 5.3.2 (Preservation of Linear Configurations)**.** *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration, any configuration $\mathcal{C}' = \langle A'; S'; D'; \theta'\rangle$ derived from $\mathcal{C}$ by $\mathcal{A}_{\mathfrak{a}C\text{-}AU_{NIF}{}^{\ell}}$ is also a linear configuration.*

Proof: We analyze each rule application in the derivation $\mathcal{C} \Longrightarrow \mathcal{C}'$. The analysis of the (Dec), (Sol), and the expansion rules is analogous to that presented in Lemma 4.1.13. We show the analysis for rule (Com):

- Rule (Com) introduces two fresh variables , say y_1 and y_2, in $\mathcal{V}ran(\theta')$ and $\theta' = \theta\{label(\mathtt{a}) \mapsto t\}$, where $t = head(lhs(\mathtt{a}))(y_1, y_2)$ with $head(lhs(\mathtt{a})) = head(rhs(\mathtt{a}))$ a C-symbols for some $\mathtt{a} \in A$. Note that y_1 and y_2 occurs once in t and for all $t' \in ran(\theta') - \{t\}$, we have two cases to analyze:

  1. If $t' \in ran(\theta)$ the lemma holds trivially.
  2. If $t' \notin ran(\theta)$, i.e., $t' = t''\{label(\mathtt{a}) \mapsto t\}$ for some $t'' \in ran(\theta)$ and $label(\mathtt{a}) \in var(t'')$. Observe that any $x \in var(t') = (var(t'') - \{label(\mathtt{a})\}) \uplus var(t)$ occurs once in t' and the lemma holds. □

The termination (Corollary 5.1.3) for the general algorithm $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$ implies the termination for the linear variant.

Corollary 5.3.3 (Termination of $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}{}^{\ell}}$)**.** *$\mathcal{A}_{\mathfrak{a}C\text{-}AU_{NIF}{}^{\ell}}$ terminates for any linear configuration $\mathcal{C}$ and it outputs a finite set of configurations.*

As for $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}}$, the analysis of algorithm $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}{}^{\ell}}$ assumes only configurations derived from initial configurations.

Lemma 5.3.4 (Auxiliary Soundness of $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}}{}^{\ell}}$)**.** *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration and a final configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}C\text{-}AU_{NIF}{}^{\ell}}(\mathcal{C})$. Then, for all $\mathtt{a} \in A$ we have that $label(\mathtt{a})\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}C}(lhs(\mathtt{a}), rhs(\mathtt{a}))$.*

Proof: We proceed by induction over the derivation length.

Base case. If the derivation has length 0, $\mathcal{C}$ is a final configuration, and the lemma holds vacuously.

Inductive Step. Now, consider a derivation having the following form:

$$\langle A; S; D; \theta\rangle \Longrightarrow \langle A'; S'; D'; \theta'\rangle \Longrightarrow^{n} \langle \emptyset; S_f; D_f; \theta_f\rangle \tag{5.3}$$

For $n \geq 0$. We assume for the induction hypothesis that for derivations of the form

$$\langle A'; S'; D'; \theta' \rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f \rangle,$$

the lemma holds, and show that the lemma holds for derivations of the form presented in Derivation 5.3. We continue the proof considering the various options for the transition from $\langle A; S; D; \theta \rangle$ to $\langle A'; S'; D'; \theta' \rangle$.

1. **(Com)**. Assume that the derivation is of the form:

$$\langle \{f(s_1, s_2) \triangleq_y f(t_1, t_2)\} \cup A_1; S; D; \theta \rangle \overset{Com}{\Longrightarrow}$$
$$\langle \{s_1 \triangleq_{x_1} t_2, s_2 \triangleq_{x_2} t_1\} \cup A_1; S; D; \theta'' \rangle \Longrightarrow^n \langle \emptyset; S_f; D_f; \theta_f \rangle$$

   where $\theta' = \theta\{y \mapsto f(x_1, x_2)\}$. By the induction hypothesis for all $\mathtt{a} \in A_1$ we get that $label(\mathtt{a})\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(lhs(\mathtt{a}), rhs(\mathtt{a}))$, $x_1\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(s_1, t_2)$ and $x_2\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(s_2, t_1)$. From Lemma 2.3.1, $y\theta_f = y(\theta\{y \mapsto f(x_1, x_2)\})\theta_f$, then $y\theta_f = f(x_1\theta_f, x_2\theta_f)$, implying that $y\theta_f \in \mathcal{G}_{\mathfrak{a}\mathrm{C}}(f(s_1, s_2), f(t_1, t_2))$. In addition, from lemma 5.3.2 we have that θ_f is a linear substitution, then $y\theta_f$ is a linear term. Therefore, $y\theta_f \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(f(s_1, s_2), f(t_1, t_2))$.

The analysis of rules (Dec), (Sol) and expansion rules is analogous to the cases of Lemma 4.1.15. □

To prove the completeness of $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$, we extend the Lemma 4.1.16 to prove that the final computed substitution derived from $\mathcal{C}$ is a linear substitution of the terms in the active set of $\mathcal{C}$.

Lemma 5.3.5. *Let $\mathcal{C} = \langle A; S; D; \theta \rangle$ be a linear configuration, and the final configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}C\text{-}AUNIF^{\ell}}(\mathcal{C})$, then for all different $\mathtt{a}, \mathtt{a}' \in A$ we have that $var(label(\mathtt{a})\theta_f) \cap var(label(\mathtt{a}')\theta_f) = \emptyset$.*

Proof: We use induction over length derivation.

Base case

If the derivation has length 0, $\mathcal{C}$ is a final configuration and the lemma holds vacuously.

Induction Step

We assume that the lemma holds for derivations of length less than or equal to n, and we prove it for derivations with length $n + 1$. Consider the derivation $\mathcal{C} \Longrightarrow \mathcal{C}' \Longrightarrow^n \mathcal{C}_f$, where $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle$. We analyze each rule application case to obtain $\mathcal{C}'$:

1. **(Com)**. Assume that the derivation is of the form:

$$\langle \{f(s_1, s_2) \triangleq_x f(t_1, t_2)\} \cup A_1; S; D; \theta \rangle \overset{Dec}{\Longrightarrow}$$
$$\langle A'; S; D; \theta\{x \mapsto f(x_1, x_2)\} \rangle$$

   where $A' = \{s_1 \triangleq_{x_1} t_2, s_2 \triangleq_{x_2} t_1\} \cup A_1$. By the induction hypothesis, for all different $\mathtt{a}, \mathtt{a}' \in A'$ the lemma holds, in particular for all $\mathtt{a}'' \in A'$ with $\mathtt{a}'' \neq x_i$,

$var(label(\mathtt{a}'')\theta_f) \cap var(x_i\theta_f) = \emptyset$, for $i \in \{1,2\}$. It remains to prove that for all $\mathtt{a}'' \in A_1$, $var(label(\mathtt{a}'')\theta_f) \cap var(x\theta_f) = \emptyset$. Observe that $x\theta_f = f(x_1\theta_f, x_2\theta_f)$, then we have that $var(x\theta_f) = var(x_1\theta_f) \cup var(x_2\theta_f)$ which implies that $var(label(\mathtt{a}'')\theta_f) \cap var(x\theta_f)$ is equal to $(var(label(\mathtt{a}'')\theta_f) \cap var(x_1\theta_f)) \cup (var(label(\mathtt{a}'')\theta_f) \cap var(x_1\theta_f))$, by induction hypothesis, each set in this union is empty. Hence, $var(label(\mathtt{a}'')\theta_f) \cap var(x\theta_f) = \emptyset$.

The analysis of the remaining rules is analogous to that in Lemma 4.1.16. □

Corollary 5.3.6. *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration and a final configuration $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}C\text{-}AUNIF^\ell}(\mathcal{C})$. Then we have that $\theta_f|_{labels(A)} \in \mathcal{G}^\ell_{\mathfrak{a}C}(\mathcal{C})$.*

Proof: The properties required to prove that $\theta_f|_{labels(A)}$ is a linear substitution generalization of $\mathcal{C}$ are obtained by Lemmas 5.3.4 and 5.3.5. □

We consider the `las` introduced in the Definition 4.1.18. From this definition, for any $\mathcal{C}$ and each $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}(\mathcal{C})$, we have a unique μ_{D_f} which produces a more specific linear generalization than θ_f. Indeed, since μ_{D_f} does not introduce variables, the soundness is obtained as a consequence of Corollary 5.3.4 and Lemma 5.3.5.

Theorem 5.3.7 (Soundness of $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}$)**.** *Let $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration, $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}C\text{-}AUNIF^\ell}(\mathcal{C})$, and μ_{D_f} the `las` of $\mathcal{C}_f$. Then, we have that $(\theta_f\mu_{D_f})|_{labels(A)} \in \mathcal{G}^\ell_{\mathfrak{a}C}(A)$.*

Example 5.3.8. Consider the terms $s = g(g(a,c),a)$ and $t = g(b,g(c,b))$ of the Example 5.1.5. Applying $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}$ to the initial linear configuration $\langle\{s \triangleq_x t\}; \emptyset; \emptyset; \iota\rangle$ we obtain the following final configurations:

$$\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}(\mathcal{C}) = \left\{ \begin{array}{l} \langle \emptyset; \{g(a,c) \triangleq_{y_1} b, a \triangleq_{y_2} g(c,b)\}; \emptyset; \theta_1\rangle, \\ \langle \emptyset; \{a \triangleq_{w_1} c, c \triangleq_{w_2} b, a \triangleq_z b\}; \emptyset; \theta_2\rangle, \\ \langle \emptyset; \{a \triangleq_{w_3} b, a \triangleq_z b\}; \emptyset; \theta_3\rangle \end{array} \right\},$$

where

$$\begin{aligned} \theta_1 &= \{x \mapsto g(y_1, y_2)\}, \\ \theta_2 &= \{x \mapsto g(g(w_1, w_2), z), y \mapsto g(w_1, w_2)\}, \\ \theta_3 &= \{x \mapsto g(g(w_3, c), z), y \mapsto g(w_3, c), w_4 \mapsto c\}. \end{aligned}$$

Since all the delayed sets are empty then the computed generalizations are $x\theta_1 = g(y_1, y_2)$, $x\theta_2 = g(g(w_1, w_2), z)$, and $x\theta_3 = g(g(w_3, c), z)$. Notice that all the generalizations are linear terms and the term $x\theta_3$ was not computed by $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}}$ in Example 5.1.5. ◇

Example 5.3.9. Let ε_f and $f(a,a)$ be terms, where f is an $\mathfrak{a}$C-symbol. Applying $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}^\ell}}$ and the $\texttt{las}$ μ, we get two linear $\mathfrak{a}$C-generalizations $f(x,a)$ and $f(a,y)$. $\diamond$

Theorem 5.3.10 (Completeness of $\mathcal{A}_{\mathfrak{a}\text{C-AU}_{\text{NIF}^\ell}}$)**.** *Consider $\mathcal{C} = \langle A; S; D; \theta\rangle$ be a linear configuration and $\gamma \in \mathcal{G}^\ell_{\mathfrak{a}C}(A)$. Then, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f\rangle \in \mathcal{A}_{\mathfrak{a}C\text{-AU}_{\text{NIF}^\ell}}(\mathcal{C})$ and μ_{D_f} the $\texttt{las}$ of $\mathcal{C}_f$ such that for all $x \in labels(A)$ we have $x\gamma \preceq_{\mathfrak{a}C} x\theta_f\mu_{D_f}$.*

Proof: We use induction on $size(A)$.

Base Case

If $size(A) = 0$ is because $\mathcal{C}$ is a final linear configuration and the theorem holds vacuously.

Induction Step

We assume that the statement holds for configurations with $size(A) < n$, $n > 2$, and we show for configurations with $size(A) = n$. The proof consists of analyzing the cases of rule application depending on the form of an AUT in $\mathcal{C}$ and the transition $\mathcal{C} \Longrightarrow \mathcal{C}'$. We can assume w.l.o.g. that variables in the range of γ and the variables used in the algorithm are disjoint. Consider that the transition is of the form

$$\langle \{s \stackrel{\triangleq}{=}_x t\} \cup A_1; S; D; \theta\rangle \Longrightarrow \mathcal{C}'$$

1. If $head(s) = head(t)$ and $head(s)$ is not a C-symbol then we have two cases covered by the rule **(Dec)**.

   (a) If $head(s) \neq \varepsilon_f$, then the derivation is of the form:
   $$\langle \{g(s_1,\ldots,s_m) \stackrel{\triangleq}{=}_x g(t_1,\ldots,t_m)\} \cup A_1; S; D; \theta\rangle \stackrel{Dec}{\Longrightarrow} \langle A'; S; D; \theta\{x \mapsto g(x_1,\ldots,x_m)\}\rangle,$$
   where $g \neq \varepsilon_f$ and $A' = \{s_1 \stackrel{\triangleq}{=}_{x_1} t_1, \ldots, s_m \stackrel{\triangleq}{=}_{x_m} t_m\} \cup A_1$ with $m \geq 0$. The analysis is analogous to case 1.(a) of Theorem 4.2.2.

   (b) If $head(s) = \varepsilon_f$, then the derivation is of the form:
   $$\langle \{\varepsilon_f \stackrel{\triangleq}{=}_x \varepsilon_f\} \cup A'; S; D; \theta\rangle \stackrel{Dec}{\Longrightarrow} \langle A'; S; D; \theta\{x \mapsto \varepsilon_f\}\rangle$$
   The analysis is analogous to the case 1.(b) of Theorem 4.2.2.

2. If $head(s) = head(t)$ and $head(s)$ is a C-symbol or $\mathfrak{a}$C-symbol, say $head(s) = f$. Assume that we have the derivations of the form:

   $$\begin{array}{ccc} & \langle \{f(s_1,s_2) \stackrel{\triangleq}{=}_x f(t_1,t_2)\} \cup A'; S; D; \theta\rangle & \\ \stackrel{\text{(Dec)}}{\swarrow\!\!\swarrow} & & \searrow\!\!\searrow^{\text{(Com)}} \\ \mathcal{C}_1 = \langle A_1; S; D; \theta\{x \mapsto f(x_1,x_2)\}\rangle & & \langle A_2; S; D; \theta\{x \mapsto f(x_3,x_4)\}\rangle = \mathcal{C}_2 \end{array}$$

   where $A_1 = \{s_1 \stackrel{\triangleq}{=}_{x_1} t_1, s_2 \stackrel{\triangleq}{=}_{x_2} t_2\} \cup A'$ and $A_2 = \{s_1 \stackrel{\triangleq}{=}_{x_3} t_2, s_2 \stackrel{\triangleq}{=}_{x_4} t_1\} \cup A'$. We have to analyze the following three subcases:

(a) If $x\gamma = f(r_1, r_2)$, where $r_1 \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(s_1, t_1)$ and $r_2 \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(s_2, t_2)$. We apply the rule **(Dec)** to $\mathcal{C}$ and the analysis is analogous to case 1.(a) i. of Theorem 4.2.2.

(b) If $x\gamma = f(r_3, r_4)$, where $r_3 \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(s_1, t_2)$ and $r_4 \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(s_2, t_1)$. We apply the rule **(Com)** to $\mathcal{C}$. We build a linear substitution generalization γ' for $\mathcal{C}_2$. Consider γ' as follows:

$$\gamma'(y) = \begin{cases} \gamma(y) & \text{if } y \in \mathit{labels}(A'), \\ r_i & \text{if } y = x_i, i \in \{3, 4\}. \end{cases}$$

Notice that $\gamma' \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(A_2)$. Indeed, $\mathit{dom}(\gamma')$ is the set of active labels in $\mathcal{C}''$. Also, if $y \in \mathit{labels}(A')$ is the label of an AUT $\mathtt{a}$, $y\gamma'\rho_l = y\gamma\rho_l \approx_{\mathfrak{a}\mathrm{C}} \mathit{lhs}(\mathtt{a})$. Similarly, for $\gamma'\rho_r$. Otherwise, $x_i\gamma'\rho_l = r_i\rho_l = (x\gamma\rho_l)|_{(i-2)} \approx_{\mathfrak{a}\mathrm{C}} r_i$, for $i \in \{3, 4\}$. Additionally, since $\gamma \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(A)$, we have that $\mathit{var}(x\gamma) \cap \mathit{var}(\mathit{label}(\mathtt{a})\gamma') = \emptyset$ for all different $\mathtt{a} \in A'$ and as $x\gamma = f(x_3\gamma', x_4\gamma')$ we have that $\mathit{var}(x_i\gamma') \cap \mathit{var}(\mathit{label}(\mathtt{a})\gamma') = \emptyset$ for $i \in \{3, 4\}$ and for all $\mathtt{a} \in A'$. Therefore, $\gamma' \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(A_2)$. By induction hypothesis, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^{\ell}}(\mathcal{C}') \subseteq \mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^{\ell}}(\mathcal{C})$ and the $\mathtt{las}$ μ_{D_f} such that $\mathit{label}(\mathtt{a})\gamma' \preceq_{\mathfrak{a}\mathrm{C}} x\theta_f\mu_{D_f}$ for all $\mathtt{a} \in A_2$. In particular, $x_i\gamma' \preceq_{\mathfrak{a}\mathrm{C}} x_i\theta_f\mu_{D_f}$, for $i \in \{3, 4\}$. It remains to prove that $x\gamma \preceq_{\mathfrak{a}\mathrm{C}} x\theta_f\mu_{D_f}$ holds. Observe that $x\gamma = f(x_3\gamma', x_4\gamma') \preceq_{\mathfrak{a}\mathrm{C}} f(x_3, x_4)\theta_f\mu_{D_f} = x\theta_f\mu_{D_f}$. Hence, for all $y \in \mathit{labels}(A)$ we got that $y\gamma \preceq_{\mathfrak{a}\mathrm{C}} y\theta_f\mu_{D_f}$ and from Lemma 5.3.2, this a linear E-generalization.

(c) If $x\gamma$ is a variable, say z. The analysis is analogous to case 1.(a) ii of Theorem 4.2.2.

3. If $\mathit{head}(s) \neq \mathit{head}(t)$ and they are not related absorptive symbols. Since the AUT $s \triangleq_x t$ is solved, then we apply the rule **(Sol)** and consider the following derivation:

$$\langle \{s \triangleq_x t\} \cup A'; S; D; \theta \rangle \stackrel{Sol}{\Longrightarrow} \langle A'; \{s \triangleq_x t\} \cup S; D; \theta \rangle,$$

Let $\gamma' = \gamma|_{\mathit{labels}(A')}$. Then, $\gamma' \in \mathcal{G}^{\ell}_{\mathfrak{a}\mathrm{C}}(A')$. In fact, for any $\mathtt{a} \in A'$, $\mathit{label}(\mathtt{a})\gamma'\rho_l = \mathit{label}(\mathtt{a})\gamma\rho_l \approx_{\mathfrak{a}} \mathit{lhs}(\mathtt{a})$. Similarly, for ρ_r and $\mathit{rhs}(\mathtt{a})$.

By induction hypothesis, there exist $\mathcal{C}_f = \langle \emptyset; S_f; D_f; \theta_f \rangle \in \mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}(\mathcal{C})$ and μ_{D_f} the abstraction substitution of $\mathcal{C}_f$ such that for all $y \in \mathit{labels}(A_f)$ we have $y\gamma' \preceq_{\mathfrak{a}\mathrm{C}} y\theta_f\mu_{D_f}$. It remains to prove that $x\gamma \preceq_{\mathfrak{a}\mathrm{C}} x\theta_f\mu_{D_f}$. Notice that $x\gamma$ should be a variable, say x', since it labels a solved equation. Without loss of generality, we assume that this variable does not belong to the variables appearing in the final configuration $\mathcal{C}_f$ and then $x\gamma = x' \preceq_{\mathfrak{a}\mathrm{C}} x = x\theta_f\mu_{D_f}$, implying that the theorem holds for this case.

4. If $\mathit{head}(s) \neq \mathit{head}(t)$ and they are related absorptive symbols, we have to consider two subcases:

(a) If $head(s) = \varepsilon_f$ and $head(t) = f$ with f $\mathfrak{a}$- or $\mathfrak{a}$C-symbol, then we can apply the rules **(ExpLA1)** and **(ExpLA2)**. Assume we have the derivations of the form:

$$\begin{array}{c} \langle \{f(s,t) \triangleq_x \varepsilon_f\} \cup A'; S; D; \theta\rangle \\ \swarrow\!\!\swarrow \text{(ExpRA1)} \qquad \text{(ExpRA2)} \searrow\!\!\searrow \\ \mathcal{C}_1 = \langle A_1; S; D_1; \theta\{x \mapsto f(x_1, x_2)\}\rangle \;\; \langle A_2; S; D_2; \theta\{x \mapsto f(x_3, x_4)\}\rangle = \mathcal{C}_2 \end{array}$$

where $A_1 = \{s \triangleq_{x_1} \varepsilon_f\} \cup A'$, $D_1 = \{t \triangleq_{x_2} \star\} \cup D$, $A_2 = \{t \triangleq_{x_4} \varepsilon_f\} \cup A'$, and $D_2 = \{s \triangleq_{x_3} \star\} \cup D$. The set $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}(\mathcal{C}) = \mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}(\mathcal{C}_1) \cup \mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}(\mathcal{C}_2)$. Also, $x\gamma$ is a term of the form $f(u, v)$ or a variable, say x'. We need to consider both cases. The analysis is analogous to item 3 of Theorem 4.2.2.

(b) If $head(s) = f$ and $head(t) = \varepsilon_f$ with f $\mathfrak{a}$- or $\mathfrak{a}$C-symbol, then we can apply the rules **(ExpRA1)** and **(ExpRA2)**. Assume we have the derivations of the form:

$$\begin{array}{c} \langle \{\varepsilon_f \triangleq_x f(s,t)\} \cup A'; S; D; \theta\rangle \\ \swarrow\!\!\swarrow \text{(ExpLA1)} \qquad \text{(ExpLA2)} \searrow\!\!\searrow \\ \mathcal{C}_1 = \langle A_1; S; D_1; \theta\{x \mapsto f(x_1, x_2)\}\rangle \;\; \langle A_2; S; D_2; \theta\{x \mapsto f(x_3, x_4)\}\rangle = \mathcal{C}_2 \end{array}$$

where $A_1 = \{\varepsilon_f \triangleq_{x_1} s\} \cup A'$, $D_1 = \{\star \triangleq_{x_2} t\} \cup D$, $A_2 = \{\varepsilon_f \triangleq_{x_4} t\} \cup A'$, and $D_2 = \{\star \triangleq_{x_3} s\} \cup D$. The set $\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}(\mathcal{C}) = \mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}(\mathcal{C}_1) \cup \mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}(\mathcal{C}_2)$. Also, $x\gamma$ is either a term of the form $f(u, v)$ or a variable, say x'. We need to consider both cases. The analysis is analogous to item 3 of Theorem 4.2.2.

As all the cases for s and t were considered, the theorem holds. □

Corollary 5.3.11 (Type of Linear Absorptive Commutative Anti-Unification Problem)**.** *The linear variant of anti-unification modulo $\mathfrak{a}C$ is of type finitary.*

Definition 5.3.12. Let s and t be terms and $\mathcal{C} = \langle \{s \triangleq_x t\}; \emptyset; \emptyset; \iota\rangle$ be a configuration. We define the complete set of computed linear $\mathfrak{a}$C-generalizations as

$$\mathfrak{C}_{\mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}}(s,t) = \{x\theta\mu_D \mid \langle \emptyset; S; D; \theta\rangle \in \mathcal{A}_{\mathfrak{a}\text{C-AUNIF}^\ell}(\mathcal{C}) \text{ and } \mu_D \texttt{ las of } \langle \emptyset; S; D; \theta\rangle\}.$$

Since we already have a complete set of finite linear generalizations, the next step is to determine the minimal complete set of generalizations. Observe that in Example 5.3.8, the computed linear $\mathfrak{a}$C-generalizations are comparable, $g(y_1, y_2) \preceq_{\mathfrak{a}\text{C}} g(g(w_1, w_2), z) \preceq_{\mathfrak{a}\text{C}} g(g(w_3, c), z)$, while in Example 5.3.9 we have $f(a, x) \approx_{\mathfrak{a}\text{C}} f(y, a)$.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we develop the formal foundations of anti-unification in absorptive theories and propose four sound algorithms that compute generalizations for pure absorptive theories, for absorptive commutative theories, and for their respective linear variants, for which solutions are restricted to linear generalizations. The two algorithms for linear variants are shown to be complete, while in the pure absorptive setting, the procedure is capable of producing minimal complete sets of generalizations. For the linear variant of the absorptive commutative case, we obtain a complete set of generalizations and discuss how this can be minimized.

The main contributions of this work are summarized below.

1. We introduced a sound algorithm for computing generalizations in pure absorptive theories. The algorithm is defined through a set of inference rules that are applied to configurations, a structured representation of the anti-unification problem. From these, a set of final configurations is derived, which provides the necessary information for computing generalizations using a method that involves building a so-called "abstraction set" from each final configuration. To systematically obtain all possible abstractions, we presented a grammatical technique. By combining these abstractions with the final configurations, the corresponding generalizations are computed. Notably, in certain cases, the use of abstractions may give rise to an infinite set of incomparable generalizations, indicating that the general problem is at least infinitary.

2. We critically analyzed the completeness approach from previous work [9], identifying flaws and limitations. We proposed a refinement method that is believed to complete the approach in that work. The method computes additional generalizations that

are essential towards achieving completeness. We establish the correct completeness statement and discuss an inductive approach to obtain a proof. However, some points in the proposed analysis remain to be finalized.

3. We developed the algorithm $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}^{\ell}}$, for computing linear generalizations in absorptive theories. Linear generalizations do not allow variable repetitions. The algorithm is sound, and together with the linear abstraction substitution achieves completeness, computing minimal complete sets of linear generalizations. This establishes that the linear absorptive anti-unification problem is of finitary type, since the algorithm generates a finite set of final configurations and each configuration allows only a unique linear substitution, resulting in a finite number of computed linear generalizations.

4. We studied combinations of absorptive and commutative symbols, introducing algorithms capable of computing generalizations and linear generalizations in this broader setting. Soundness and completeness for the linear variant were established, demonstrating the effectiveness of our approach in algebraic contexts that are more expressive than pure absorptive theory. The linear absorptive commutative anti-unification problem is likewise finitary.

Overall, this thesis contributes both theoretical insights and practical algorithms for anti-unification in absorptive and absorptive commutative theories, including the linear variants. The results demonstrate that anti-unification in these settings is a rich and challenging problem, motivating further research in its formal and computational aspects.

6.2 Future Work

We refined the approach to construct the abstraction set from final configurations in [9], by incorporating the notion of *extended store* and *extended set of abstraction substitutions*. The discussed completeness statement analysis presents difficulties in the base case, where it remains necessary to establish the completeness of the extended set of abstraction substitutions built from final configurations. An immediate and natural direction for future work, therefore, is to complete the completeness analysis for the refined algorithm. Once completeness is achieved, it will be possible to formalize the minimization of the *computed extended generalizations*, which would allow us to determine that the anti-unification problem in absorptive theories is indeed *infinitary*.

If completeness for pure absorptive anti-unification can be achieved, similar methods could immediately be applied to analyze the case of absorptive commutative anti-

unification. This extension is feasible, since the behavior of the combined setting resembles that of the linear absorptive case, for which completeness has already been established.

Another important direction is the study of efficient algorithms for absorptive anti-unification. Since the proposed approach is not focused on efficiency, there are plenty of opportunities for improvement. Indeed, the computation of final configurations relies on the application of (branching) expansion rules, which give rise to an exponential number of final configurations. Additionally, the construction of the extended store also introduces the exploration of exponential combinations. A deeper analysis of this aspect would clarify the computational cost of the procedure and contribute to its practical applicability.

An interesting research direction is to investigate absorptive anti-unification in richer equational theories. In particular, analyzing the problem in the presence of associative, absorptive, and absorptive–associative symbols, as well as combinations involving commutativity, associativity, and absorption within the same theory, would open new perspectives and bring the theory closer to more general and practical applications.

Finally, a promising direction for future work is the formalization of absorptive anti-unification in a proof assistant. While several successful formalizations of unification and matching have been developed — such as Contejean's work on AC-matching in Coq [28], and the series of formalizations developed in PVS by Ayala-Rincón and collaborators for nominal unification [5], nominal C-unification [8], nominal AC-matching [6], and AC-unification [7], as well as the formalization in Coq by Ayala-Rincón et al. [3] for nominal C-matching through unification with protected variables. To date, the only known formalization of an anti-unification algorithm is the syntactic case developed by Ayala-Rincón et al. [4], which received an Honourable Mention at the 17th NASA Formal Methods Symposium (NFM 2025). Building on this syntactic foundation, the refined algorithms and correctness results for absorptive anti-unification presented in this thesis could be formalized within a similar framework. Such a development would provide machine-checked guarantees of soundness and, eventually, completeness, while laying the groundwork for further extensions to richer equational theories.

Bibliography

[1] Alpuente, M., Escobar, S., Espert, J., and Meseguer, J. A modular order-sorted equational generalization algorithm. *Information and Computation* vol. 235 (2014), pp. 98–136. https://doi.org/10.1016/j.ic.2014.01.006. v, 2, 20, 21, 23, 24, 25

[2] Alpuente, M., Escobar, S., Espert, J., and Meseguer, J. Order-sorted equational generalization algorithm revisited. *Annals of Mathematics and Artificial Intelligence vol. 90*, 5 (2022), pp. 499–522. https://doi.org/10.1007/s10472-021-09771-1. v, 2, 20

[3] Ayala-Rincón, M., de Carvalho Segundo, W., Fernández, M., and Nantes-Sobrinho, D. A formalisation of nominal C-matching through unification with protected variables. In *Proceedings of the 13th Workshop on Logical and Semantic Frameworks with Applications, LSFA* (2018), vol. 344 of *Electronic Notes in Theoretical Computer Science*, pp. 47–65. https://doi.org/10.1016/j.entcs.2019.07.004. 112

[4] Ayala-Rincón, M., de Lima, T. A., Lima, M. J. D., Moscato, M. M., and Kutsia, T. Verification of an anti-unification algorithm in pvs. In *NASA Formal Methods, NFM* (2025), vol. 15682 of *Lecture Notes in Computer Science*, Springer, pp. 54–71. https://doi.org/10.1007/978-3-031-93706-4_4. 112

[5] Ayala-Rincón, M., Fernández, M., and Oliveira, A. C. R. Completeness in PVS of a nominal unification algorithm. In *Proceedings of the 10th Workshop on Logical and Semantic Frameworks, with Applications, LSFA 2015* (2016), vol. 323 of *Electronic Notes in Theoretical Computer Science*, pp. 57–74. https://doi.org/10.1016/j.entcs.2016.06.005. 112

[6] Ayala-Rincón, M., Fernández, M., Silva, G. F., Kutsia, T., and Nantes-Sobrinho, D. Nominal AC-matching. In *Intelligent Computer Mathematics - 16th International Conference, CICM 2023* (2023), C. Dubois and M. Kerber, Eds., vol. 14101 of *Lecture Notes in Computer Science*, Springer, pp. 53–68. https://doi.org/10.1007/978-3-031-42753-4_4. 112

[7] Ayala-Rincón, M., Fernández, M., Silva, G. F., Kutsia, T., and Nantes-Sobrinho, D. Certified First-Order AC-Unification and Applications. *Journal of Automated Reasoning 68*, 4 (2024), 25. https://doi.org/10.1007/s10817-024-09714-5. 112

[8] Ayala-Rincón, M., Fernández, M., Silva, G. F., and Nantes-Sobrinho, D. A certified functional nominal C-unification algorithm. In *Logic-Based Program Synthesis and Transformation - 29th International Symposium, LOPSTR 2019* (2020), vol. 12042 of *Lecture Notes in Computer Science*, Springer, pp. 123–138. https://doi.org/10.1007/978-3-030-45260-5_8. 112

[9] Ayala-Rincón, M., Cerna, D. M., Barragán, A. F. G., and Kutsia, T. Equational anti-unification over absorption theories. In *12th International Joint Conference on Automated Reasoning IJCAR* (2024), vol. 14740 of *LNAI of LNCS*, Springer, pp. 317–337. https://doi.org/10.1007/978-3-031-63501-4_17. vi, 3, 59, 60, 61, 110, 111

[10] Aït-Kaci, H., and Pasi, G. Fuzzy lattice operations on first-order terms over signatures with similar constructors: A constraint-based approach. *Fuzzy Sets and Systems,* vol. 391, (2020), pp. 1–46. https://doi.org/10.1016/j.fss.2019.03.019. v, 2

[11] Baader, F. Unification, weak unification, upper bound, lower bound and generalization problems. In *Proceedings of the 4th International Conference on Rewriting Techniques and Applications (RTA'91)* (1991), vol. 488 of *Lecture Notes in Computer Science*, Springer, pp. 86–97. https://doi.org/10.1007/3-540-53904-2_88. 19

[12] Baader, F., and Nipkow, T. *Term Rewriting and All That.* Cambridge University Press, 1998. https://doi.org/10.1017/CBO9781139172752. 5, 12, 36

[13] Bader, J., Scott, A., Pradel, M., and Chandra, S. Getafix: learning to fix bugs automatically. *Proceedings of the ACM on Programming Languages,* vol. 3, OOPSLA (2019), pp. 159:1–159:27. https://doi.org/10.1145/3360585. iv, 1

[14] Barwell, A. D., Brown, C., and Hammond, K. Finding parallel functional pearls: Automatic parallel recursion scheme detection in Haskell functions via anti-unification. *Future Generation Computer System,* vol. 79 (2018), pp. 669–686. https://doi.org/10.1016/j.future.2017.07.024. iv, 1

[15] Baumgartner, A. *Anti-Unification Algorithms: Design, Analysis, and Implementation.* PhD thesis, RISC, JKU Linz, 2015. https://epub.jku.at/obvulihs/content/titleinfo/818923. v, 2

[16] Baumgartner, A., and Kutsia, T. Unranked second-order anti-unification. In *Information and Computation* (2017), vol. 255, pp. 262–286. https://doi.org/10.1016/j.ic.2017.01.005. v, 2

[17] Baumgartner, A., Kutsia, T., Levy, J., and Villaret, M. Nominal anti-unification. In *26th International Conference on Rewriting Techniques and Applications, RTA* (2015), vol. 36 of *LIPIcs*, pp. 57–73. https://doi.org/10.4230/LIPIcs.RTA.2015.57. v, 2

[18] Baumgartner, A., Kutsia, T., Levy, J., and Villaret, M. Higher-order pattern anti-unification in linear time. *Journal of Automated Reasoning,* vol. 58, 2 (2017), pp. 293–310. https://doi.org/10.1007/s10817-016-9383-3. v, 2

[19] Baumgartner, A., Kutsia, T., Levy, J., and Villaret, M. Term-graph anti-unification. In *3rd International Conference on Formal Structures for Computation and Deduction, FSCD* (2018), vol. 108 of *LIPIcs*, pp. 9:1–9:17. https://doi.org/10.4230/LIPIcs.FSCD.2018.9. v, 2

[20] Bulychev, P. E., Kostylev, E. V., and Zakharov, V. A. Anti-unification algorithms and their applications in program analysis. In *Perspectives of Systems Informatics, PSI* (2009), vol. 5947 of *LNCS*, Springer, p. 413–423. https://doi.org/10.1007/978-3-642-11486-1_35. iv, 1

[21] Burghardt, J. E-generalization using grammars. *Artif. Intell. 165*, 1 (2005), 1–35. https://doi.org/10.1016/j.artint.2005.01.008. v, 2

[22] Cao, D., Kunkel, R., Nandi, C., Willsey, M., Tatlock, Z., and Polikarpova, N. Babble: Learning better abstractions with e-graphs and anti-unification. *Proceedings of the ACM on Programming Languages,* vol. 7, POPL (2023), pp. 396–424. https://doi.org/10.1145/3571207. iv, 1

[23] Cerna, D. M. Anti-unification and the theory of semirings. *Theoretical Computer Science,* vol. 848, (2020), pp. 133–139. https://doi.org/10.1016/j.tcs.2020.10.020. v, 2, 20, 29

[24] Cerna, D. M., and Kutsia, T. Higher-order equational pattern anti-unification. In *3rd International Conference on Formal Structures for Computation and Deduction, FSCD* (2018), vol. 108 of *LIPIcs*, pp. 12:1–12:17. https://doi.org/10.4230/LIPIcs.FSCD.2018.12. v, 2

[25] Cerna, D. M., and Kutsia, T. Idempotent anti-unification. *ACM Transactions on Computational Logic,* vol. 21, 2 (2020), pp. 10:1–10:32. https://doi.org/10.1145/3359060. v, 2, 20

[26] Cerna, D. M., and Kutsia, T. Unital anti-unification: Type and algorithms. In *5th International Conference on Formal Structures for Computation and Deduction, FSCD* (2020), vol. 167 of *LIPIcs*, pp. 26:1–26:20. https://doi.org/10.4230/LIPICS.FSCD.2020.26. v, 2, 20, 25, 26, 27, 28

[27] Cerna, D. M., and Kutsia, T. Anti-unification and generalization: A survey. In *32nd International Joint Conference on Artificial Intelligence, IJCAI* (2023), pp. 6563–6573. https://doi.org/10.24963/ijcai.2023/736. 11

[28] Contejean, E. A certified AC matching algorithm. In *Rewriting Techniques and Applications, 15th International Conference, RTA 2004* (2004), vol. 3091 of *Lecture Notes in Computer Science*, Springer, pp. 70–84. https://doi.org/10.1007/978-3-540-25979-4_5. 112

[29] Cropper, A., Dumancic, S., Evans, R., and Muggleton, S. H. Inductive logic programming at 30. *Machine Learning,* vol. 111, 1 (2022), pp. 147–172. https://doi.org/10.1007/s10994-021-06089-1. iv, 1

[30] Gulwani, S. Programming by examples - and its applications in data wrangling. In *NATO Science for Peace and Security Series*, vol. 45. IOS Press, (20160, pp. 137 – 158. https://doi.org/10.3233/978-1-61499-627-9-137. iv, 1

[31] Huet, G. P. Résolution d'équations dans les langages d'ordre $1, 2, \ldots, \omega$. *Théorème Automatique de Résolution 1* (1976), 27–57. https://gallium.inria.fr/~huet/PUBLIC/Huet1976.pdf. 19

[32] Jain, N., Gandhi, S., Sonwane, A., Kanade, A., Natarajan, N., Parthasarathy, S., Rajamani, S. K., and Sharma, R. Staticfixer: From static analysis to static repair. *CoRR abs/2307.12465* (2023). https://doi.org/10.4230/10.48550/arXiv.2307.12465. iv, 1

[33] Krumnack, U., Schwering, A., Gust, H., and Kühnberger, K. Restricted higher-order anti-unification for analogy making. In *20th Australian Joint Conference on Artificial Intelligence, AI* (2007), vol. 4830 of *LNCS*, Springer, pp. 273–282. https://doi.org/10.1007/978-3-540-76928-6_29. v, 2

[34] Kutsia, T., Levy, J., and Villaret, M. Anti-unification for unranked terms and hedges. *Journal of Automated Reasoning,* vol. 52, 2 (2014), pp. 155–190. https://doi.org/10.1007/s10817-013-9285-6. v, 2

[35] Kutsia, T., and Pau, C. Matching and generalization modulo poximity and tolerance relations. In *13th International Tbilisi Symposium on Logic, Language and Computation, TbiLLC* (2019), vol. 13206 of *LNCS*, Springer, pp. 323—-342. https://doi.org/10.1007/978-3-030-98479-3_16. v, 2

[36] Kutsia, T., and Pau, C. A framework for approximate generalization in quantitative theories. In *International Joint Conference on Automated Reasoning, IJCAR* (2022), vol. 13385 of *LNCS*, Springer, pp. 578—-596. https://doi.org/10.1007/978-3-031-10769-6_34. v, 2

[37] Mehta, S., Bhagwan, R., Kumar, R., Bansal, C., Maddila, C. S., Ashok, B., Asthana, S., Bird, C., and Kumar, A. Rex: Preventing bugs and misconfiguration in large services using correlated change analysis. In *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020* (2020), USENIX Association, pp. 435–448. https://dl.acm.org/doi/10.5555/3388242.3388274. iv, 1

[38] Pau, C. *Symbolic Techniques for Approximate Reasoning.* PhD thesis, Research Institute for Symbolic Computation, Johannes Kepler University Linz, 2022. https://www3.risc.jku.at/publications/download/risc_6513/thesis.pdf. v, 2

[39] Plotkin, G. D. A note on inductive generalization. *Machine Intell.* vol. 5, 1 (1970), pp. 153–163. v, 2, 19, 20, 21

[40] Popplestone, R. J. An experiment in automatic induction. *Machine Intell.* vol. 5, 1 (1970), pp. 203–215. 21

[41] Reynolds, J. C. Transformational systems and the algebraic structure of atomic formulas. *Machine Intell.* vol. 5, 1 (1970), pp. 135–151. v, 2, 19, 20, 21

[42] Schmidt-Schauss, M., and Nantes-Sobrinho, D. Nominal anti-unification with atom-variables. In *7th International Conference on Formal Structures for Computation and Deduction, FSCD* (2022), vol. 228 of *LIPIcs*, pp. 7:1–7:22. https://doi.org/10.4230/LIPIcs.FSCD.2022.7. v, 2

[43] Schmidt-Schauss, M., and Nantes-Sobrinho, D. Towards fast nominal anti-unification of letrec-expressions. In *29th International Conference on Automated Deduction, CADE* (2023), vol. 14132 of *LNCS*, Springer, pp. 456–473. https://doi.org/10.1007/978-3-031-38499-8_26. v, 2

[44] Sousa, R. R., Soares, G., Gheyi, R., Barik, T., and D'Antoni, L. Learning quick fixes from code repositories. In *35th Simpósio Brasileiro de Engenharia de Software, SBES* (2021), ACM. https://doi.org/10.1145/3474624.3474650. iv, 1

[45] Winter, E. R., Nowack, V., Bowes, D., Counsell, S., Hall, T., Haraldsson, S. Ó., Woodward, J. R., Kirbas, S., Windels, E., McBello, O., Atakishiyev, A., Kells, K., and Pagano, M. W. Towards developer-centered automatic program repair: findings from Bloomberg. In *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE* (2022), ACM. https://doi.org/10.1145/3540250.3558953. iv, 1

Appendix A

Extended Computed Generalizations of the Terms $g(f(a, g(a, c)), \varepsilon_f)$ and $g(\varepsilon_f, f(g(c, c), b))$

We present in detail the steps for computing all the generalizations obtained with $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ and the extended set of abstraction substitutions for the terms $g(f(a, g(a, c)), \varepsilon_f)$ and $g(\varepsilon_f, f(g(c, c), b))$. The associated initial configuration is

$$\langle \{g(f(a, g(a, c)), \varepsilon_f) \triangleq_x g(\varepsilon_f, f(g(c, c), b))\}; \emptyset; \emptyset; \iota \rangle.$$

$$\langle \{g(f(a, g(a, c)), f(f(a, g(a, c)), f(a, g(a, c)))) \triangleq_x g(\varepsilon_f, \varepsilon_f)\}; \emptyset; \emptyset; \iota \rangle.$$

Applying $\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}$ to this configuration, we obtain the following four final configurations:

$$\mathcal{A}_{\mathfrak{a}\text{-AUNIF}}(\mathcal{C}) = \left\{ \begin{array}{l} \langle \emptyset; \{a \triangleq_{y_1} \varepsilon_f, \varepsilon_f \triangleq_{z_1} g(c, c)\}; \{g(a, c) \triangleq_{y_2} \star, \star \triangleq_{z_2} b\}; \theta_1 \rangle, \\ \langle \emptyset; \{a \triangleq_{y_1} \varepsilon_f, \varepsilon_f \triangleq_{z_4} b\}; \{g(a, c) \triangleq_{y_2} \star, \star \triangleq_{z_3} g(c, c)\}; \theta_2 \rangle, \\ \langle \emptyset; \{g(a, c) \triangleq_{y_4} \varepsilon_f, \varepsilon_f \triangleq_{z_5} g(c, c)\}; \{a \triangleq_{y_3} \star, \star \triangleq_{z_6} b\}; \theta_3 \rangle, \\ \langle \emptyset; \{g(a, c) \triangleq_{y_4} \varepsilon_f, \varepsilon_f \triangleq_{z_8} b\}; \{a \triangleq_{y_3} \star, \star \triangleq_{z_7} g(c, c)\}; \theta_4 \rangle, \end{array} \right\},$$

where

$$\begin{aligned} \theta_1 &= \{x \mapsto g(f(y_1, y_2), f(z_1, z_2)), \ldots\}, \\ \theta_2 &= \{x \mapsto g(f(y_1, y_2), f(z_3, z_4)), \ldots\}, \\ \theta_3 &= \{x \mapsto g(f(y_3, y_4), f(z_5, z_6)), \ldots\}, \\ \theta_4 &= \{x \mapsto g(f(y_3, y_4), f(z_7, z_8)), \ldots\}. \end{aligned}$$

Each of the final configurations above is denoted by $\mathcal{C}_i$, for $i = 1, \ldots, 4$. Moreover, we denote their final store and delayed sets by S_i and D_i, respectively. The extended stores for all four final configurations are below.

$$
\begin{aligned}
S_{\mathcal{C}_1} &= \{a \triangleq_{y_1} \varepsilon_f, \varepsilon_f \triangleq_{z_1} g(c,c)\} \cup \{g(a,c) \triangleq_{x_1} b, a \triangleq_{x_2} b, c \triangleq_{x_3} b\},\\
S_{\mathcal{C}_2} &= \{a \triangleq_{y_1} \varepsilon_f, \varepsilon_f \triangleq_{z_4} b\} \cup \{g(a,c) \triangleq_{x_4} c, a \triangleq_{x_5} g(c,c), a \triangleq_{x_6} c, c \triangleq_{x_7} g(c,c)\},\\
S_{\mathcal{C}_3} &= \{g(a,c) \triangleq_{y_4} \varepsilon_f, \varepsilon_f \triangleq_{z_5} g(c,c)\} \cup \{a \triangleq_{x_8} b\},\\
S_{\mathcal{C}_4} &= \{g(a,c) \triangleq_{y_4} \varepsilon_f, \varepsilon_f \triangleq_{z_8} b\} \cup \{a \triangleq_{x_9} g(c,c), a \triangleq_{x_{10}} c\}.
\end{aligned}
$$

Now, we analyze the set of abstraction substitutions for each final configuration.

-**Final Configuration $\mathcal{C}_1$.** The extended abstraction sets are:

$$
\begin{aligned}
\uparrow_{y_2}(D_1, S_{\mathcal{C}_1}) &= \uparrow(g(a,c), \sigma^l_{S_{\mathcal{C}_1}})\\
&= \{x_1, g(a,c), g(y_1,c), g(x_2,c), g(x_2,x_3), g(a,x_3), g(y_1,x_3)\},\\
\uparrow_{z_2}(D_1, S_{\mathcal{C}_1}) &= \uparrow(b, \sigma^r_{S_{\mathcal{C}_1}}) = \{b, x_1, x_2, x_3\}.
\end{aligned}
$$

The 28 set of extended abstraction substitutions, $\Psi(D_1, S_{\mathcal{C}_1})$, is given below.

$$
\left\{
\begin{array}{lll}
\{y_2 \mapsto x_1,\ z_2 \mapsto b\}, & \{y_2 \mapsto x_1,\ z_2 \mapsto x_1\}, & \{y_2 \mapsto x_1,\ z_2 \mapsto x_2\},\\
\{y_2 \mapsto x_1,\ z_2 \mapsto x_3\}, & \{y_2 \mapsto g(a,c),\ z_2 \mapsto b\}, & \{y_2 \mapsto g(a,c),\ z_2 \mapsto x_1\},\\
\{y_2 \mapsto g(a,c),\ z_2 \mapsto x_2\}, & \{y_2 \mapsto g(a,c),\ z_2 \mapsto x_3\}, & \{y_2 \mapsto g(x_2,c),\ z_2 \mapsto b\},\\
\{y_2 \mapsto g(x_2,c),\ z_2 \mapsto x_1\}, & \{y_2 \mapsto g(a,x_3),\ z_2 \mapsto b\}, & \{y_2 \mapsto g(a,x_3),\ z_2 \mapsto x_1\},\\
\{y_2 \mapsto g(y_1,c),\ z_2 \mapsto b\}, & \{y_2 \mapsto g(y_1,c),\ z_2 \mapsto x_1\}, & \{y_2 \mapsto g(x_2,c),\ z_2 \mapsto x_2\},\\
\{y_2 \mapsto g(x_2,c),\ z_2 \mapsto x_3\}, & \{y_2 \mapsto g(y_1,c),\ z_2 \mapsto x_2\}, & \{y_2 \mapsto g(y_1,c),\ z_2 \mapsto x_3\},\\
\{y_2 \mapsto g(a,x_3),\ z_2 \mapsto x_2\}, & \{y_2 \mapsto g(a,x_3),\ z_2 \mapsto x_3\}, & \{y_2 \mapsto g(x_2,x_3),\ z_2 \mapsto b\},\\
\{y_2 \mapsto g(x_2,x_3),\ z_2 \mapsto x_1\}, & \{y_2 \mapsto g(y_1,x_3),\ z_2 \mapsto b\}, & \{y_2 \mapsto g(y_1,x_3),\ z_2 \mapsto x_1\},\\
\{y_2 \mapsto g(y_1,x_3),\ z_2 \mapsto x_2\}, & \{y_2 \mapsto g(y_1,x_3),\ z_2 \mapsto x_3\}, & \{y_2 \mapsto g(x_2,x_3),\ z_2 \mapsto x_2\},\\
\{y_2 \mapsto g(x_2,x_3),\ z_2 \mapsto x_3\} & &
\end{array}
\right\}
$$

This set of extended abstraction substitutions gives rise to 28 extended computed generalizations.

1. $g(f(y_1,x_1),\ f(z_1,b))$
2. $g(f(y_1,x_1),\ f(z_1,x_1))$
3. $g(f(y_1,x_1),\ f(z_1,x_2))$
4. $g(f(y_1,x_1),\ f(z_1,x_3))$
5. $g(f(y_1,g(a,c)),\ f(z_1,b))$
6. $g(f(y_1,g(a,c)),\ f(z_1,x_1))$
7. $g(f(y_1,g(a,c)),\ f(z_1,x_2))$
8. $g(f(y_1,g(a,c)),\ f(z_1,x_3))$
9. $g(f(y_1,g(x_2,c)),\ f(z_1,b))$
10. $g(f(y_1,g(x_2,c)),\ f(z_1,x_1))$
11. $g(f(y_1,g(a,x_3)),\ f(z_1,b))$
12. $g(f(y_1,g(a,x_3)),\ f(z_1,x_1))$

13. $g(f(y_1, g(y_1, c)),\ f(z_1, b))$
14. $g(f(y_1, g(y_1, c)),\ f(z_1, x_1))$
15. $g(f(y_1, g(x_2, c)),\ f(z_1, x_2))$
16. $g(f(y_1, g(x_2, c)),\ f(z_1, x_3))$
17. $g(f(y_1, g(y_1, c)),\ f(z_1, x_2))$
18. $g(f(y_1, g(y_1, c)),\ f(z_1, x_3))$
19. $g(f(y_1, g(a, x_3)),\ f(z_1, x_2))$
20. $g(f(y_1, g(a, x_3)),\ f(z_1, x_3))$
21. $g(f(y_1, g(x_2, x_3)),\ f(z_1, b))$
22. $g(f(y_1, g(x_2, x_3)),\ f(z_1, x_1))$
23. $g(f(y_1, g(y_1, x_3)),\ f(z_1, b))$
24. $g(f(y_1, g(y_1, x_3)),\ f(z_1, x_1))$
25. $g(f(y_1, g(y_1, x_3)),\ f(z_1, x_2))$
26. $g(f(y_1, g(y_1, x_3)),\ f(z_1, x_3))$
27. $g(f(y_1, g(x_2, x_3)),\ f(z_1, x_2))$
28. $g(f(y_1, g(x_2, x_3)),\ f(z_1, x_3))$

By preserving only the least general ones, we obtain the six `lggs` below.

$$\left\{ \begin{array}{ll} g(f(y_1, g(a, c)), f(z_1, b)), & g(f(y_1, x_1), f(z_1, x_1)), \\ g(f(y_1, g(y_1, c)), f(z_1, b)), & g(f(y_1, g(y_1, x_3)), f(z_1, x_3)), \\ g(f(y_1, g(x_2, c)), f(z_1, x_2)), & g(f(y_1, g(a, x_3)), f(z_1, x_3)) \end{array} \right\}$$

-Final Configuration $\mathcal{C}_2$. Now, consider the second final configuration. The abstraction sets are:

$$\uparrow_{y_2}(D_2, S_{\mathcal{C}_2}) = \uparrow(g(a, c), \sigma^l_{S_{\mathcal{C}_2}}) = \left\{ \begin{array}{l} g(a, c),\ g(y_1, c),\ x_4,\ g(x_5, c),\ g(x_6, c), \\ g(a, x_7),\ g(y_1, x_7),\ g(x_5, x_7),\ g(x_6, x_7) \end{array} \right\},$$

$$\uparrow_{z_3}(D_2, S_{\mathcal{C}_2}) = \uparrow(g(c, c), \sigma^r_{S_{\mathcal{C}_2}}) = \left\{ \begin{array}{l} g(c, c),\ x_5,\ g(c, x_6),\ g(x_6, x_6),\ g(x_6, c),\ g(x_6, x_4), \\ x_7,\ g(c, x_4),\ g(x_4, c),\ g(x_4, x_4),\ g(x_4, x_6) \end{array} \right\}.$$

The set of abstraction substitutions $\Psi(D_2, S_{\mathcal{C}_2})$ has 99 elements. We omit the explicit list due to space constraints and show the corresponding 99 extended computed generalizations below.

1. $g(f(y_1, g(a, c)), f(g(c, c), z_4))$
2. $g(f(y_1, g(a, c)), f(g(c, x_4), z_4))$
3. $g(f(y_1, g(a, c)), f(g(c, x_6), z_4))$
4. $g(f(y_1, g(a, c)), f(g(x_4, c), z_4))$
5. $g(f(y_1, g(a, c)), f(g(x_4, x_4), z_4))$
6. $g(f(y_1, g(a, c)), f(g(x_4, x_6), z_4))$
7. $g(f(y_1, g(a, c)), f(g(x_6, c), z_4))$
8. $g(f(y_1, g(a, c)), f(g(x_6, x_4), z_4))$
9. $g(f(y_1, g(a, c)), f(g(x_6, x_6), z_4))$
10. $g(f(y_1, g(a, c)), f(x_5, z_4))$
11. $g(f(y_1, g(a, c)), f(x_7, z_4))$
12. $g(f(y_1, g(a, x_7)), f(g(c, c), z_4))$
13. $g(f(y_1, g(a, x_7)), f(g(c, x_4), z_4))$
14. $g(f(y_1, g(a, x_7)), f(g(c, x_6), z_4))$
15. $g(f(y_1, g(a, x_7)), f(g(x_4, c), z_4))$
16. $g(f(y_1, g(a, x_7)), f(g(x_4, x_4), z_4))$
17. $g(f(y_1, g(a, x_7)), f(g(x_4, x_6), z_4))$
18. $g(f(y_1, g(a, x_7)), f(g(x_6, c), z_4))$
19. $g(f(y_1, g(a, x_7)), f(g(x_6, x_4), z_4))$
20. $g(f(y_1, g(a, x_7)), f(g(x_6, x_6), z_4))$
21. $g(f(y_1, g(a, x_7)), f(x_5, z_4))$
22. $g(f(y_1, g(a, x_7)), f(x_7, z_4))$
23. $g(f(y_1, g(x_5, c)), f(g(c, c), z_4))$
24. $g(f(y_1, g(x_5, c)), f(g(c, x_4), z_4))$

25. $g(f(y_1, g(x_5, c)), f(g(c, x_6), z_4))$
26. $g(f(y_1, g(x_5, c)), f(g(x_4, c), z_4))$
27. $g(f(y_1, g(x_5, c)), f(g(x_4, x_4), z_4))$
28. $g(f(y_1, g(x_5, c)), f(g(x_4, x_6), z_4))$
29. $g(f(y_1, g(x_5, c)), f(g(x_6, c), z_4))$
30. $g(f(y_1, g(x_5, c)), f(g(x_6, x_4), z_4))$
31. $g(f(y_1, g(x_5, c)), f(g(x_6, x_6), z_4))$
32. $g(f(y_1, g(x_5, c)), f(x_5, z_4))$
33. $g(f(y_1, g(x_5, c)), f(x_7, z_4))$
34. $g(f(y_1, g(x_5, x_7)), f(g(c, c), z_4))$
35. $g(f(y_1, g(x_5, x_7)), f(g(c, x_4), z_4))$
36. $g(f(y_1, g(x_5, x_7)), f(g(c, x_6), z_4))$
37. $g(f(y_1, g(x_5, x_7)), f(g(x_4, c), z_4))$
38. $g(f(y_1, g(x_5, x_7)), f(g(x_4, x_4), z_4))$
39. $g(f(y_1, g(x_5, x_7)), f(g(x_4, x_6), z_4))$
40. $g(f(y_1, g(x_5, x_7)), f(g(x_6, c), z_4))$
41. $g(f(y_1, g(x_5, x_7)), f(g(x_6, x_4), z_4))$
42. $g(f(y_1, g(x_5, x_7)), f(g(x_6, x_6), z_4))$
43. $g(f(y_1, g(x_5, x_7)), f(x_5, z_4))$
44. $g(f(y_1, g(x_5, x_7)), f(x_7, z_4))$
45. $g(f(y_1, g(x_6, c)), f(g(c, c), z_4))$
46. $g(f(y_1, g(x_6, c)), f(g(c, x_4), z_4))$
47. $g(f(y_1, g(x_6, c)), f(g(c, x_6), z_4))$
48. $g(f(y_1, g(x_6, c)), f(g(x_4, c), z_4))$
49. $g(f(y_1, g(x_6, c)), f(g(x_4, x_4), z_4))$
50. $g(f(y_1, g(x_6, c)), f(g(x_4, x_6), z_4))$
51. $g(f(y_1, g(x_6, c)), f(g(x_6, c), z_4))$
52. $g(f(y_1, g(x_6, c)), f(g(x_6, x_4), z_4))$
53. $g(f(y_1, g(x_6, c)), f(g(x_6, x_6), z_4))$
54. $g(f(y_1, g(x_6, c)), f(x_5, z_4))$
55. $g(f(y_1, g(x_6, c)), f(x_7, z_4))$
56. $g(f(y_1, g(x_6, x_7)), f(g(c, c), z_4))$
57. $g(f(y_1, g(x_6, x_7)), f(g(c, x_4), z_4))$
58. $g(f(y_1, g(x_6, x_7)), f(g(c, x_6), z_4))$
59. $g(f(y_1, g(x_6, x_7)), f(g(x_4, c), z_4))$
60. $g(f(y_1, g(x_6, x_7)), f(g(x_4, x_4), z_4))$
61. $g(f(y_1, g(x_6, x_7)), f(g(x_4, x_6), z_4))$
62. $g(f(y_1, g(x_6, x_7)), f(g(x_6, c), z_4))$
63. $g(f(y_1, g(x_6, x_7)), f(g(x_6, x_4), z_4))$
64. $g(f(y_1, g(x_6, x_7)), f(g(x_6, x_6), z_4))$
65. $g(f(y_1, g(x_6, x_7)), f(x_5, z_4))$
66. $g(f(y_1, g(x_6, x_7)), f(x_7, z_4))$
67. $g(f(y_1, g(y_1, c)), f(g(c, c), z_4))$
68. $g(f(y_1, g(y_1, c)), f(g(c, x_4), z_4))$
69. $g(f(y_1, g(y_1, c)), f(g(c, x_6), z_4))$
70. $g(f(y_1, g(y_1, c)), f(g(x_4, c), z_4))$
71. $g(f(y_1, g(y_1, c)), f(g(x_4, x_4), z_4))$
72. $g(f(y_1, g(y_1, c)), f(g(x_4, x_6), z_4))$
73. $g(f(y_1, g(y_1, c)), f(g(x_6, c), z_4))$
74. $g(f(y_1, g(y_1, c)), f(g(x_6, x_4), z_4))$
75. $g(f(y_1, g(y_1, c)), f(g(x_6, x_6), z_4))$
76. $g(f(y_1, g(y_1, c)), f(x_5, z_4))$
77. $g(f(y_1, g(y_1, c)), f(x_7, z_4))$
78. $g(f(y_1, g(y_1, x_7)), f(g(c, c), z_4))$
79. $g(f(y_1, g(y_1, x_7)), f(g(c, x_4), z_4))$
80. $g(f(y_1, g(y_1, x_7)), f(g(c, x_6), z_4))$
81. $g(f(y_1, g(y_1, x_7)), f(g(x_4, c), z_4))$
82. $g(f(y_1, g(y_1, x_7)), f(g(x_4, x_4), z_4))$
83. $g(f(y_1, g(y_1, x_7)), f(g(x_4, x_6), z_4))$
84. $g(f(y_1, g(y_1, x_7)), f(g(x_6, c), z_4))$
85. $g(f(y_1, g(y_1, x_7)), f(g(x_6, x_4), z_4))$
86. $g(f(y_1, g(y_1, x_7)), f(g(x_6, x_6), z_4))$
87. $g(f(y_1, g(y_1, x_7)), f(x_5, z_4))$
88. $g(f(y_1, g(y_1, x_7)), f(x_7, z_4))$
89. $g(f(y_1, x_4), f(g(c, c), z_4))$
90. $g(f(y_1, x_4), f(g(c, x_4), z_4))$
91. $g(f(y_1, x_4), f(g(c, x_6), z_4))$
92. $g(f(y_1, x_4), f(g(x_4, c), z_4))$
93. $g(f(y_1, x_4), f(g(x_4, x_4), z_4))$
94. $g(f(y_1, x_4), f(g(x_4, x_6), z_4))$
95. $g(f(y_1, x_4), f(g(x_6, c), z_4))$
96. $g(f(y_1, x_4), f(g(x_6, x_4), z_4))$
97. $g(f(y_1, x_4), f(g(x_6, x_6), z_4))$
98. $g(f(y_1, x_4), f(x_5, z_4))$
99. $g(f(y_1, x_4), f(x_7, z_4))$

By preserving only the least general ones, we get the 11 lggs below.

$$\left\{\begin{array}{ll} g(f(y_1, g(x_6, c)), f(g(x_6, x_6), z_4)), & g(f(y_1, g(a, x_7)), f(x_7, z_4)), \\ g(f(y_1, g(x_6, c)), f(g(c, x_6), z_4)), & g(f(y_1, g(x_5, c)), f(x_5, z_4)), \\ g(f(y_1, g(x_6, c)), f(g(x_6, c), z_4)), & g(f(y_1, g(a, c)), f(g(c, c), z_4)), \\ g(f(y_1, g(y_1, c)), f(g(c, c), z_4)), & g(f(y_1, g(y_1, x_7)), f(x_7, z_4)) \\ g(f(y_1, x_4), f(g(x_4, c), z_4)), & g(f(y_1, x_4), f(g(x_4, x_4), z_4)) \\ g(f(y_1, x_4), f(g(c, x_4), z_4)) & \end{array}\right\}$$

-Final Configuration $\mathcal{C}_3$. The corresponding abstraction sets of the labels in the delayed set of the third final configuration are:

$$\uparrow_{y_3}(D_3, S_{\mathcal{C}_3}) = \uparrow(a, \sigma^l_{S_3}) = \{a, x_8\}, \quad \uparrow_{z_6}(D_3, S_{\mathcal{C}_3}) = \uparrow(b, \sigma^r_{S_3}) = \{b, x_8\}.$$

The resulting abstraction substitutions are listed below.

$$\Psi(D_3, S_{\mathcal{C}_3}) = \left\{\begin{array}{ll} \{y_3 \mapsto a, z_6 \mapsto b\}, & \{y_3 \mapsto a, z_6 \mapsto x_8\}, \\ \{y_3 \mapsto x_8, z_6 \mapsto b\}, & \{y_3 \mapsto x_8, z_6 \mapsto x_8\} \end{array}\right\}.$$

These substitutions lead to four generalizations, two of which are lggs (the highlighted ones).

1. $g(f(a, y_4), f(z_5, b))$
2. $g(f(a, y_4), f(z_5, x_8))$
3. $g(f(x_8, y_4), f(z_5, b))$
4. $g(f(x_8, y_4), f(z_5, x_8))$

-Final Configuration $\mathcal{C}_4$. The corresponding abstraction sets of the labels in the delayed set D_4 are:

$$\uparrow_{y_3}(D_4, S_{\mathcal{C}_4}) = \uparrow(a, \sigma^l_{S_3}) = \{a, x_9, x_{10}\}, \quad \uparrow_{z_7}(D_4, S_{\mathcal{C}_4}) = \uparrow(g(c, c), \sigma^r_{S_1}) = \{g(c, c), x_9\}.$$

The resulting abstraction substitutions are listed below.

$$\Psi(D_4, S_{\mathcal{C}_4}) = \left\{\begin{array}{ll} \{y_3 \mapsto a, z_7 \mapsto g(c, c)\}, & \{y_3 \mapsto a, z_7 \mapsto x_9\}, \\ \{y_3 \mapsto x_9, z_7 \mapsto g(c, c)\}, & \{y_3 \mapsto x_9, z_7 \mapsto x_9\}, \\ \{y_3 \mapsto x_{10}, z_7 \mapsto g(c, c)\}, & \{y_3 \mapsto x_{10}, z_7 \mapsto x_9\} \end{array}\right\}.$$

These substitutions lead to six generalizations, two of which are lggs (the highlighted ones).

1. $g(f(a, y_4), f(g(c, c), z_8))$
2. $g(f(x_9, y_4), f(g(c, c), z_8))$
3. $g(f(x_{10}, y_4), f(g(c, c), z_8))$
4. $g(f(a, y_4), f(x_9, z_8))$
5. $g(f(x_9, y_4), f(x_9, z_8))$
6. $g(f(x_{10}, y_4), f(x_9, z_8))$

Finally, the set of 21 incomparable `lgg`s obtained from all the final configurations is listed below.

$$\left\{\begin{array}{ll}
g(f(y_1, g(a, c)), f(z_1, b)), & g(f(y_1, g(y_1, c)), f(z_1, b)), \\
g(f(y_1, x_1), f(z_1, x_1)), & g(f(y_1, g(y_1, x_3)), f(z_1, x_3)), \\
g(f(y_1, g(x_2, c)), f(z_1, x_2)), & g(f(y_1, g(a, x_3)), f(z_1, x_3)), \\
g(f(y_1, g(y_1, c)), f(g(c, c), z_4)), & g(f(y_1, g(a, c)), f(g(c, c), z_4)), \\
g(f(y_1, g(x_6, c)), f(g(c, x_6), z_4)), & g(f(y_1, g(x_5, c)), f(x_5, z_4)), \\
g(f(y_1, g(x_6, c)), f(g(x_6, c), z_4)), & g(f(y_1, g(a, x_7)), f(x_7, z_4)), \\
g(f(y_1, g(x_6, c)), f(g(x_6, x_6), z_4)), & g(f(y_1, g(y_1, x_7)), f(x_7, z_4)), \\
g(f(y_1, x_4), f(g(x_4, c), z_4)), & g(f(y_1, x_4), f(g(x_4, x_4), z_4)), \\
g(f(y_1, x_4), f(g(c, x_4), z_4)), & g(f(a, y_4), f(g(c, c), z_8)), \\
g(f(a, y_4), f(z_5, b)), & g(f(x_9, y_4), f(x_9, z_8)), \\
g(f(x_8, y_4), f(z_5, x_8)) &
\end{array}\right\}$$

Notice that the highlighted `lgg`s correspond to the generalizations computed in Example 3.2.4, while with the extension, we obtain 15 additional `lgg`s.

Index