

Submitted by
Mara Antesberger
k11803026

Submitted at
**Research Institute for
Symbolic Computation
(RISC)**

Supervisor
Priv.-Doz. Dipl.-Math.
Dr. **Temur Kutsia**

February, 2026

EXTENDING APPROXIMATE REASONING TO UNRANKED TERM STRUCTURES

Unification, Matching and Anti-Unification



Master Thesis

to obtain the academic degree of

Diplom-Ingenieurin

in the Master's Program

Computational Mathematics

Abstract

This thesis introduces new procedures for solving unranked unification, matching and anti-unification problems with fuzzy similarity and proximity relations. We extend existing algorithms for unranked terms in such a way that, instead of requiring syntactic equality between symbols, they can also handle both similarity and proximity relations. This allows for the computation of approximate solutions annotated with quantitative degrees reflecting their similarity or closeness. We prove soundness, completeness, and, whenever applicable, termination, showing that the proposed algorithms and procedures behave correctly and exhaustively under the intended semantics. Each procedure is accompanied by small examples that demonstrate its application. Overall, the thesis provides a unified framework for fundamental techniques for reasoning about unranked term structures that goes beyond strict syntactic equality, allowing terms to be related by more general notions such as similarity or proximity.

Kurzfassung

Diese Arbeit stellt neue Verfahren zur Lösung von unrankierten Unifikations-, Matching- und Anti-Unifikationsproblemen unter unscharfen Ähnlichkeits- und Proximitätsrelationen vor. Bereits bestehende Algorithmen für unrankierte Terme werden so erweitert, dass sie anstelle der strikten syntaktischen Gleichheit zwischen Symbolen auch Ähnlichkeits- und Proximitätsrelationen verarbeiten können. Dies ermöglicht die Berechnung approximativer Lösungen, die mit quantitativen Graden annotiert sind, welche den Grad ihrer Ähnlichkeit oder Nähe widerspiegeln. Es werden Korrektheit (Soundness), Vollständigkeit (Completeness) und – soweit anwendbar – Terminierung bewiesen. Damit wird gezeigt, dass die vorgeschlagenen Algorithmen und Verfahren unter der beabsichtigten Semantik korrekt und vollständig arbeiten. Jedes Verfahren wird durch kleine Beispiele ergänzt, die seine Anwendung veranschaulichen. Insgesamt bietet die Arbeit ein einheitliches Rahmenwerk für grundlegende Techniken für das Schlussfolgern über unrankierten Termstrukturen, das über strikte syntaktische Gleichheit hinausgeht und erlaubt, Terme über allgemeinere Konzepte wie Ähnlichkeit oder Proximität miteinander in Beziehung zu setzen.

Contents

1	Introduction	1
2	Preliminaries	4
2.1	Terms and Substitutions	4
2.2	X-Terms	7
2.3	Equations and Solved Sets	10
2.4	Fuzzy Relations	11
2.5	Ordering	15
2.6	Unification, Matching, Anti-Unification	15
2.6.1	Unification	16
2.6.2	Matching	17
2.6.3	Anti-Unification	17
3	Unification	19
3.1	Fuzzy Unranked Unification with Similarity Relations	22
3.2	Fuzzy Unranked Unification with Proximity Relations	34
4	Matching	52
4.1	Fuzzy Unranked Matching with Similarity Relations	53
4.2	Fuzzy Unranked Matching with Proximity Relations	64
5	Anti-Unification	74
5.1	Fuzzy Unranked Anti-unification with Similarity Relations	77
5.2	Fuzzy Unranked Anti-Unification with Proximity Relations	86
6	Conclusion	94

1 Introduction

Reasoning with symbolic structures is an important task in many areas of computer science, including automated deduction, knowledge representation, program analysis, and natural language processing. Operations like unification, matching, and anti-unification form the core computational basis of these reasoning tasks. In their classical form, these operations work on *ranked* terms, where every function symbol has a fixed number of arguments, and they assume exact equality between symbols.

Many real-world applications, however, require more flexibility. *Unranked terms*, whose function symbols can take any number of arguments, appear naturally in fields such as tree-structured data, XML representation, and natural-language syntax. In addition, exact equality of symbols is often too strict: in various practical reasoning scenarios, it is difficult to guarantee and, therefore, it is more useful to compare symbols by fuzzy similarity or proximity relations, which describe to what degree two symbols are alike. This naturally leads to approximate reasoning, where solutions are annotated with a *degree* that indicates how well they satisfy a given problem.

A proximity relation is a reflexive and symmetric binary fuzzy relation (fuzzy tolerance), while a similarity relation, in addition, satisfies a special form of transitivity (fuzzy equivalence). Unification, matching, and anti-unification techniques with similarity or proximity relations is well-studied. Early research on unification modulo similarity emerged in the early 2000s [13, 14, 15, 40, 50]. In 2002, Maria Sessa introduced her ‘weak’ unification algorithm in [49]. The algorithm was able to handle unification problems under a similarity relation and was suitable for practical use. Versions of similarity-based unification have been successfully integrated into reasoning and logic programming formalisms, see, e.g., [36, 37, 42, 48, 51, 52]. In most of these works, unifiable terms have the same arities. Later, in [1], Hassan Aït-Kaci and Gabriella Pasi built on Sessa’s idea and extended it to work with arity-mismatched similarity unification problems, permitting two similar terms to have different number of arguments. Signatures that permit such terms are called fully fuzzy ones, emphasizing possible mismatch between similar expressions not only in the names of symbols, but also in their arities. The mentioned paper by Aït-Kaci and Pasi covers also similarity-based anti-unification problems in fully fuzzy signatures.

A proximity-based weak unification algorithm was developed by Pascual Julián Iranzo and Clemente Rubio-Manzano in [22] in 2015. It utilizes proximity blocks to determine closeness of terms, where blocks represent cliques in graphs induced by proximity relations. A restriction is that the same term is forbidden to belong to two or more different cliques at the same time. Together with Fernando Sáenz-Pérez, Julián Iranzo proposed a further improvement of this algorithm in [26]. Proximity-block-based unification is the main computational mechanism for the fuzzy logic programming languages Bousi~Prolog [24] and FASILL [21], for fuzzy Datalog [23], and was used, e.g., in an integration of WordNet into fuzzy logic programming [25].

A different way of handling proximity relations in unification was proposed by Temur Kutsia and Cleo Pau in 2019 in [35] by using a class-based approach. A proximity class of an expression is a set of all expressions that are proximal to the given one by a nonzero degree. (It corresponds to the neighbourhood of a node in a graph induced by the proximity relation). They also studied matching and anti-unification from the proximity-class-based perspective in [34] and constructed algorithms for solving proximity-based unification, matching and anti-unification problems under fully fuzzy signatures in [33, 44]. More detailed descriptions of these class-based problems and the corresponding algorithms can be found in Cleo Pau's doctoral thesis [43].

Unification, matching, and anti-unification for unranked terms has been explored for syntactic equivalence in several works, e.g., [4, 6, 10, 17, 18, 29, 30, 46, 55]. These techniques found applications in knowledge representation, management, and reasoning [7, 17, 19], program analysis, transformation, and repair [3, 41, 46, 47, 53], rewriting and rule-based transformations [8, 12, 18, 39], XML processing [20, 31], mathematical assistant systems [54], etc. However, the combination of fuzzy relations and unranked terms has been largely unexplored until now, except a version of proximity-based matching [11] developed in the context of rule-based programming, and a proximity-based generalization [5] for unranked nominal expressions. This thesis aims to fill this gap by extending some of the unification, matching, and anti-unification procedures to handle unranked terms together with similarity and proximity relations. We use the minimum T-norm to lift these relations from symbols to terms, and to define the transitivity property for similarity relations.

In order to ease the entry into this topic, we begin by collecting all the notions which that will be needed throughout the thesis. They are presented in Chapter 2 and range from the common concepts, such as terms and substitutions, to graded proximity classes and orderings. The chapter also gives an informal overview of unification, matching, and anti-unification.

Chapter 3 focuses on the unification procedure. After looking at the concept of syntactic unranked unification introduced in [28, 29], these ideas are then used to create procedures first for unranked unification with similarity relations, and then for unranked unification with

proximity relations. For solving under similarity, we take inspirations from [49], as well as from [44] for notational conventions. Solving with proximity relations is built on the ideas of presented in [35] and [43].

In Chapter 4, quantitative versions of unranked matching is investigated. Building on the syntactic approach of Besik Dundua, Temur Kutsia, and Mircea Marin [10], the algorithms are adapted to work with both similarity and proximity while keeping consistent with the notation we have introduced so far. As in [10], algorithms for linear matching and for reconstructing solutions of linearized non-linear problems are described. When solving with proximity relations, proximity classes are used in the form of extended terms (X-terms), following the ideas from [34]. This gives a compact representation of the solution set, which is the main difference from the algorithm presented in [11], where solutions are computed explicitly, not in a compact form.

Based on the syntactic unranked anti-unification algorithm introduced by Temur Kutsia, Jordi Levy and Mateu Villaret in [30], the final part deals with extending the rigid version of syntactic unranked anti-unification to the fuzzy case. After altering all the needed notions to work with similarity relations, we show the rewritten rules to generalize terms under consideration of both a rigidity function and similarity relation. Guiding the way to incorporate proximity relations is again [34].

2 Preliminaries

In this chapter the basic notions around terms and substitutions, extended terms, solved sets, fuzzy relations and ordering, as well as an introduction to unification, matching and anti-unification are given.

2.1 Terms and Substitutions

The inspiration for the notations for terms and substitution largely comes from [29]. We consider a countable set of variables \mathcal{V} and a countable set of function symbols \mathcal{F} . The set of function symbols is partitioned into the set of *fixed-arity* function symbols \mathcal{F}_{ix} as well as the set of *flexible-arity* or *variadic* function symbols \mathcal{F}_{lex} , hence $\mathcal{F} = \mathcal{F}_{\text{ix}} \cup \mathcal{F}_{\text{lex}}$. The *arity* or length of the argument list n of $f \in \mathcal{F}_{\text{ix}}$ will be written as $\text{ar}(f) = n$ or sometimes we simply write $f \in \mathcal{F}^n$. Constants are fixed arity function symbols with an empty argument list, i.e., $\text{ar}(c) = 0$ for a constant c . Flexible arity function symbols are not bound by arity, i.e., one instance could have arity n , while another instance could have arity m .

The set of variables is comprised of the set of *individual* variable symbols \mathcal{V}_I and the set of *sequence* variable symbols \mathcal{V}_S , i.e., $\mathcal{V} = \mathcal{V}_I \cup \mathcal{V}_S$.

The set of terms is defined over \mathcal{F} and \mathcal{V} and written as $\mathcal{T}(\mathcal{F}, \mathcal{V})$. A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ is constructed as

$$\begin{aligned}
 t &:= t_I \mid t_S \\
 t_I &:= x \mid f(t_1, \dots, t_m) \mid g(t_{I_1}, \dots, t_{I_n}) \\
 t_S &:= \bar{x} \mid t_I,
 \end{aligned}$$

where t_I represents *individual terms* and t_S represents *sequence terms*; also $n, m \geq 0$, $x \in \mathcal{V}_I$, $\bar{x} \in \mathcal{V}_S$, $f \in \mathcal{F}_{\text{lex}}$ and $g \in \mathcal{F}_{\text{ix}}$ with $\text{ar}(g) = n$. Note that fixed arity function symbols do not apply to sequence variables.

Here is a short overview, which symbols will be generally used to represent certain notions:

- individual variables: x, y, z, v

- sequence variables: $\bar{x}, \bar{y}, \bar{z}, \bar{v}$
- individual or sequence variables: $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{v}$
- constants: a, b, c
- function symbols: f, g, h
- terms: s, t, r
- sequences of terms: $\tilde{s}, \tilde{t}, \tilde{r}$

Certain chapters might have slight deviations; however, changes will be stated.

For simplicity, throughout the thesis we assume that the only fixed-arity symbols are constants, although the results can be easily extended to the unrestricted case.

A term of the form $f()$ can be written as simply as f . Throughout the thesis, we will be dealing with finite sequences of terms, which we denote by $\ulcorner s_1, \dots, s_n \urcorner$. Such a sequence can be, in particular, empty. It will be represented as $\ulcorner \urcorner$. If a sequence of terms consists of only one element it can be simply written as a term, i.e. $\ulcorner t \urcorner = t$.

The *head* of a term is either the function symbol f if $t = f(s_1, \dots, s_n)$ or if $t = \mathbf{x}$ with $\mathbf{x} \in \mathcal{V}$, then $\text{head}(t) = \mathbf{x}$. The head of a sequence of terms is the sequence of the term heads, i.e. $\text{head}(\ulcorner s_1, \dots, s_n \urcorner) = \ulcorner \text{head}(s_1), \dots, \text{head}(s_n) \urcorner$.

The set of variables occurring in a term t will be denoted by $\mathcal{V}(t)$. Analogously the set of individual variables $\mathcal{V}_I(t)$, the set of sequence variables $\mathcal{V}_S(t)$ and the set of function symbols $\mathcal{F}(t)$ occurring in t are defined. If $\mathcal{V}(t) = \emptyset$ then t is called *ground*. If no variable occurs more than once in a term t , then t is called *linear*.

The notion of *positions* of terms and sequences and how to denote them, is taken from [43] and [30]: The set of positions of a term s is denoted by $\text{pos}(s)$ and defined as

$$\begin{aligned} \text{pos}(\mathbf{x}) &:= \{\ulcorner \urcorner\}, \\ \text{pos}(f(s_1 \dots, s_n)) &:= \{\ulcorner \urcorner\} \cup \bigcup_{i=1}^n \{i.p \mid p \in \text{pos}(s_i)\}. \end{aligned}$$

The symbol at position p in s can be denoted as $s[p]$, whilst we write $s|_p$ for the subterm of s at position p . If we want to address a specific element within a sequence of terms \tilde{s} , we write $\tilde{s}|_i$ to denote the i 'th element of the sequence. In order to denote a specific subsequence of \tilde{s} , we use the notation $\tilde{s}|_i^j$ with $i < j$, which represents the subsequence of \tilde{s} between position i and j , with i and j not included, i.e. $\tilde{s}|_i^j = \ulcorner \tilde{s}|_{i+1}, \dots, \tilde{s}|_{j-1} \urcorner$.

The *size* of a term represents the number of symbols occurring in a term: $\text{size}(\mathbf{x}) = 1$, $\text{size}(f(s_1, \dots, s_n)) = 1 + \sum_{i=1}^n \text{size}(s_i)$ and $\text{size}(\ulcorner t_1, \dots, t_n \urcorner) = \sum_{i=1}^n \text{size}(t_i)$. Additionally,

it is defined on pairs of sequences and their sets in a straightforward way: $\text{size}((\tilde{s}, \tilde{t})) = \text{size}(\tilde{s}) + \text{size}(\tilde{t})$ and $\text{size}(S) = \sum_{(\tilde{s}, \tilde{t}) \in S} \text{size}((\tilde{s}, \tilde{t}))$.

A *substitution* is a mapping from variables to terms, specifically from individual variables to individual terms and from sequence variables to finite (possibly empty) sequences of terms. All but finitely many variables are mapped to themselves. We will use the notation of representing a substitution as a finite set of bindings:

$$\{x_1 \mapsto t_1, \dots, x_n \mapsto t_n, \bar{x}_1 \mapsto \tilde{t}_1, \dots, \bar{x}_m \mapsto \tilde{t}_m\}.$$

A substitution is applied to a term in the following way:

$$t\sigma = \begin{cases} x\sigma & \text{if } x \in \mathcal{V}_I, \\ \bar{x}\sigma & \text{if } \bar{x} \in \mathcal{V}_S, \\ f(t_1\sigma, \dots, t_n\sigma) & \text{if } t = f(t_1, \dots, t_n). \end{cases}$$

We will be using $\sigma, \theta, \varphi, \dots$ to represent arbitrary substitutions and ε for the identity substitution, i.e., the substitution mapping everything to itself.

The set of variables which get changed by a substitution σ is called the *domain* and is defined as

$$\text{dom}(\sigma) = \{\mathbf{x} \mid \mathbf{x}\sigma \neq \mathbf{x}\},$$

the set of terms which the variables of the domain of σ are mapped to is called the *range* and is defined as

$$\text{ran}(\sigma) = \{\mathbf{x}\sigma \mid \mathbf{x} \in \text{dom}(\sigma)\},$$

and the set of variables occurring in the range of σ is called the *variable range* and is defined as

$$\text{vran}(\sigma) = \mathcal{V}(\text{ran}(\sigma)).$$

In the case we want to *restrict* a substitution σ to a certain set of variables \mathcal{X} we can write $\sigma|_{\mathcal{X}}$ which means for $\mathbf{x} \in \mathcal{X}$ that $\mathbf{x}\sigma|_{\mathcal{X}} = \mathbf{x}\sigma$ and for $\mathbf{x} \notin \mathcal{X}$ that $\mathbf{x}\sigma|_{\mathcal{X}} = \mathbf{x}$.

Definition 2.1

A composition $\sigma\theta$ of two substitutions σ and θ is done in the following way:

1. θ has to be applied to every term in $\text{ran}(\sigma)$ and afterwards trivial mappings such as $\mathbf{x} \mapsto \mathbf{x}$ have to be removed,
2. remove those mappings $\mathbf{x} \mapsto t$ of θ where $\mathbf{x} \in \text{dom}(\sigma)$,
3. take the union of the modified substitutions from Step 1 and Step 2.

Another way of representing a substitution is through the *triangular form*, as described in [2], which is a sequence of bindings in the form:

$$\lceil x_1 \mapsto t_1; \dots; x_n \mapsto t_n; \bar{x}_1 \mapsto \tilde{t}_1; \dots; \bar{x}_m \mapsto \tilde{t}_m \rceil,$$

which represents the composition of the single binding substitutions in the respective order: $\{x_1 \mapsto t_1\} \cdots \{x_n \mapsto t_n\} \{\bar{x}_1 \mapsto \tilde{t}_1\} \cdots \{\bar{x}_m \mapsto \tilde{t}_m\}$.

If we want to *replace* a specific binding within a triangular substitution, we will use the notation as defined in [43],

$$\text{rep}(\sigma, \sigma_i \rightsquigarrow \sigma'_i),$$

which given a substitution $\sigma = \lceil \sigma_1; \dots; \sigma_i; \dots; \sigma_n \rceil$, replaces the binding σ_i by the new binding σ'_i , i.e. $\text{rep}(\sigma, \sigma_i \rightsquigarrow \sigma'_i) = \lceil \sigma_1; \dots; \sigma'_i; \dots; \sigma_n \rceil$.

A comparison between two substitutions can be done the following ways:

Definition 2.2

A substitution σ is syntactically equal to a substitution θ on a finite set of variables \mathcal{X} , denoted as $\sigma =^{\mathcal{X}} \theta$, if $\mathbf{x}\sigma = \mathbf{x}\theta$ for all $\mathbf{x} \in \mathcal{X}$.

Note that if no \mathcal{X} is written, then it is assumed over $\text{dom}(\sigma) \cup \text{dom}(\theta)$.

Definition 2.3

A substitution σ is called syntactically more general than a substitution θ on a set of variables \mathcal{X} , if there exists a substitution ν such that $\sigma\nu =^{\mathcal{X}} \theta$. We then write $\sigma \preceq^{\mathcal{X}} \theta$. θ is called an instance of σ , and σ is called a generalization of θ .

If it holds that $\sigma \preceq^{\mathcal{X}} \theta$ and $\theta \preceq^{\mathcal{X}} \sigma$ then σ and θ are equigeneral, written as $\sigma \approx^{\mathcal{X}} \theta$.

It is also possible to compare terms in the same way: A term s is said to be syntactically more general than a term t , if there exists a substitution σ such that $s\sigma = t$. This relation is written as $s \preceq t$.

Since \preceq is also transitive, it is a quasi-ordering which is called the instantiation quasi-ordering.

2.2 X-Terms

Following [34] and [43], we are also going to need an extension of terms that handles finite sets of function symbols instead of just a single function symbol. Such finite sets of function symbols will be denoted by the bold-faced letters **f**, **g**, **h**.

An *extended term*, also called *X-term* for short, is defined over \mathcal{F} and \mathcal{V} as

$$\mathbf{t} := \mathbf{x} \mid \mathbf{f}(\mathbf{t}_1, \dots, \mathbf{t}_n),$$

with $\mathbf{x} \in \mathcal{V}$ and \mathbf{f} being a non-empty finite set of unranked function symbols or constants (in the latter case, $n = 0$). The set of X-terms is written as $\mathcal{T}_X(\mathcal{F}, \mathcal{V})$ and its elements will be denoted by the bold-faced letters $\mathbf{s}, \mathbf{t}, \mathbf{r}$. Slightly abusing the notion we can view ‘regular’ terms as X-terms with only one symbol in each set.

Below we extend the notions defined in Section 2.1 to X-terms.

The *head* of an X-term is defined as $\text{head}(\mathbf{x}) := \mathbf{x}$ and $\text{head}(\mathbf{f}(\mathbf{t}_1, \dots, \mathbf{t}_n)) := \mathbf{f}$.

Substitutions over X-terms or *extended substitutions / X-substitutions* are mappings from variables to X-terms, obeying the same restrictions as their standard counterparts from Section 2.1. Not much changes with the switch to X-terms, e.g., we write

$$\mathbf{f}(\mathbf{t}_1, \dots, \mathbf{t}_n)\sigma = \mathbf{f}(\mathbf{t}_1\sigma, \dots, \mathbf{t}_n\sigma).$$

Again we will use bold-faced letters to denote X-substitutions: σ, θ, φ .

We will define the *size* of X-terms as the structural size:

$$\begin{aligned} \text{size}(\mathbf{x}) &= 1, \\ \text{size}(\mathbf{f}(\mathbf{t}_1, \dots, \mathbf{t}_n)) &= 1 + \sum_{i=1}^n \text{size}(\mathbf{t}_i). \end{aligned}$$

In order to extract all possible terms and substitutions from an X-term and X-substitution respectively, we define $\text{terms}(\mathbf{t})$ as the *set of terms represented by an X-term \mathbf{t}* :

$$\text{terms}(\mathbf{x}) := \{\mathbf{x}\}, \quad \text{terms}(\mathbf{f}(\mathbf{t}_1, \dots, \mathbf{t}_n)) := \{f(t_1, \dots, t_n) \mid f \in \mathbf{f}, t_i \in \text{terms}(\mathbf{t}_i), 1 \leq i \leq n\}$$

and the *set of all substitutions represented by an X-substitution σ* as:

$$\text{subs}(\sigma) := \{\sigma \mid \sigma(x) \in \text{terms}(\sigma(\mathbf{x})) \text{ for all } \mathbf{x} \in \mathcal{V}\}.$$

A finite, possibly empty, sequence of X-terms can be written as $\tilde{\mathbf{s}} = \lceil \mathbf{s}_1, \dots, \mathbf{s}_n \rceil$. We will use $\tilde{\mathbf{s}}, \tilde{\mathbf{t}}, \tilde{\mathbf{r}}$, to denote sequences of X-terms.

Next, we define the notion of intersection of two extended terms. Let $\mathbf{s} = \mathbf{f}(\tilde{\mathbf{s}})$ and $\mathbf{t} = \mathbf{g}(\tilde{\mathbf{t}})$. The intersection $\mathbf{s} \sqcap \mathbf{t}$ is defined as:

$$\mathbf{s} \sqcap \mathbf{t} = \begin{cases} \emptyset, & \text{if } \mathbf{f} \cap \mathbf{g} = \emptyset \text{ and } \tilde{\mathbf{s}} \sqcap \tilde{\mathbf{t}} \neq \emptyset, \\ (\mathbf{f} \cap \mathbf{g})(\tilde{\mathbf{s}} \sqcap \tilde{\mathbf{t}}), & \text{otherwise.} \end{cases}$$

The intersection of sequences of X-terms $\tilde{\mathbf{s}} \sqcap \tilde{\mathbf{t}}$ is defined recursively as follows:

$$\begin{aligned} \sqcap \sqcap \sqcap &= \sqcap, \\ \mathbf{x} \sqcap \mathbf{x} &= \mathbf{x}, \text{ for all } \mathbf{x} \in \mathcal{V}, \\ \lceil \mathbf{r}_1, \tilde{\mathbf{r}}_1 \rceil \sqcap \lceil \mathbf{r}_2, \tilde{\mathbf{r}}_2 \rceil &= \begin{cases} \lceil \mathbf{r}, (\tilde{\mathbf{r}}_1 \sqcap \tilde{\mathbf{r}}_2) \rceil & \text{if } \mathbf{r} = \mathbf{r}_1 \sqcap \mathbf{r}_2 \neq \emptyset, \\ \emptyset, & \text{otherwise,} \end{cases} \\ \lceil \bar{x}, \tilde{\mathbf{r}}_1 \rceil \sqcap \tilde{\mathbf{t}} &= \lceil \tilde{\mathbf{t}}_1, (\tilde{\mathbf{r}}_1 \sqcap \tilde{\mathbf{t}}_2) \rceil, \text{ for } \tilde{\mathbf{t}} = \lceil \tilde{\mathbf{t}}_1, \tilde{\mathbf{t}}_2 \rceil, \\ \tilde{\mathbf{s}} \sqcap \lceil \bar{y}, \tilde{\mathbf{r}}_2 \rceil &= \lceil \tilde{\mathbf{s}}_1, (\tilde{\mathbf{s}} \sqcap \tilde{\mathbf{r}}_2) \rceil, \text{ for } \tilde{\mathbf{s}} = \lceil \tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2 \rceil. \end{aligned}$$

In any other case, the intersection yields \emptyset .

Note that the last two steps can lead to branching due to the sequence variables and how much to assign to them. Some branches might lead to \emptyset while others result in a sequence of X-terms.

Example 2.1. Let $\mathbf{s} = \{f, g\}(\{a, b, c\}, x, \{a\})$ and $\mathbf{t} = \{g\}(\{b, c\}, x, \{a, c\})$. Then $\mathbf{s} \sqcap \mathbf{t} = \{g\}(\{a, b\}, x, \{a\})$.

Example 2.2. Let $\mathbf{s} = \{f, g\}(\{a, b, c\}, x, \{a\})$ and $\mathbf{t} = \{g\}(\{b, c\}, x, \{b, c\})$. Then $\mathbf{s} \sqcap \mathbf{t} = \emptyset$.

Example 2.3. Given

$$\tilde{\mathbf{s}} = \lceil \{f\}(\{a, b\}), \{f, g\}(x), \bar{x} \rceil \text{ and } \tilde{\mathbf{t}} = \lceil \{f, h\}(\{b\}), \{f\}(x), \{h\}(\{b\}), \{g, h\}(y) \rceil,$$

we will compute $\tilde{\mathbf{s}} \sqcap \tilde{\mathbf{t}}$:

- position 1: $\{f\}(\{a, b\}) \sqcap \{f, h\}(\{b\}) = \{f\}(\{b\})$
- position 2: $\{f, g\}(x) \sqcap \{f\}(x) = \{f\}(x)$
- position 3: $\lceil \bar{x} \rceil \sqcap \lceil \{h\}(\{b\}), \{g, h\}(y) \rceil = \lceil \{h\}(\{b\}), \{g, h\}(y) \rceil$ (this is the only successful branch)

resulting in $\tilde{\mathbf{s}} \sqcap \tilde{\mathbf{t}} = \lceil \{f\}(\{b\}), \{f\}(x), \{h\}(\{b\}), \{g, h\}(y) \rceil$

Example 2.4. Given

$$\tilde{\mathbf{s}} = \ulcorner \{f\}(\{a, b\}), \{f, g\}(x), \{h\}(\{b\}) \urcorner \text{ and } \tilde{\mathbf{t}} = \ulcorner \{f, h\}(\{b\}), \{f\}(x), \{h\}(\{c\}) \urcorner,$$

we will compute $\tilde{\mathbf{s}} \sqcap \tilde{\mathbf{t}}$:

- position 1: $\{f\}(\{a, b\}) \sqcap \{f, h\}(\{b\}) = \{f\}(\{b\})$
- position 2: $\{f, g\}(x) \sqcap \{f\}(x) = \{f\}(x)$
- position 3: $\{h\}(\{b\}) \sqcap \{h\}(\{c\}) = \emptyset$

therefore, this intersection is not successful and $\tilde{\mathbf{s}} \sqcap \tilde{\mathbf{t}} = \emptyset$.

2.3 Equations and Solved Sets

Equations between terms are pairs of term sequences, written as $\tilde{s} \approx \tilde{t}$. Equations between X-terms are defined analogously. By equation we mean a term equation or an X-term equation. A solved equation has one of the following forms: $x \approx s$, $x \approx \mathbf{s}$, $\bar{x} \approx \tilde{t}$ or $\bar{x} \approx \tilde{\mathbf{t}}$, where the right-hand side does not contain the variable in the left-hand side.

Next, we define some notions about solved equations. These are borrowed from [10].

Definition 2.4 (Solved Sets)

A solved set of equations is a set consisting of solved equations of the form $x \approx s$, $x \approx \mathbf{s}$, $\bar{x} \approx \tilde{t}$ or $\bar{x} \approx \tilde{\mathbf{t}}$ such that the variables in the left-hand side of solved equations occur in the entire set only once.

Definition 2.5 (Substitutions generated by solved equations)

Each solved equation generates a substitution or an X-substitution:

$$\begin{aligned} \Sigma(x \approx s) &= \{x \mapsto s\}, & \Sigma(x \approx \mathbf{s}) &= \{x \mapsto \mathbf{s}\}, \\ \Sigma(\bar{x} \approx \tilde{t}) &= \{\bar{x} \mapsto \tilde{t}\}, & \Sigma(\bar{x} \approx \tilde{\mathbf{t}}) &= \{\bar{x} \mapsto \tilde{\mathbf{t}}\}. \end{aligned}$$

Definition 2.6 (Substitutions generated by solved sets)

Substitutions and X-substitutions generated from solved sets are defined as follows:

$$\begin{aligned} \Sigma(\emptyset) &= \varepsilon, \\ \Sigma(\{eq_1, \dots, eq_n\}) &= \sigma_1 \cup \dots \cup \sigma_n \\ &\text{with } \sigma_i = \Sigma(eq_i) \text{ for } 1 \leq i \leq n, n \geq 0 \\ &\text{and } \{eq_1, \dots, eq_n\} \text{ being a solved set,} \end{aligned}$$

$$\Sigma(\{\mathbf{eq}_1, \dots, \mathbf{eq}_m\}) = \sigma_1 \cup \dots \cup \sigma_n$$

with $\sigma_j = \Sigma(\mathbf{eq}_j)$ for $1 \leq j \leq m$, $m \geq 0$
and $\{\mathbf{eq}_1, \dots, \mathbf{eq}_m\}$ being a solved set.

Let \mathbf{S} be a set of solved sets, then $\Sigma(\mathbf{S}) := \bigcup_{S \in \mathbf{S}} \Sigma(S)$ is the set of substitution generated from the set of solved sets.

2.4 Fuzzy Relations

This section establishes the basic notions of similarity and proximity relations as defined in [49] and [35].

A binary fuzzy relation \mathcal{R} on a set U is a mapping from $U \times U$ to the interval $[0, 1]$. The relation on U that is used to describe all pairs that fulfill a certain degree $\lambda \in (0, 1]$ is called the λ -cut and is denoted by $\mathcal{R}_\lambda = \{(a, b) \mid \mathcal{R}(a, b) \geq \lambda\}$, where λ is called the *cut value*.

Definition 2.7 (Proximity Relation)

A fuzzy relation \mathcal{R} on a set U is called a proximity relation if the following properties hold:

- *reflexivity*: $\mathcal{R}(x, x) = 1$ for any $x \in U$,
- *symmetry*: $\mathcal{R}(x, y) = \mathcal{R}(y, x)$ for any $x, y \in U$,
- *strictness*: $\mathcal{R}(x, y) = 1$ if $x = y$.

A *triangular norm* or *t-norm* $\wedge : [0, 1] \times [0, 1] \rightarrow [0, 1]$ is a binary operation, which is associative, commutative and non-decreasing, such that for any $x \in [0, 1]$, $x \wedge 1 = 1 \wedge x = x$. In this thesis, we will be using the minimum t-norm, due to its idempotent behaviour.

Definition 2.8 (Similarity Relation)

A fuzzy relation \mathcal{R} on a set U is called a similarity relation if the following properties hold:

- *reflexivity*: $\mathcal{R}(x, x) = 1$ for any $x \in U$,
- *symmetry*: $\mathcal{R}(x, y) = \mathcal{R}(y, x)$ for any $x, y \in U$,
- *transitivity*: $\mathcal{R}(x, y) \wedge \mathcal{R}(y, z) \leq \mathcal{R}(x, z)$ for any $x, y, z \in U$,
- *strictness*: $\mathcal{R}(x, y) = 1 \iff x = y$.

Hence, the only difference between proximity and similarity relations is the transitivity property: similarity relations are transitive proximity relations.

In order to relax the comparisons between terms from equality to similarity/proximity relations, we first need to define how those relations act on terms. We want to carry over the idea that two terms are related to a degree in the range of $[0, 1]$. So let \mathcal{R} be a similarity/proximity relation defined over the sets \mathcal{F} and \mathcal{V} . For symbols s_1 and s_2 , we require $\mathcal{R}(s_1, s_2) := 0$ if

- s_1 is a constant and s_2 is an unranked function symbol,
- $s_1 \in \mathcal{F}$ and $s_2 \notin \mathcal{F}$,
- $s_1, s_2 \in \mathcal{V}$ and $s_1 \neq s_2$.

Now let $s = f(s_1, \dots, s_n)$ and $t = g(t_1, \dots, t_n)$ be terms and define $\mathcal{R}(s, t)$ in the following way:

$$\mathcal{R}(f(s_1, \dots, s_n), g(t_1, \dots, t_n)) = \mathcal{R}(f, g) \wedge \left(\bigwedge_{i=1}^n \mathcal{R}(s_i, t_i) \right).$$

However if we have $s = f(s_1, \dots, s_n)$ and $t = g(t_1, \dots, t_m)$ with $n \neq m$, then $\mathcal{R}(s, t) = 0$. So for two terms $s = f(s_1, \dots, s_n)$ and $t = g(t_1, \dots, t_m)$ with flexible-arity function symbol as heads, $f, g \in \mathcal{F}_{\text{lex}}$, even though $\mathcal{R}(f, g) > 0$, because of the mismatch of arities in the term s and t it still holds that $\mathcal{R}(s, t) = 0$.

Two terms s and t are (\mathcal{R}, λ) -close, written as $s \simeq_{\mathcal{R}, \lambda} t$, if $\mathcal{R}(s, t) \geq \lambda$.

Given two sequences of terms $\tilde{s} = \ulcorner s_1, \dots, s_n \urcorner$ and $\tilde{t} = \ulcorner t_1, \dots, t_n \urcorner$, we define $\mathcal{R}(\tilde{s}, \tilde{t})$ as

$$\mathcal{R}(\tilde{s}, \tilde{t}) := \bigwedge_{i=1}^n \mathcal{R}(s_i, t_i).$$

It is clear that if $\tilde{s} = \ulcorner s_1, \dots, s_n \urcorner$ and $\tilde{t} = \ulcorner t_1, \dots, t_m \urcorner$ with $n \neq m$, then $\mathcal{R}(\tilde{s}, \tilde{t}) := 0$.

Definition 2.9

We define the degree of an (\mathcal{R}, λ) -equation $s \simeq_{\mathcal{R}, \lambda} t$ as $\deg_{\mathcal{R}}(s \simeq_{\mathcal{R}, \lambda} t) = \mathcal{R}(s, t)$. This notion is also extended to a finite set of (\mathcal{R}, λ) -equations Γ as $\deg_{\mathcal{R}}(\Gamma) = \bigwedge_{eq \in \Gamma} \deg_{\mathcal{R}}(eq)$.

An (\mathcal{R}, λ) -proximity class of an symbol $s \in U$, denoted by $\mathbf{pc}_{\mathcal{R}, \lambda}(s)$, is a set of all the elements of U which are ‘close’ to a s with a degree of at least λ under the proximity relation \mathcal{R} : $\mathbf{pc}_{\mathcal{R}, \lambda}(s) := \{t \mid \mathcal{R}(s, t) \geq \lambda\}$.

A (\mathcal{R}, λ) -proximity class of a term is defined as:

$$\begin{aligned} \mathbf{pc}_{\mathcal{R}, \lambda}(x) &:= \{x\} \\ \mathbf{pc}_{\mathcal{R}, \lambda}(\bar{x}) &:= \{\bar{x}\} \\ \mathbf{pc}_{\mathcal{R}, \lambda}(f(t_1, \dots, t_n)) &:= \mathbf{pc}_{\mathcal{R}, \lambda}(f)(\mathbf{pc}_{\mathcal{R}, \lambda}(t_1), \dots, \mathbf{pc}_{\mathcal{R}, \lambda}(t_n)). \end{aligned}$$

Note that by this definition, a proximity class is an X-term.

The proximity class for a sequence of terms $\tilde{s} = \lceil s_1, \dots, s_n \rceil$ is defined as

$$\mathbf{pc}_{\mathcal{R}, \lambda}(\tilde{s})_{\mathcal{R}, \lambda} = \lceil \mathbf{pc}_{\mathcal{R}, \lambda}(s_1)_{\mathcal{R}, \lambda}, \dots, \mathbf{pc}_{\mathcal{R}, \lambda}(s_n)_{\mathcal{R}, \lambda} \rceil.$$

It is possible to associate a function symbol f with a degree α in the form of a *graded function symbol* $\langle f, \alpha \rangle$ [34]. The intersection of two sets of graded function symbols \mathbf{f}_1 and \mathbf{f}_2 is defined as

$$\mathbf{f}_1 \cap \mathbf{f}_2 := \{\langle f, \alpha \wedge \beta \rangle \mid \langle f, \alpha \rangle \in \mathbf{f}_1, \langle f, \beta \rangle \in \mathbf{f}_2\},$$

equivalent to a regular intersection of sets of function symbols, but taking into account the degrees. The notion of X-term can be extended to graded X-terms, where instead of sets of symbols, we have graded sets of symbols.

A *graded (\mathcal{R}, λ) -proximity class* for a symbol s is defined as

$$\mathbf{gpc}_{\mathcal{R}, \lambda}(s) = \{\langle t, \alpha \rangle \mid \mathcal{R}(s, t) = \alpha \geq \lambda\}.$$

For a term, the *graded (\mathcal{R}, λ) -proximity class* is constructed as

$$\begin{aligned} \mathbf{gpc}_{\mathcal{R}, \lambda}(x) &:= \{\langle x, 1 \rangle\} \\ \mathbf{gpc}_{\mathcal{R}, \lambda}(\bar{x}) &:= \{\langle \bar{x}, 1 \rangle\} \\ \mathbf{gpc}_{\mathcal{R}, \lambda}(f(t_1, \dots, t_n)) &:= \mathbf{gpc}_{\mathcal{R}, \lambda}(f)(\mathbf{gpc}_{\mathcal{R}, \lambda}(t_1), \dots, \mathbf{gpc}_{\mathcal{R}, \lambda}(t_n)). \end{aligned}$$

using again the graded (\mathcal{R}, λ) -proximity classes for the deconstructed term. Note that graded proximity classes are graded X-terms.

Applying terms and subs to graded X-terms and graded X-substitutions results in sets of graded terms and substitutions respectively. Given a graded X-term $\mathbf{f}(\mathbf{s}_1, \dots, \mathbf{s}_n)$, then:

$$\begin{aligned} \text{terms}(\mathbf{f}(\mathbf{s}_1, \dots, \mathbf{s}_n)) &:= \\ &\{\langle f(s_1, \dots, s_n), \alpha \rangle \mid \langle f, \alpha_0 \rangle \in \mathbf{f}, \langle s_i, \alpha_i \rangle \in \text{terms}(\mathbf{s}_i), 1 \leq i \leq n, \alpha = \alpha_0 \wedge \dots \wedge \alpha_n\} \end{aligned}$$

and given a graded substitution $\{x_1 \mapsto \mathbf{s}_1, \dots, x_n \mapsto \mathbf{s}_n\}$, then

$$\text{subs}(\{x_1 \mapsto \mathbf{s}_1, \dots, x_n \mapsto \mathbf{s}_n\}) := \{ \langle \{x_1 \mapsto s_1, \dots, x_n \mapsto s_n\}, \alpha \rangle \mid \langle s_i, \alpha_i \rangle \in \text{terms}(\mathbf{s}_i), 1 \leq i \leq n, \alpha = \alpha_1 \wedge \dots \wedge \alpha_n \}.$$

Example 2.5. Let \mathcal{R} be a proximity relation with $\mathcal{R}(f, g) = 0.9$, $\mathcal{R}(f, h) = 0.8$, $\mathcal{R}(a, c) = 0.6$ and $\mathcal{R}(b, c) = 0.7$, and $t = f(g(a, b), h(c))$ be a term. Then the graded proximity class of t is

$$\mathbf{pc}_{\mathcal{R}, \lambda}(t) = \{ \langle f, 1 \rangle, \langle g, 0.9 \rangle \} \{ \langle g, 1 \rangle \} \{ \langle a, 1 \rangle, \langle c, 0.6 \rangle \}, \{ \langle b, 1 \rangle, \langle c, 0.7 \rangle \}, \{ \langle h, 1 \rangle, \langle f, 0.8 \rangle \} \{ \langle c, 1 \rangle, \langle a, 0.6 \rangle, \langle b, 0.7 \rangle \}.$$

Example 2.6. Given the same proximity relation \mathcal{R} and term t as in example 2.5 and an additional term $s = h(f(a, c), f(b))$ with

$$\mathbf{pc}_{\mathcal{R}, \lambda}(s) = \{ \langle h, 1 \rangle, \langle f, 0.8 \rangle \} \{ \langle f, 1 \rangle, \langle g, 0.9 \rangle, \langle h, 0.8 \rangle \} \{ \langle a, 1 \rangle, \langle c, 0.6 \rangle \}, \{ \langle c, 1 \rangle, \langle a, 0.6 \rangle \}, \{ \langle f, 1 \rangle, \langle g, 0.9 \rangle, \langle h, 0.8 \rangle \} \{ \langle b, 1 \rangle, \langle c, 0.7 \rangle \},$$

we can now compute the graded intersection $\mathbf{pc}_{\mathcal{R}, \lambda}(s) \sqcap \mathbf{pc}_{\mathcal{R}, \lambda}(t)$ of the graded proximity classes:

$$\mathbf{pc}_{\mathcal{R}, \lambda}(s) \sqcap \mathbf{pc}_{\mathcal{R}, \lambda}(t) = \{ \langle f, 0.8 \rangle \} \{ \langle g, 0.9 \rangle \} \{ \langle a, 1 \rangle, \langle c, 0.6 \rangle \}, \{ \langle c, 0.7 \rangle \}, \{ \langle f, 0.8 \rangle, \langle h, 0.8 \rangle \} \{ \langle b, 0.7 \rangle, \langle c, 0.7 \rangle \}.$$

Extracting the graded terms from this graded X-term gives us the set

$$\begin{aligned} \text{terms}(\mathbf{pc}_{\mathcal{R}, \lambda}(s) \sqcap \mathbf{pc}_{\mathcal{R}, \lambda}(t)) = & \{ \langle f(g(a, c), f(b)), 0.7 \rangle, \langle f(g(a, c), f(c)), 0.7 \rangle, \\ & \langle f(g(a, c), h(b)), 0.7 \rangle, \langle g(g(a, c), h(c)), 0.7 \rangle, \\ & \langle f(g(c, c), f(b)), 0.7 \rangle, \langle f(g(c, c), f(c)), 0.7 \rangle, \\ & \langle f(g(c, c), h(b)), 0.7 \rangle, \langle g(g(c, c), h(c)), 0.7 \rangle \}. \end{aligned}$$

It is also possible to extend the notion of ‘more general’ from a syntactic relation to a fuzzy relation:

Definition 2.10

A substitution σ is (\mathcal{R}, λ) -equal to a substitution θ on a finite set of variables \mathcal{X} , denoted as $\sigma \simeq_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$, if

- for all $\bar{x} \in \mathcal{X}$ there exists $s_1, \dots, s_n, t_1, \dots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, $n \geq 0$, such that $\bar{x}\sigma = \ulcorner s_1, \dots, s_n \urcorner \simeq_{\mathcal{R}, \lambda} \ulcorner t_1 \dots, t_n \urcorner = \bar{x}\theta$;

- for all $x \in \mathcal{X}$, $x\sigma \simeq_{\mathcal{R},\lambda} x\theta$.

Definition 2.11

A substitution σ is called (\mathcal{R}, λ) -more general than a substitution θ on a set of variables \mathcal{X} , if there exists a substitution ν such that $\sigma\nu \simeq_{\mathcal{R},\lambda}^{\mathcal{X}} \theta$. We then write $\sigma \lesssim_{\mathcal{R},\lambda}^{\mathcal{X}} \theta$, and θ is called an (\mathcal{R}, λ) -instance of σ .

If it holds that $\sigma \lesssim_{\mathcal{R},\lambda}^{\mathcal{X}} \theta$ and $\theta \lesssim_{\mathcal{R},\lambda}^{\mathcal{X}} \sigma$ then σ and θ are equigeneral, written as $\sigma \approx_{\mathcal{R},\lambda}^{\mathcal{X}} \theta$.

It is also possible to compare terms in the same way: A term s is said to be (\mathcal{R}, λ) -more general than a term t , or s is an (\mathcal{R}, λ) -generalization of t , if there exists a substitution σ such that $s\sigma \simeq_{\mathcal{R},\lambda} t$. This relation is written as $s \lesssim_{\mathcal{R},\lambda} t$. We can additionally say that s is an $(\mathcal{R}, \lambda)_\alpha$ -generalization of t if $\mathcal{R}(s\sigma, t) = \alpha$.

If \mathcal{R} is not transitive (like with a proximity relation) then $\lesssim_{\mathcal{R},\lambda} t$ is also not transitive and therefore not a quasi-ordering.

2.5 Ordering

We will need a well-founded ordering on multisets in order to prove the termination of some algorithms. For this we will use the Dershowitz-Manna ordering from [9].

Definition 2.12 (Multiset Ordering)

Given a set S and a well founded partial ordering $<$ on S . The Dershowitz-Manna Ordering $<_{DM}$ is defined on the set of all finite multisets $\mathcal{M}(S)$ such that for $M, N \in \mathcal{M}(S)$, $M <_{DM} N$ if $M = (N - X) + Y$ for some $X, Y \in \mathcal{M}(S)$ with $\emptyset \neq X \subseteq N$ and for all $y \in Y$ there exists an $x \in X$ such that $y < x$.

To denote multisets we will use the following notation: $M = \{\!\!\{ \cdot \}\!\!\}$.

2.6 Unification, Matching, Anti-Unification

In this section, we explain the basic principles of unification, matching and anti-unification.

2.6.1 Unification

The fundamental purpose of unification is to solve equations between syntactic expressions. Solutions are substitutions which, after applying to the given expressions, turn them into equal ones. Given two terms s and t , a *unification problem* is written as $s \simeq^? t$. A substitution which solves a unification problem is called a *unifier* of the terms. If a unification problem is solvable, it may have infinitely many solutions. Among them, the interesting ones are those that are more general than the others. For some problems, there exists a single such unifier, which is unique modulo variable renaming. It is called a most general unifier (mgu). Such problems are called *unitary*.

An example for a unification problem could be to unify $s = f(x, g(g(z)))$ and $t = f(a, g(y))$. There are infinitely many unifiers of this problem, e.g.,

$$\{x \mapsto a, y \mapsto g(z)\}$$

$$\{x \mapsto a, y \mapsto g(a), z \mapsto a\}$$

$$\{x \mapsto a, y \mapsto g(b), z \mapsto b\}$$

$$\{x \mapsto a, y \mapsto g(g(a)), z \mapsto g(a)\}$$

$$\{x \mapsto a, y \mapsto g(z), u \mapsto a\} \text{ (a completely irrelevant variable } u \text{ in the substitution)}$$

etc.

However, the substitution $\{x \mapsto a, y \mapsto g(z)\}$ is more general than the others. In fact, it is the most general unifier of the problem, which is unitary.

There are problems which have finitely or infinitely many unifiers that are not comparable between themselves with respect to the instantiation relation, but any other unifier is less general than some of these unifiers. They are called *finitary* and *infinitary* problems, respectively. Such problems are characterized by *minimal complete sets of unifiers*: They are sets of unifiers of the given problem such that (a) for any unifier of the problem, one can find a more general one in this set, and (b) no two distinct elements of the set are comparable with respect to the instantiation relation. Then finitary (resp. infinitary) problems have a finite (resp. infinite) minimal complete set of unifiers. There exist theories and problems for which such a minimal complete set of unifiers does not exist because minimality and completeness contradict each other. For our unranked languages, every solvable unification problem does have a minimal complete set of unifiers, but it can be infinite in some cases.

Example 2.7.

$$f(\bar{x}, a, \bar{y}) \simeq^? f(a, z)$$

This problem is finitary. The minimal complete set of unifiers contains two elements:

$$\{\bar{x} \mapsto \ulcorner, \bar{y} \mapsto z\} \text{ and } \{\bar{x} \mapsto a, z \mapsto a, \bar{y} \mapsto \ulcorner\}$$

$$f(\bar{x}, a) \simeq? f(a, \bar{x})$$

An infinitary problem. The minimal complete set of unifiers has the following elements:

$$\{\bar{x} \mapsto \ulcorner\}, \{\bar{x} \mapsto a\}, \{\bar{x} \mapsto \ulcorner a, a \urcorner\}, \dots$$

Sequences are allowed in unranked unification equations, i.e., $\tilde{s} \simeq? \tilde{t}$ is a valid unranked unification problem.

2.6.2 Matching

Matching is a specialized form of unification. Whereas in a unification problem both sides can contain variables, in a matching problem only one (typically, the left) side contains variables, whilst the other one is ground. This means a solution, called a matcher, has to be applied only to the left term in order to match it to the right, as opposed to a unifier which has to be applied to both sides to create a new, unified term. Given two terms s and t , with t being ground, a *matching problem* is written as $s \ll? t$.

Unranked matching equations are pairs of sequences, written as $\tilde{s} \ll? \tilde{t}$, where \tilde{t} is ground.

Unranked matching is finitary. For instance, the matching problem $f(\bar{x}, a, \bar{y}) \simeq? f(a, a)$ has two incomparable matchers: $\{\bar{x} \mapsto \ulcorner, \bar{y} \mapsto a\}$ and $\{\bar{x} \mapsto a, \bar{y} \mapsto \ulcorner\}$.

2.6.3 Anti-Unification

Anti-unification can be seen as a kind of dual problem to unification. Whereas the aim of unification is to find the most general common instance of two terms s and t (via applying their most general unifier to them), anti-unification aims to compute the least general term whose instances are s and t , i.e., the task is to find the least general common generalization (lgg) of s and t . A more concrete example: given two terms $s = f(a, f(b, b))$ and $t = f(a, f(c, c))$, we want to find a generalization for them. Of course, a trivial generalization would be a new variable, e.g., y , as y can be mapped to both s and t . However, that is a very general generalization. We aim to find the *least general generalization* or *lgg*. The $lgg(s, t)$ would be $f(a, f(y, y))$. Here we have preserved as much of the shared structure and generalized the differences in the uniform way (notice the two occurrences of y). It is easy to see that y can be mapped to b and c to recreate s and t , respectively.

The following notions are based on [30]. An *anti-unification triple* (AUT) is the triple $\mathbf{x} : \tilde{s} \triangleq \tilde{t}$, where if \mathbf{x} is an individual variable, then \tilde{s} and \tilde{t} are individual terms, and \mathbf{x} does not occur in them. The intuition is that \mathbf{x} , called the generalization variable, represents the (most general)

generalization of \tilde{s} and \tilde{t} . An *anti-unifier*, or a solution of an AUT $\mathbf{x} : \tilde{s} \triangleq \tilde{t}$ is a substitution σ such that $\mathbf{x}\sigma$ is a generalization of \tilde{s} and \tilde{t} .

Similar to unification and matching, anti-unification problems may have one or more solutions. For our unranked language, AUTs are always solvable, and they may have at most finitely many incomparable generalizations. The notion of a minimal complete set of generalizations is defined analogously, but dually, to a minimal complete set of unifiers: its elements are those generalizations that are less general than other generalizations of the given problem.

Looking back at our small example: The lgg of $x : f(a, f(b, b)) \triangleq f(a, f(c, c))$ would be $\sigma = \{x \mapsto f(a, f(y, y))\}$ with $x\sigma = f(a, f(y, y))$ being an lgg of $f(a, f(b, b))$ and $f(a, f(c, c))$.

An unranked generalization problem $x : f(f(a), f(a)) \triangleq f(f(a), f)$ has three elements in the minimal complete set of generalizations: $\{x \mapsto f(f(a), f(\bar{y}))\}$, $\{x \mapsto f(f(\bar{y}, \bar{z}), f(\bar{y}))\}$, and $\{x \mapsto f(f(\bar{y}, \bar{z}), f(\bar{z}))\}$.

3 Unification

In this chapter, the procedure used to unify unranked sequence problems introduced by Temur Kutsia in *Solving equations with sequence variables and sequence functions* [30] will be modified to work with similarity and proximity relations instead of syntactic equality.

In addition to the concept of *more general* in substitutions, we need a slight distinction if a substitution maps a sequence variable to an empty sequence:

Definition 3.1

A substitution σ is called *erasing* on a finite set of variables \mathcal{X} if there exist $\bar{x} \in \mathcal{X}$ such that $\bar{x}\sigma = \square$. Otherwise, it is *non-erasing*.

Definition 3.2

If a substitution σ is syntactically more general than θ on a set of variables \mathcal{X} (i.e., if $\sigma \preceq^{\mathcal{X}} \theta$) and there exists a substitution ν , non-erasing on \mathcal{X} , such that $\sigma\nu =^{\mathcal{X}} \theta$, then σ is called *syntactically strongly more general than θ* . It is written as $\sigma \triangleleft^{\mathcal{X}} \theta$.

The same concept can be defined in a fuzzy setting as well. If a substitution σ is (\mathcal{R}, λ) -more general than θ (i.e., if $\sigma \preceq_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$) and there exists a substitution ν , non-erasing on \mathcal{X} , such that $\sigma\nu \simeq_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$, then σ is called (\mathcal{R}, λ) -strongly more general than θ . It is written as $\sigma \triangleleft_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$.

The notion of a syntactic minimal complete set of unifiers is defined as follows:

Definition 3.3

The set of all syntactic unifiers of an unranked syntactic unification problem Γ is denoted by $\mathcal{U}(\Gamma)$. A set of substitutions S is a *complete set of syntactic unifiers* of Γ for a set of variables \mathcal{X} if

- $S \subseteq \mathcal{U}(\Gamma)$, i.e., each element of S is a syntactic unifier of Γ , and
- for any syntactic unifier θ of Γ , there exists $\sigma \in S$ such that $\sigma \preceq^{\mathcal{X}} \theta$.

The set S is, in addition, *minimal*, if no two distinct elements of it are $\preceq^{\mathcal{X}}$ -comparable.

To make this work self-contained, we first recall the original syntactic unranked unification algorithm \mathfrak{U} from [30] and then show how it can be modified to deal with proximity and similarity. The algorithm \mathfrak{U} works on the following inference rules \mathfrak{J} below. Note that the rules concerning sequence function symbols are not shown here, since we do not consider those symbols.

P-U: Projection

$$\langle \Gamma; \sigma \rangle \Longrightarrow \langle \Gamma\vartheta; \sigma\vartheta \rangle, \quad \text{where } \vartheta \neq \varepsilon, \text{ dom}(\vartheta) \subseteq \mathcal{V}_S(\Gamma) \text{ and } \text{ran}(\vartheta) = \emptyset.$$

T-U: Trivial

$$\langle \{s =^? s\} \cup \Gamma'; \sigma \rangle \Longrightarrow \langle \Gamma'; \sigma \rangle.$$

O1-U: Orient 1

$$\langle \{s =^? x\} \cup \Gamma'; \sigma \rangle \Longrightarrow \langle \{x =^? s\} \cup \Gamma'; \sigma \rangle, \quad \text{if } s \notin \mathcal{V}_T.$$

O2-U: Orient 2

$$\begin{aligned} & \langle \{f(s, s_1, \dots, s_n) =^? f(\bar{x}, t_1, \dots, t_m)\} \cup \Gamma'; \sigma \rangle \\ & \Longrightarrow \langle \{f(\bar{x}, t_1, \dots, t_m) =^? f(s, s_1, \dots, s_n)\} \cup \Gamma'; \sigma \rangle, \\ & \text{if } s \notin \mathcal{V}_S. \end{aligned}$$

TD-U: Total Decomposition

$$\begin{aligned} & \langle \{f(s_1, \dots, s_n) =^? f(t_1, \dots, t_n)\} \cup \Gamma'; \sigma \rangle \Longrightarrow \langle \{s_1 =^? t_1, \dots, s_n =^? t_n\} \cup \Gamma'; \sigma \rangle, \\ & \text{if } f(s_1, \dots, s_n) \neq f(t_1, \dots, t_n) \text{ and } s_i, t_i \notin \mathcal{V}_S \text{ for all } 1 \leq i \leq n. \end{aligned}$$

IVE-U: Individual Variable Elimination

$$\langle \{x =^? t\} \cup \Gamma'; \sigma \rangle \Longrightarrow \langle \Gamma'\vartheta; \sigma\vartheta \rangle, \quad \text{if } x \notin \mathcal{V}_T(t) \text{ and } \vartheta = \{x \mapsto t\}.$$

PD-U: Partial Decomposition

$$\begin{aligned} & \langle \{f(s_1, \dots, s_n) =^? f(t_1, \dots, t_m)\} \cup \Gamma'; \sigma \rangle \\ & \Longrightarrow \langle \{s_1 =^? t_1, \dots, s_{k-1} =^? t_{k-1}, f(s_k, \dots, s_n) =^? f(t_k, \dots, t_m)\} \cup \Gamma'; \sigma \rangle, \\ & \text{if } f(s_1, \dots, s_n) \neq f(t_1, \dots, t_m), \text{ for some } 1 < k \leq \min(n, m), \text{ either } s_k \in \mathcal{V}_S \text{ or } t_k \in \mathcal{V}_S, \\ & \text{and for all } 1 \leq i < k \text{ we have } s_i, t_i \notin \mathcal{V}_S. \end{aligned}$$

SVE1-U: Sequence Variable Elimination 1

$$\begin{aligned} & \langle \{f(\bar{x}, s_1, \dots, s_n) =^? f(\bar{x}, t_1, \dots, t_m)\} \cup \Gamma'; \sigma \rangle \\ & \Longrightarrow \langle \{f(s_1, \dots, s_n) =^? f(t_1, \dots, t_m)\} \cup \Gamma'; \sigma \rangle, \\ & \text{if } f(\bar{x}, s_1, \dots, s_n) \neq f(\bar{x}, t_1, \dots, t_m). \end{aligned}$$

SVE2-U: Sequence Variable Elimination 2

$$\begin{aligned} & \langle \{f(\bar{x}, s_1, \dots, s_n) =^? f(t, t_1, \dots, t_m)\} \cup \Gamma'; \sigma \rangle \\ & \Longrightarrow \langle \{f(s_1, \dots, s_n) =^? f(t_1, \dots, t_m)\} \cup \Gamma'\vartheta; \sigma\vartheta \rangle, \\ & \text{if } \bar{x} \notin \mathcal{V}_S(t) \text{ and } \vartheta = \{\bar{x} \mapsto t\}. \end{aligned}$$

W1-U: Widening 1

$$\begin{aligned}
 & \langle \{f(\bar{x}, s_1, \dots, s_n) \stackrel{?}{=} f(t, t_1, \dots, t_m)\} \cup \Gamma'; \sigma \rangle \\
 & \implies \langle \{f(\bar{x}, s_1\vartheta, \dots, s_n\vartheta) \stackrel{?}{=} f(t_1\vartheta, \dots, t_m\vartheta)\} \cup \Gamma'\vartheta; \sigma\vartheta \rangle, \\
 & \text{if } \bar{x} \notin \mathcal{V}_S(t) \text{ and } \vartheta = \{\bar{x} \mapsto \lceil t, \bar{x} \rceil\}.
 \end{aligned}$$

W2-U: Widening 2

$$\begin{aligned}
 & \langle \{f(\bar{x}, s_1, \dots, s_n) \stackrel{?}{=} f(\bar{y}, t_1, \dots, t_m)\} \cup \Gamma'; \sigma \rangle \\
 & \implies \langle \{f(s_1\vartheta, \dots, s_n\vartheta) \stackrel{?}{=} f(\bar{y}, t_1\vartheta, \dots, t_m\vartheta)\} \cup \Gamma'\vartheta; \sigma\vartheta \rangle, \\
 & \text{with } \vartheta = \{\bar{y} \mapsto \lceil \bar{x}, \bar{y} \rceil\}.
 \end{aligned}$$

These rules are applied to a configuration of the form $\langle \Gamma; \sigma \rangle$, where Γ is a set of unification problems and σ is a substitution built along the way to be the unifier of Γ . In the starting configuration, σ is set to ε . Through a *selection strategy*, an equation is chosen from Γ to be transformed in the next step. By transforming configurations step by step (guided by the selection strategy), a *derivation* of the form $\langle \Gamma_1; \sigma_1 \rangle \implies_{P/BT} \langle \Gamma_2; \sigma_2 \rangle \implies_{BT}^+ \langle \Gamma_n; \sigma_n \rangle$ is constructed, with BT being *basic transformations*, i.e., any rule except the projection rule. Branching can occur by either the possibility to apply a rule in different ways, or by having multiple rules applicable to the selected equation. A derivation either stops via a decision algorithm, recognising when the given problem is not solvable, resulting in failure written as \perp , or the final pair $\langle \emptyset; \theta \rangle$, where θ is a substitution solving the original unification problem Γ . The decision algorithm relies on decidability of word equations, which is quite a difficult problem [27, 38, 45]. It is possible that some branches produce a solution, whereas others result in failure. What we get, is a finitely branching potentially infinite tree of derivations. The set of substitutions produced by the procedure \mathfrak{U} is written as $Sol_{\mathfrak{U}}(\Gamma)$. This set can be empty (when the problem is not solvable), finite, or infinite.

Since we will not use a decision algorithm to check whether our configurations are solvable, a ‘lighter’ version of this unification procedure, also introduced in [29], is more appropriate for us. It has the following extra rules, which take over the role of detecting failure cases.

IVOC-U: Individual Variable Occurrence Check

$$\langle \{x \stackrel{?}{=} t\} \cup \Gamma'; \sigma; \alpha \rangle \implies \perp, \quad \text{if } x \neq t \text{ and } x \in \mathcal{V}_T(t).$$

SVOC-U: Sequence Variable Occurrence Check

$$\langle \{f(\bar{x}, s_1, \dots, s_n) \stackrel{?}{=} f(t, t_1, \dots, t_m)\} \cup \Gamma'; \sigma; \alpha \rangle \implies \perp, \quad \text{if } \bar{x} \neq t \text{ and } \bar{x} \in \mathcal{V}_S(t).$$

SC-U: Symbol Clash

$$\langle \{f(s_1, \dots, s_n) \stackrel{?}{=} g(t_1, \dots, t_m)\} \cup \Gamma'; \sigma; \alpha \rangle \implies \perp, \quad \text{if } f \neq g.$$

AD: Arity Disagreement

$$\begin{aligned}
 & \langle \{f(s_1, \dots, s_n) \stackrel{?}{=} f(t_1, \dots, t_m)\} \cup \Gamma'; \sigma; \alpha \rangle \implies \perp, \\
 & \text{if } n \neq m \text{ and } s_i \notin \mathcal{V}_S \text{ and } t_j \notin \mathcal{V}_S \text{ for all } 1 \leq i \leq n \text{ and } 1 \leq j \leq m.
 \end{aligned}$$

E1: **Empty 1**

$$\langle \{f() \stackrel{?}{=} f(t, t_1, \dots, t_n)\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow \perp.$$

E2: **Empty 2**

$$\langle \{f(s, s_1, \dots, s_n) \stackrel{?}{=} f()\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow \perp.$$

Due to the fact that these rules cannot detect every failure case, the procedure without the decision algorithms is not guaranteed to terminate even if there is no solution, at least not without further modifications. This problem will transfer to unranked unification both with similarity and proximity relations. One example of an unsolvable unification problem, which would not terminate with \mathfrak{U} , would be $f(\bar{x}) \stackrel{?}{=} f(a, \bar{x})$.

There are some special cases when the termination of the ‘light’ procedure is guaranteed. One such way, is e.g., to demand that one side of the equations has to be ground, i.e., devoid of any variables, which would just be a matching problem. Another way is to restrict the positions where sequence variables can occur to the last position within terms and sequences of terms, so called KIF (Knowledge Interchange Format) fragment, where it was first considered [16]. Some other terminating fragments have been considered in [32].

3.1 Fuzzy Unranked Unification with Similarity Relations

We start with the definition of a minimal complete set of unifiers modulo similarity relations:

Definition 3.4 (Minimal complete set of (\mathcal{R}, λ) -unifiers for similarity relations)

Let \mathcal{R} be a similarity relation, λ be a cut value, and Γ be a (\mathcal{R}, λ) -unification problem. The set of all unifiers of Γ is denoted by $\mathcal{U}(\Gamma, \mathcal{R}, \lambda)$. (Hence, $\sigma \in \mathcal{U}(\Gamma, \mathcal{R}, \lambda)$ iff $\deg_{\mathcal{R}}(\Gamma\sigma) \geq \lambda$.)

We also write $\sigma \in \mathcal{U}(\Gamma, \mathcal{R}, \lambda)_{\alpha}$ if $\deg_{\mathcal{R}}(\Gamma\sigma) = \alpha \geq \lambda$.

We say that a set of substitutions S is a complete set of (\mathcal{R}, λ) -unifiers of Γ for a set of variables \mathcal{X} if

- $S \subseteq \mathcal{U}(\Gamma, \mathcal{R}, \lambda)$, i.e., each element of S is an (\mathcal{R}, λ) -unifier of Γ , and
- for any (\mathcal{R}, λ) -unifier θ of Γ , there exists $\sigma \in S$ such that $\sigma \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$.

The set S is, in addition, minimal, if no two distinct elements of it are $\lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}}$ -comparable.

In order to accommodate a fuzzy relation instead of syntactic equivalence and in order to track the degree of similarity throughout our unification problem, we need to make some changes to the rules of the unification procedure \mathfrak{U} . These changes were inspired by [49] and [44]. It is also worth mentioning that in contrast to the syntactic version of this procedure

we will allow for equations of the form $\bar{x} \simeq_{\mathcal{R},\lambda}^? \tilde{s}$, meaning equations between sequences of terms instead of demanding the sequences of terms to be the arguments of a function symbol.

P-U_{sim}: Projection

$$\langle \Gamma; \sigma; \alpha \rangle \Longrightarrow \langle \Gamma\vartheta; \sigma\vartheta; \alpha \rangle$$

with $\vartheta \neq \varepsilon$, $\text{dom}(\vartheta) \subseteq \mathcal{V}_S(\Gamma)$ and $\text{ran}(\vartheta) = \emptyset$.

T-U_{sim}: Trivial

$$\langle \{\tilde{s} \simeq_{\mathcal{R},\lambda}^? \tilde{s}\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow \langle \Gamma'; \sigma; \alpha \rangle.$$

O1-U_{sim}: Orient 1

$$\langle \{s \simeq_{\mathcal{R},\lambda}^? x\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow \langle \{x \simeq_{\mathcal{R},\lambda}^? s\} \cup \Gamma'; \sigma; \alpha \rangle$$

if $s \notin \mathcal{V}_I$.

O2-U_{sim}: Orient 2

$$\langle \{\ulcorner s, s_1, \dots, s_n \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner \bar{x}, t_1, \dots, t_m \urcorner\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow$$

$$\langle \{\ulcorner \bar{x}, t_1, \dots, t_m \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner s, s_1, \dots, s_n \urcorner\} \cup \Gamma'; \sigma; \alpha \rangle$$

if $s \notin \mathcal{V}_S$.

HR-U_{sim}: Head Removal

$$\langle \{f(s_1, \dots, s_n) \simeq_{\mathcal{R},\lambda}^? g(t_1, \dots, t_m)\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow$$

$$\langle \{\ulcorner s_1, \dots, s_n \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner t_1, \dots, t_m \urcorner\} \cup \Gamma'; \sigma; \alpha \wedge \beta \rangle$$

if $f(s_1, \dots, s_n) \neq g(t_1, \dots, t_m)$ and $\mathcal{R}(f, g) = \beta \geq \lambda$.

IVE-U_{sim}: Individual Variable Elimination

$$\langle \{x \simeq_{\mathcal{R},\lambda}^? t\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow \langle \Gamma'\vartheta; \sigma\vartheta; \alpha \rangle$$

if $x \notin \mathcal{V}_I(t)$ and $\vartheta = \{x \mapsto t\}$.

DEC-U_{sim}: Decomposition

$$\langle \{\ulcorner s, s_1, \dots, s_n \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner t, t_1, \dots, t_m \urcorner\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow$$

$$\langle \{s \simeq_{\mathcal{R},\lambda}^? t, \ulcorner s_1, \dots, s_n \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner t_1, \dots, t_m \urcorner\} \cup \Gamma'; \sigma; \alpha \rangle$$

if $\ulcorner s, s_1, \dots, s_n \urcorner \neq \ulcorner t, t_1, \dots, t_m \urcorner$, $n + m > 0$, $s \notin \mathcal{V}_S$ and $t \notin \mathcal{V}_S$.

SVE1-U_{sim}: Sequence Variable Elimination 1

$$\langle \{\ulcorner \bar{x}, s_1, \dots, s_n \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner \bar{x}, t_1, \dots, t_m \urcorner\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow$$

$$\langle \{\ulcorner s_1, \dots, s_n \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner t_1, \dots, t_m \urcorner\} \cup \Gamma'; \sigma; \alpha \rangle$$

if $\ulcorner \bar{x}, s_1, \dots, s_n \urcorner \neq \ulcorner \bar{x}, t_1, \dots, t_m \urcorner$.

SVE2-U_{sim}: Sequence Variable Elimination 2

$$\langle \{\ulcorner \bar{x}, s_1, \dots, s_n \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner t, t_1, \dots, t_m \urcorner\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow$$

$$\langle \{\ulcorner s_1, \dots, s_n \urcorner \vartheta \simeq_{\mathcal{R},\lambda}^? \ulcorner t_1, \dots, t_m \urcorner \vartheta\} \cup \Gamma'\vartheta; \sigma\vartheta; \alpha \rangle$$

if $\bar{x} \notin \mathcal{V}_S(t)$ and $\vartheta = \{\bar{x} \mapsto t\}$.

W1-U_{sim}: Widening 1

$$\begin{aligned} & \langle \{\lceil \bar{x}, s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t, t_1, \dots, t_m \rceil\} \cup \Gamma'; \sigma; \alpha \rangle \implies \\ & \langle \{\lceil \bar{x}, s_1 \vartheta, \dots, s_n \vartheta \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t_1 \vartheta, \dots, t_m \vartheta \rceil\} \cup \Gamma \vartheta; \sigma \vartheta; \alpha \rangle \\ & \text{if } \bar{x} \notin \mathcal{V}_S(t) \text{ and } \vartheta = \{\bar{x} \mapsto \lceil t, \bar{x} \rceil\}. \end{aligned}$$

W2-U_{sim}: Widening 2

$$\begin{aligned} & \langle \{\lceil \bar{x}, s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \bar{y}, t_1, \dots, t_m \rceil\} \cup \Gamma'; \sigma; \alpha \rangle \implies \\ & \langle \{\lceil s_1 \vartheta, \dots, s_n \vartheta \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \bar{y}, t_1 \vartheta, \dots, t_m \vartheta \rceil\} \cup \Gamma' \vartheta; \sigma \vartheta; \alpha \rangle \\ & \text{with } \vartheta = \{\bar{y} \mapsto \lceil \bar{x}, \bar{y} \rceil\}. \end{aligned}$$

IVOC-U_{sim}: Individual Variable Occurrence Check

$$\begin{aligned} & \langle \{x \simeq_{\mathcal{R}, \lambda}^? t\} \cup \Gamma'; \sigma; \alpha \rangle \implies \perp \\ & \text{if } x \in \mathcal{V}_I(t) \text{ and } x \neq t. \end{aligned}$$

SVOC-U_{sim}: Sequence Variable Occurrence Check

$$\begin{aligned} & \langle \{\lceil \bar{x}, s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t, t_1, \dots, t_m \rceil\} \cup \Gamma'; \sigma; \alpha \rangle \implies \perp \\ & \text{if } \bar{x} \in \mathcal{V}_S(t) \text{ and } \bar{x} \neq t. \end{aligned}$$

SC-U_{sim}: Symbol Clash

$$\begin{aligned} & \langle \{f(s_1, \dots, s_n) \simeq_{\mathcal{R}, \lambda}^? g(t_1, \dots, t_m)\} \cup \Gamma'; \sigma; \alpha \rangle \implies \perp \\ & \text{if } \mathcal{R}(f, g) < \lambda. \end{aligned}$$

E1-U_{sim}: Empty 1

$$\begin{aligned} & \langle \{\lceil \lceil \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t, t_1, \dots, t_n \rceil\} \cup \Gamma'; \sigma; \alpha \rangle \implies \perp \\ & \text{with } t \notin \mathcal{V}_S. \end{aligned}$$

E2-U_{sim}: Empty 2

$$\begin{aligned} & \langle \{\lceil s, s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \lceil \rceil\} \cup \Gamma'; \sigma; \alpha \rangle \implies \perp \\ & \text{with } s \notin \mathcal{V}_S. \end{aligned}$$

Given a similarity relation \mathcal{R} and a cut value λ , these modified rules of the unranked fuzzy unification algorithm with similarity relations \mathcal{U}_{sim} are now applied to a triple of the form $\langle \Gamma; \sigma; \alpha \rangle$, where Γ is a set of (\mathcal{R}, λ) -unification problems, σ is a substitution tasked with building the unifier of Γ , and α being the new addition meant to track the degree of unification within the problem.

Aside from the addition of α , changes have occurred in a number of rules compared to [29], specifically those handling sequence variables. Whereas before, these rules were applied to individual terms with the sequence variable being the first argument, now it is applied to sequences of terms with the sequence variable being the first element of the sequence. The rules affected by this change are (O2-U_{sim}), (DEC-U_{sim}), (SVE1-U_{sim}), (SVE2-U_{sim}), (W1-U_{sim}) and (W2-U_{sim}).

Another rule has to be modified, namely (SC- U_{sim}). Previously, this rule would take hold in the case that the heads of the equation sides were not equal, now however it only applies in cases when the heads are not similar enough: $\mathcal{R}(\text{head}(s), \text{head}(t)) < \lambda$.

One more thing that has changed is the way decomposition works. Away from total and partial decomposition, we now have a rule, (HR- U_{sim}), which removes the term heads and leaves us with the argument sequences and a single decomposition rule (DEC- U_{sim}) which removes only the first argument pair each time it is applied.

The way this procedure works has not changed with the modifications. An initial configuration $\langle \Gamma; \varepsilon; 1 \rangle$ is set and a selection strategy determines which equation of the (\mathcal{R}, λ) -unification problem Γ is to be solved next. Depending on the form of this equation, a rule is applied and the step to the next configuration is made. This goes on until either Γ becomes the empty set or one of the failure rules applies. In the case that we end in $\langle \emptyset; \sigma; \alpha \rangle$, σ is a computed answer for Γ with unification degree α . Later we prove that it is a valid unifier of Γ , solving the problem with a degree α .

If a rule can be applied in multiple ways, or more than one rule is suitable for a chosen equation, branching can occur and is done concurrently. Branches can still end in either failure or success (or extend indefinitely).

The set of substitutions solving the (\mathcal{R}, λ) -unification problem Γ , produced by \mathcal{U}_{sim} , is denoted by $\text{Sol}_{\mathcal{U}_{\text{sim}}}(\Gamma)$.

Example 3.1. Let $\Gamma = \{f(\bar{x}, g(a), x, \bar{y}) \simeq_{\mathcal{R}, \lambda}^? g(f(x, \bar{x}), b, a, c)\}$, with a similarity relation $\mathcal{R} : \mathcal{R}(f, g) = 0.9, \mathcal{R}(a, b) = 0.7, \mathcal{R}(b, c) = 0.8, \mathcal{R}(a, c) = 0.7$ and a cut value $\lambda = 0.6$:

$$\begin{aligned}
 \langle \Gamma; \varepsilon; 1 \rangle &\Longrightarrow_{(\text{HR-}U_{\text{sim}})} \langle \{ \ulcorner \bar{x}, g(a), x, \bar{y} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner f(x, \bar{x}), b, a, c \urcorner \}; \varepsilon; 0.9 \rangle \\
 &\Longrightarrow_{(\text{SVOC-}U_{\text{sim}})} \perp \\
 \\
 \langle \Gamma; \varepsilon; 1 \rangle &\Longrightarrow_{(\text{P-}U_{\text{sim}})} \langle \{ f(g(a), x, \bar{y}) \simeq_{\mathcal{R}, \lambda}^? g(f(x), b, a, c) \}; \{ \bar{x} \mapsto \ulcorner \urcorner \}; 1 \rangle \\
 &\Longrightarrow_{(\text{HR-}U_{\text{sim}})} \langle \{ \ulcorner g(a), x, \bar{y} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner f(x), b, a, c \urcorner \}; \{ \bar{x} \mapsto \ulcorner \urcorner \}; 0.9 \rangle \\
 &\Longrightarrow_{(\text{DEC-}U_{\text{sim}})} \langle \{ g(a) \simeq_{\mathcal{R}, \lambda}^? f(x), \ulcorner x, \bar{y} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner b, a, c \urcorner \}; \{ \bar{x} \mapsto \ulcorner \urcorner \}; 0.9 \rangle \\
 &\Longrightarrow_{(\text{HR-}U_{\text{sim}})} \langle \{ a \simeq_{\mathcal{R}, \lambda}^? x, \ulcorner x, \bar{y} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner b, a, c \urcorner \}; \{ \bar{x} \mapsto \ulcorner \urcorner \}; 0.9 \rangle \\
 &\Longrightarrow_{(\text{O1-}U_{\text{sim}})} \langle \{ x \simeq_{\mathcal{R}, \lambda}^? a, \ulcorner x, \bar{y} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner b, a, c \urcorner \}; \{ \bar{x} \mapsto \ulcorner \urcorner \}; 0.9 \rangle \\
 &\Longrightarrow_{(\text{IVE-}U_{\text{sim}})} \langle \{ \ulcorner a, \bar{y} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner b, a, c \urcorner \}; \{ \bar{x} \mapsto \ulcorner \urcorner, x \mapsto a \}; 0.9 \rangle \\
 &\Longrightarrow_{(\text{DEC-}U_{\text{sim}})} \langle \{ a \simeq_{\mathcal{R}, \lambda}^? b, \ulcorner \bar{y} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner a, c \urcorner \}; \{ \bar{x} \mapsto \ulcorner \urcorner, x \mapsto a \}; 0.9 \rangle \\
 &\Longrightarrow_{(\text{HR-}U_{\text{sim}})} \langle \{ \ulcorner \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \urcorner, \ulcorner \bar{y} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner a, c \urcorner \}; \{ \bar{x} \mapsto \ulcorner \urcorner, x \mapsto a \}; 0.7 \rangle \\
 &\Longrightarrow_{(\text{T-}U_{\text{sim}})} \langle \{ \ulcorner \bar{y} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner a, c \urcorner \}; \{ \bar{x} \mapsto \ulcorner \urcorner, x \mapsto a \}; 0.7 \rangle \\
 &\Longrightarrow_{(\text{W1-}U_{\text{sim}})} \langle \{ \bar{y} \simeq_{\mathcal{R}, \lambda}^? c \}; \{ \bar{x} \mapsto \ulcorner \urcorner, x \mapsto a, \bar{y} \mapsto \ulcorner a, \bar{y} \urcorner \}; 0.7 \rangle
 \end{aligned}$$

$$\begin{aligned}
 & \Longrightarrow_{(\text{SVE2-U}_{\text{sim}})} \langle \{\}; \{\bar{x} \mapsto \ulcorner, x \mapsto a, \bar{y} \mapsto \lceil a, c \rceil\}; 0.7 \rangle \\
 \\
 \langle \Gamma; \varepsilon; 1 \rangle & \Longrightarrow_{(\text{P-U}_{\text{sim}})} \langle \{f(\bar{x}, g(a), x) \simeq_{\mathcal{R}, \lambda}^? g(f(x, \bar{x}), b, a, c)\}; \{\bar{y} \mapsto \ulcorner\}; 1 \rangle \\
 & \Longrightarrow_{(\text{HR-U}_{\text{sim}})} \langle \{\ulcorner \bar{x}, g(a), x \urcorner \simeq_{\mathcal{R}, \lambda}^? \lceil f(x, \bar{x}), b, a, c \rceil\}; \{\bar{y} \mapsto \ulcorner\}; 0.9 \rangle \\
 & \Longrightarrow_{(\text{SVOC-U}_{\text{sim}})} \perp \\
 \\
 \langle \Gamma; \varepsilon; 1 \rangle & \Longrightarrow_{(\text{P-U}_{\text{sim}})} \langle \{f(g(a), x) \simeq_{\mathcal{R}, \lambda}^? g(f(x), b, a, c)\}; \{\bar{x} \mapsto \ulcorner, \bar{y} \mapsto \ulcorner\}; 1 \rangle \\
 & \Longrightarrow_{(\text{HR-U}_{\text{sim}})} \langle \{\ulcorner g(a), x \urcorner \simeq_{\mathcal{R}, \lambda}^? \lceil f(x), b, a, c \rceil\}; \{\bar{x} \mapsto \ulcorner, \bar{y} \mapsto \ulcorner\}; 0.9 \rangle \\
 & \Longrightarrow_{(\text{DEC-U}_{\text{sim}})} \langle \{g(a) \simeq_{\mathcal{R}, \lambda}^? f(x), \lceil x \urcorner \simeq_{\mathcal{R}, \lambda}^? \lceil b, a, c \rceil\}; \{\bar{x} \mapsto \ulcorner, \bar{y} \mapsto \ulcorner\}; 0.9 \rangle \\
 & \Longrightarrow_{(\text{DEC-U}_{\text{sim}})} \langle \{g(a) \simeq_{\mathcal{R}, \lambda}^? f(x), x \simeq_{\mathcal{R}, \lambda}^? b, \ulcorner \simeq_{\mathcal{R}, \lambda}^? \lceil a, c \rceil\}; \\
 & \quad \{\bar{x} \mapsto \ulcorner, \bar{y} \mapsto \ulcorner\}; 0.9 \rangle \\
 & \Longrightarrow_{(\text{E1-U}_{\text{sim}})} \perp
 \end{aligned}$$

Similar to \mathcal{U} , the procedure \mathcal{U}_{sim} is not terminating. It has two reasons: first, the unification problem might have infinitely many solutions and the procedure keeps computing them. Second, due to the fact that the decision algorithm is not used, sometimes it might not be able to detect that a problem at some branch is unsolvable and keep expanding that branch.¹

Next, we want to prove soundness and completeness of \mathcal{U}_{sim} . For this, the proofs of \mathcal{U} are taken from [29] and modified to accommodate the changes we made replacing syntactic equality with similarity relations as well as the removal of sequence function symbols. In order to do that, we need some auxiliary results:

Lemma 3.1

Given a similarity relation \mathcal{R} , a λ -cut and an (\mathcal{R}, λ) -unification problem Γ , if $\mathcal{U}(\Gamma, \mathcal{R}, \lambda)_\alpha = \mathcal{U}(\Delta, \mathcal{R}, \lambda)_\alpha$, then $\mathcal{U}(\Gamma\sigma, \mathcal{R}, \lambda)_\alpha = \mathcal{U}(\Delta\sigma, \mathcal{R}, \lambda)_\alpha$ for any substitution σ .

Proof.

$$\begin{aligned}
 \theta \in \mathcal{U}(\Gamma\sigma, \mathcal{R}, \lambda)_\alpha & \text{ iff } \text{deg}_{\mathcal{R}}(\Gamma\sigma\theta) = \alpha \text{ iff } \sigma\theta \in \mathcal{U}(\Gamma, \mathcal{R}, \lambda)_\alpha \\
 & \text{ iff } \sigma\theta \in \mathcal{U}(\Delta, \mathcal{R}, \lambda)_\alpha \text{ iff } \text{deg}_{\mathcal{R}}(\Delta\sigma\theta) = \lambda \text{ iff } \theta \in \mathcal{U}(\Delta\sigma, \mathcal{R}, \lambda).
 \end{aligned}$$

□

¹This later case can be excluded if we had a decision algorithm for similarity-based unranked unification problems, which, in itself, would depend on decidability of similarity-based word unification, but we are not aware of any work in this direction.

Lemma 3.2

Given a similarity relation \mathcal{R} , a λ -cut and an (\mathcal{R}, λ) -unification problem Γ , if $\langle \Gamma; \sigma; \alpha \rangle \Longrightarrow \langle \Delta; \sigma\theta; \alpha \wedge \beta \rangle$ then $\mathcal{U}(\Delta, \mathcal{R}, \lambda) = \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)$.

Additionally if $\varphi \in \mathcal{U}(\Delta, \mathcal{R}, \lambda)_\gamma$ then $\varphi \in \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)_{\gamma \wedge \beta}$.

Proof. It is easy to see that this holds for the rules which neither change the substitution σ and therefore only replacing the unification problem with an equivalent one, nor change the unification degree α . So here we will show some of the rules which are not that trivial:

(IVE- \mathcal{U}_{sim}) We have $\langle \{x \simeq_{\mathcal{R}, \lambda}^? t\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow_S \langle \Delta; \sigma\theta; \alpha \rangle$ with $\Delta = \Gamma'\theta$, $\theta = \{x \mapsto t\}$ and $\beta = 1$.

Since $x \notin \mathcal{V}_I(t)$ we know that $x\theta \simeq_{\mathcal{R}, \lambda} t\theta$ and therefore also $\Gamma\theta = \{x \simeq_{\mathcal{R}, \lambda}^? t\}\theta \cup \Gamma'\theta = \{x\theta \simeq_{\mathcal{R}, \lambda}^? t\theta\} \cup \Gamma'\theta$.

“ \subseteq ” Let $\varphi \in \mathcal{U}(\Delta, \mathcal{R}, \lambda)_\gamma$ which also means $\deg_{\mathcal{R}}(\Delta\varphi) = \gamma \geq \lambda$. In order to show that $\varphi \in \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)_{\gamma \wedge \beta} \subseteq \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)$ we can show that:

$$\begin{aligned}
 \deg_{\mathcal{R}}(\Gamma\theta\varphi) &= \deg(\{x\theta \simeq_{\mathcal{R}, \lambda}^? t\theta\}\varphi \cup \Gamma'\theta\varphi) \\
 &= \deg_{\mathcal{R}}(\{x\theta \simeq_{\mathcal{R}, \lambda}^? t\theta\}\varphi) \wedge \deg_{\mathcal{R}}(\Gamma'\theta\varphi) \\
 &= \deg_{\mathcal{R}}(\{t \simeq_{\mathcal{R}, \lambda}^? t\}\varphi) \wedge \deg_{\mathcal{R}}(\Delta\varphi) \\
 &= 1 \wedge \gamma = \beta \wedge \gamma \geq \lambda
 \end{aligned}$$

“ \supseteq ” Let $\varphi \in \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)$ which also means $\deg_{\mathcal{R}}(\Gamma\theta\varphi) \geq \lambda$. In order to show that $\varphi \in \mathcal{U}(\Delta, \mathcal{R}, \lambda) \iff \deg_{\mathcal{R}}(\Delta\varphi) \geq \lambda$, we can simply rewrite $\deg_{\mathcal{R}}(\Gamma\theta\varphi) \geq \lambda$ as we have done above:

$$\begin{aligned}
 \deg_{\mathcal{R}}(\Gamma\theta\varphi) &= \deg_{\mathcal{R}}(\{t \simeq_{\mathcal{R}, \lambda}^? t\}\varphi) \wedge \deg_{\mathcal{R}}(\Delta\varphi) \\
 &= 1 \wedge \deg_{\mathcal{R}}(\Delta\varphi) \\
 &= \deg_{\mathcal{R}}(\Delta\varphi) \geq \lambda.
 \end{aligned}$$

(HR- \mathcal{U}_{sim}) We have $\langle \{f(s_1, \dots, s_n) \simeq_{\mathcal{R}, \lambda}^? g(t_1, \dots, t_m)\} \cup \Gamma'; \sigma; \alpha \rangle \Longrightarrow_{\text{HR}} \langle \Delta; \sigma\theta; \alpha \wedge \beta \rangle$ with $\Delta = \{\ulcorner s_1, \dots, s_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner t_1, \dots, t_m \urcorner\} \cup \Gamma'$, $\theta = \varepsilon$, $\beta = \mathcal{R}(f, g) \geq \lambda$ and $f(s_1, \dots, s_n) \neq g(t_1, \dots, t_m)$, and $\mathcal{R}(f, g) = \beta \geq \lambda$.

“ \subseteq ” Let $\varphi \in \mathcal{U}(\Delta, \mathcal{R}, \lambda)_\gamma$ which also means $\deg_{\mathcal{R}}(\Delta\varphi) = \gamma$. In order to show that $\varphi \in \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)_{\gamma \wedge \beta} \subseteq \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)$ we can show that:

$$\begin{aligned}
 \deg_{\mathcal{R}}(\Gamma\theta\varphi) &= \deg_{\mathcal{R}}(\{f(s_1, \dots, s_n) \simeq_{\mathcal{R}, \lambda}^? g(t_1, \dots, t_m)\}\varphi \cup \Gamma'\varphi) \\
 &= \deg_{\mathcal{R}}(\{f(s_1, \dots, s_n) \simeq_{\mathcal{R}, \lambda}^? g(t_1, \dots, t_m)\}\varphi) \wedge \deg_{\mathcal{R}}(\Gamma'\varphi)
 \end{aligned}$$

$$\begin{aligned}
 &= \mathcal{R}(f, g) \wedge \deg_{\mathcal{R}}(\{\ulcorner s_1, \dots, s_n \urcorner \simeq_{\mathcal{R}, \lambda}^{\ulcorner} t_1, \dots, t_m \urcorner\} \varphi) \wedge \deg_{\mathcal{R}}(\Gamma' \varphi) \\
 &= \mathcal{R}(f, g) \wedge \deg_{\mathcal{R}}(\{\ulcorner s_1, \dots, s_n \urcorner \simeq_{\mathcal{R}, \lambda}^{\ulcorner} t_1, \dots, t_m \urcorner\} \varphi \cup \Gamma' \varphi) \\
 &= \mathcal{R}(f, g) \wedge \deg_{\mathcal{R}}(\Delta \varphi) \\
 &= \beta \wedge \gamma \geq \lambda.
 \end{aligned}$$

“ \supseteq ” Let $\varphi \in \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda) = \mathcal{U}(\Gamma, \mathcal{R}, \lambda)$. In order to show that $\varphi \in \mathcal{U}(\Delta, \mathcal{R}, \lambda) \implies \deg_{\mathcal{R}}(\Delta \varphi) \geq \lambda$, we can again rewrite $\deg_{\mathcal{R}}(\Gamma \varphi) = \gamma \geq \lambda$ as

$$\begin{aligned}
 \deg_{\mathcal{R}}(\Gamma \varphi) &= \deg_{\mathcal{R}}(\{f(s_1, \dots, s_n) \simeq_{\mathcal{R}, \lambda}^? g(t_1, \dots, t_m)\} \varphi \cup \Gamma' \varphi) \\
 &= \mathcal{R}(f, g) \wedge \deg_{\mathcal{R}}(\{\ulcorner s_1, \dots, s_n \urcorner \simeq_{\mathcal{R}, \lambda}^{\ulcorner} t_1, \dots, t_m \urcorner\} \varphi \cup \Gamma' \varphi) \\
 &= \mathcal{R}(f, g) \wedge \deg_{\mathcal{R}}(\Delta \varphi).
 \end{aligned}$$

So knowing that $\deg_{\mathcal{R}}(\Gamma \varphi) \geq \gamma$ we can differentiate two cases:

Case 1: Lets assume $\mathcal{R}(f, g) \leq \deg_{\mathcal{R}}(\Delta \varphi)$. It follows immediately that $\deg_{\mathcal{R}}(\Delta \varphi) \geq \mathcal{R}(f, g) \geq \lambda$.

Case 2: Lets assume $\mathcal{R}(f, g) \geq \deg_{\mathcal{R}}(\Delta \varphi)$. This means $\mathcal{R}(f, g) \wedge \deg_{\mathcal{R}}(\Delta \varphi) = \deg_{\mathcal{R}}(\Delta \varphi)$ and therefore $\deg_{\mathcal{R}}(\Gamma \varphi) = \deg_{\mathcal{R}}(\Delta \varphi) = \gamma \geq \lambda$.

Hence $\deg_{\mathcal{R}}(\Delta, \varphi) \geq \lambda \iff \varphi \in \mathcal{U}(\Delta, \mathcal{R}, \lambda)$.

The proofs for the other rules with changing substitutions follow analogously to [29] with similar modification as just presented with the proof of the (IVE- U_{sim}) rule. \square

Lemma 3.3

Given a similarity relation \mathcal{R} , a λ -cut and an (\mathcal{R}, λ) -unification problem Γ , if $\langle \Gamma; \sigma; \alpha \rangle \implies^+ \langle \Delta; \sigma\theta; \alpha \wedge \beta \rangle$, then $\mathcal{U}(\Delta, \mathcal{R}, \lambda) = \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)$.

Additionally if $\varphi \in \mathcal{U}(\Delta, \mathcal{R}, \lambda)_{\gamma}$ then $\varphi \in \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)_{\gamma \wedge \beta}$.

Proof. This will be proven by induction on the number of steps, using Lemma 3.2. If the step length of this derivation was only 1 then this would already be proven by Lemma 3.2. Lets assume as induction hypothesis that our lemma holds for derivations of length n .

Now let

$$\begin{aligned}
 \langle \Gamma; \sigma; \alpha \rangle &\implies \langle \Delta_1; \sigma\theta_1; \alpha \wedge \beta_1 \rangle \implies \dots \implies \langle \Delta_n; \sigma\theta_1 \dots \theta_n; \alpha \wedge \beta_1 \wedge \dots \wedge \beta_n \rangle \\
 &\implies \langle \Delta_{n+1}; \sigma\theta_1 \dots \theta_{n+1}; \alpha \wedge \beta_1 \wedge \dots \wedge \beta_{n+1} \rangle
 \end{aligned}$$

be our derivation where $\theta = \theta_1 \dots \theta_{n+1}$ and $\beta = \beta_1 \wedge \dots \wedge \beta_{n+1}$.

The induction hypothesis tells us that $\mathcal{U}(\Delta_n, \mathcal{R}, \lambda) = \mathcal{U}(\Gamma\theta_1 \cdots \theta_n, \mathcal{R}, \lambda)$ and by Lemma 3.2 we know that $\mathcal{U}(\Delta_{n+1}, \mathcal{R}, \lambda) = \mathcal{U}(\Delta_n\theta_{n+1}, \mathcal{R}, \lambda)$.

We want to show that $\mathcal{U}(\Delta_{n+1}, \mathcal{R}, \lambda) = \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)$ which can be done using Lemma 3.1 and our induction hypothesis:

$$\begin{aligned}
 \mathcal{U}(\Delta_{n+1}, \mathcal{R}, \lambda) &= \mathcal{U}(\Delta_n\theta_{n+1}, \mathcal{R}, \lambda) \\
 &= \mathcal{U}(\Gamma\theta_1 \cdots \theta_n\theta_{n+1}, \mathcal{R}, \lambda) \\
 &= \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)
 \end{aligned}$$

We additionally want to show that if $\varphi \in \mathcal{U}(\Delta_{n+1}, \mathcal{R}, \lambda)_\gamma$, then $\varphi \in \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)_{\gamma\wedge\beta}$. By induction hypothesis, if $\varphi \in \mathcal{U}(\Delta_n, \mathcal{R}, \lambda)_\gamma$, then $\varphi \in \mathcal{U}(\Gamma\theta_1 \cdots \theta_n, \mathcal{R}, \lambda)_{\gamma\wedge\beta_1\wedge\cdots\wedge\beta_n}$. From Lemma 3.2 we know that if

$$\begin{aligned}
 \varphi \in \mathcal{U}(\Delta_{n+1}, \mathcal{R}, \lambda)_\gamma &\stackrel{\text{L3.2}}{\implies} \varphi \in \mathcal{U}(\Delta_n\theta_{n+1}, \mathcal{R}, \lambda)_{\gamma\wedge\beta_{n+1}} \\
 &\text{iff } \varphi\theta_{n+1} \in \mathcal{U}(\Delta_n, \mathcal{R}, \lambda)_{\gamma\wedge\beta_{n+1}} \\
 &\stackrel{\text{IH}}{\implies} \varphi\theta_{n+1} \in \mathcal{U}(\Gamma\theta_1 \cdots \theta_n, \mathcal{R}, \lambda)_{\gamma\wedge\beta_{n+1}\wedge\beta_1\wedge\cdots\wedge\beta_n} \\
 &\text{iff } \varphi \in \mathcal{U}(\Gamma\theta_1 \cdots \theta_n\theta_{n+1}, \mathcal{R}, \lambda)_{\gamma\wedge\beta} \\
 &\text{iff } \varphi \in \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)_{\gamma\wedge\beta}
 \end{aligned}$$

□

Lemma 3.4

Given a similarity relation \mathcal{R} , a λ -cut and an (\mathcal{R}, λ) -unification problem Γ , if $\langle \Gamma; \varepsilon; 1 \rangle \implies^+ \langle \emptyset; \theta; \alpha \rangle$ then $\theta \in \mathcal{U}(\Gamma, \mathcal{R}, \lambda)_\alpha$ with similarity degree α .

Proof. By Lemma 3.3 we know that $\mathcal{U}(\emptyset, \mathcal{R}, \lambda) = \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)$. Since $\varepsilon \in \mathcal{U}(\emptyset, \mathcal{R}, \lambda)_1$ it follows that $\varepsilon \in \mathcal{U}(\Gamma\theta, \mathcal{R}, \lambda)_{1\wedge\alpha}$ iff $\theta \in \mathcal{U}(\Gamma, \mathcal{R}, \lambda)_\alpha$. □

Theorem 3.5 (Soundness of $\mathfrak{U}_{\text{sim}}$)

Let \mathcal{R} be a similarity relation, λ a cut value for \mathcal{R} , and Γ an (\mathcal{R}, λ) -unification problem. Then $\text{Sol}_{\mathfrak{U}_{\text{sim}}}(\Gamma, \mathcal{R}, \lambda) \subseteq \mathcal{U}(\Gamma, \mathcal{R}, \lambda)$.

Proof. This follows from the definition of the procedure $\mathfrak{U}_{\text{sim}}$ and Lemma 3.4. □

The following definitions, taken from [29], will be used for proving the completeness of $\mathfrak{U}_{\text{sim}}$.

Definition 3.5

The length of the image of a set of variables \mathcal{X} with respect to a substitution σ , denoted as $\text{Len}(\mathcal{X}, \sigma)$, is defined as $\sum_{v \in \mathcal{X}} \text{len}(v\sigma)$, where $\text{len}(v\sigma)$ is 1 if $v\sigma$ is a single term, and is n if $v\sigma = \ulcorner t_1, \dots, t_n \urcorner$ for some terms t_1, \dots, t_n , $n \geq 0$.

We denote by $\text{Dif}(\mathcal{X}, \theta, \sigma)$ the length difference $\text{Len}(\mathcal{X}, \theta) - \text{Len}(\mathcal{X}, \sigma)$. Obviously, $\text{Dif}(\mathcal{X}, \theta, \sigma) = \text{Dif}(\mathcal{V}_S(\mathcal{X}), \theta, \sigma)$.

Lemma 3.6

For all σ, θ and \mathcal{X} , if $\text{ran}(\sigma) \subseteq \mathcal{X}$ and $\sigma \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$, then $\text{Len}(\mathcal{X}, \sigma) \leq \text{Len}(\mathcal{X}, \theta)$.

Proof. This follows from the properties of length of an image and (\mathcal{R}, λ) -more-general. \square

Now we continue with proving completeness of the procedure $\mathfrak{U}_{\text{sim}}$. We want to show that for any solution $\theta \in \mathcal{U}(\Gamma, \mathcal{R}, \lambda)_{\beta}$ with similarity degree $\beta \geq \lambda$, there exists a derivation from the initial configuration $\langle \Gamma; \varepsilon; 1 \rangle$ which terminates with success and the substitution in the final configuration of the derivation is (\mathcal{R}, λ) -strongly more general than θ and has a similarity degree α with $\alpha \geq \beta \geq \lambda$.

Lemma 3.7

Let Γ be an (\mathcal{R}, λ) -unification problem, $\mathcal{X} = \mathcal{V}(\Gamma)$ the set of variables occurring in Γ , $\theta \in \mathcal{U}(\Gamma, \mathcal{R}, \lambda)_{\beta}$ a solution of the problem with similarity degree $\beta \geq \lambda$ and S a selection strategy. If θ is non-erasing on \mathcal{X} , then there exists a derivation of the form $\langle \Gamma_1; \sigma_1; \alpha_1 \rangle \Longrightarrow_{BT}^+ \langle \emptyset; \sigma_k; \alpha_k \rangle$ where $\Gamma_1 = \Gamma$, $\sigma_1 = \varepsilon$ and $\alpha_1 = 1$, such that $\sigma_k \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$ and $\alpha_k \geq \beta \geq \lambda$.

Proof. The proof will be split into first constructing a fitting derivation and secondly showing the successful termination of such a construction.

Step 1: Constructing a derivation of the form

$$\langle \Gamma_1; \sigma_1; \alpha_1 \rangle \Longrightarrow_{BT} \langle \Gamma_2; \sigma_2; \alpha_2 \rangle \Longrightarrow_{BT} \dots,$$

with $\sigma_i \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$ and $\alpha_i \geq \beta$.

In order to construct this derivation recursively, we start with $\Gamma_1 = \Gamma$, $\sigma_1 = \varepsilon$ and $\alpha_1 = 1$. It is easy to see that $\sigma_1 = \varepsilon \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$ and $\alpha_1 = 1 \geq \beta \geq \lambda$.

Assuming that we have already constructed the derivation up to $\langle \Gamma_n; \sigma_n; \alpha_n \rangle$ with $n \geq 1$ and $\Gamma_n \neq \emptyset$, we have to create a configuration $\langle \Gamma_{n+1}; \sigma_{n+1}; \alpha_{n+1} \rangle$ such that $\langle \Gamma_n; \sigma_n; \alpha_n \rangle \Longrightarrow_{BT} \langle \Gamma_{n+1}; \sigma_{n+1}; \alpha_{n+1} \rangle$, $\sigma_{n+1} \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$ and $\alpha_{n+1} \geq \beta$.

Knowing that $\langle \Gamma_n; \sigma_n; \alpha_n \rangle$ is already part of our derivation, we can use the fact that $\sigma_n \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$, which means that there exists a φ , non-erasing on \mathcal{X} , such that $\sigma_n \varphi \simeq_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$. From this we get

$\sigma_n \varphi \in \mathcal{U}(\Gamma, \mathcal{R}, \lambda)$ and therefore $\varphi \in \mathcal{U}(\Gamma \sigma_n, \mathcal{R}, \lambda)$. By Lemma 3.3 and $\Gamma_1 = \Gamma$ we know that $\varphi \in \mathcal{U}(\Gamma_n, \mathcal{R}, \lambda)$.

Next, we are going to take a closer look at the pair $s \simeq_{\mathcal{R}, \lambda}^? t$ chosen by the selection algorithm from $\Gamma_n = \{s \simeq_{\mathcal{R}, \lambda}^? t\} \cup \Gamma'_n$. The following cases emerge, depending on the chosen pair:

1. $s \simeq_{\mathcal{R}, \lambda}^? t$ is a pair of identical terms. In this case, the next step in the derivation will be $\langle \Gamma_n; \sigma_n; \alpha_n \rangle \Longrightarrow_{\text{T-U}_{\text{sim}}} \langle \Gamma'_n; \sigma_n; \alpha_n \rangle$. Consequently, $\sigma_{n+1} = \sigma_n \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$ and $\alpha_{n+1} = \alpha_n \geq \beta$.
2. $s \simeq_{\mathcal{R}, \lambda}^? t$ is a pair of distinct individual variables, where $s = x$ and $t = y$. For $\psi = \{x \mapsto y\}$ we want to show that $\sigma_{n+1} = \sigma_n \psi \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$. Therefore we have to show that there exists a substitution $\bar{\varphi}$ such that $\sigma_n \psi \bar{\varphi} \approx_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$, meaning that for all variables $z \in \mathcal{X}$, $\mathcal{R}(\sigma_n \psi \bar{\varphi} z, \theta z) \geq \lambda$ holds. Selecting our φ for $\bar{\varphi}$, we already know that $x\varphi \simeq_{\mathcal{R}, \lambda} y\varphi$ as well as $\forall z \in \mathcal{X} : \mathcal{R}(\sigma_n \varphi z, \theta z) \geq \lambda$, since $\sigma_n \varphi \approx_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$. Because $x\psi \approx_{\mathcal{R}, \lambda} y\psi$, it is easy to see that $\psi\varphi \simeq_{\mathcal{R}, \lambda}^{\mathcal{X}} \varphi$ and therefore $\sigma_n \psi \varphi \simeq_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$. Hence $\sigma_{n+1} = \sigma_n \psi \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$.

Due to the similarity degree not changing in this step we get $\alpha_{n+1} = \alpha_n \geq \beta$. We can also set $\Gamma_{n+1} = \Gamma_n \psi$ and extend the derivation with the step $\langle \Gamma_n; \sigma_n; \alpha_n \rangle \Longrightarrow_{\text{IVE-U}_{\text{sim}}} \langle \Gamma_{n+1}; \sigma_{n+1}; \alpha_{n+1} \rangle$.

3. $s \simeq_{\mathcal{R}, \lambda}^? t$ is a pair of an individual variable and a non-variable term. In the case that $s = x$ and $x \notin \mathcal{V}_I(t)$, we can reuse the same step ($\text{IVE-U}_{\text{sim}}$) as in case 1. However, if $t = x$ with s being the non-variable term and $x \notin \mathcal{V}_I(s)$, we continue with the derivation step $\langle \Gamma_n; \sigma_n; \alpha_n \rangle \Longrightarrow_{\text{O1-U}_{\text{sim}}} \langle \Gamma_{n+1}; \sigma_{n+1}; \alpha_{n+1} \rangle$, where $\Gamma_{n+1} = \{x \simeq_{\mathcal{R}, \lambda}^? s\} \cup \Gamma'_n$, $\sigma_{n+1} = \sigma_n$ and $\alpha_{n+1} = \alpha_n$.
4. $s \simeq_{\mathcal{R}, \lambda}^? t$ is a pair of distinct non-variable terms with $s = f(s_1, \dots, s_n)$ and $t = g(t_1, \dots, t_m)$, where $\mathcal{R}(f, g) \geq \lambda$. Here the rule (HR-U_{sim}) will be used, resulting in $\Gamma_{n+1} = \{\ulcorner s_1, \dots, s_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner t_1, \dots, t_m \urcorner\} \cup \Gamma'_n$, $\sigma_{n+1} = \sigma_n$ and $\alpha_{n+1} = \alpha \wedge \mathcal{R}(f, g)$. It is easy to argue that $\mathcal{R}(f, g) \geq \beta$ since it would have been considered in θ with its unification degree of β . Therefore $\alpha_{n+1} \geq \beta$.
5. $s \simeq_{\mathcal{R}, \lambda}^? t$ is a pair of distinct sequences of terms with $s = \ulcorner s_1, \dots, s_n \urcorner$ and $t = \ulcorner t_1, \dots, t_m \urcorner$. We can differentiate between the following cases:

- if $s_1, t_1 \notin \mathcal{V}_S(\Gamma)$, the rule ($\text{DEC-U}_{\text{sim}}$) is used. We get $\alpha_{n+1} = \alpha_n$ for which we already know that $\alpha_n \geq \beta$ holds and also $\sigma_{n+1} = \sigma_n \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$;
- if $s_1, t_1 \in \mathcal{V}_S(\Gamma)$ and $s_1 = t_1$, the derivation is extended by $\langle \Gamma_n; \sigma_n; \alpha_n \rangle \Longrightarrow_{\text{SV1-U}_{\text{sim}}} \langle \Gamma_{n+1}; \sigma_{n+1}; \alpha_{n+1} \rangle$, where $\Gamma_{n+1} = \{\ulcorner s_2, \dots, s_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner t_2, \dots, t_m \urcorner\} \cup \Gamma'_n$, $\sigma_{n+1} = \sigma_n \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$ and $\alpha_{n+1} = \alpha_n \geq \beta$;
- if $s_1 \notin \mathcal{V}_S(\Gamma)$ and $t_1 \in \mathcal{V}_S(\Gamma)$, the step $\langle \Gamma_n; \sigma_n; \alpha_n \rangle \Longrightarrow_{\text{O2-U}_{\text{sim}}} \langle \Gamma'_n; \sigma_n; \alpha_n \rangle$ will be performed;

- if $s_1 \in \mathcal{V}_S(\Gamma)$, $s_1 \notin \mathcal{V}_S(t_1)$ and $m > 0$, further case distinctions can be made depending on the properties of t_1 . The proofs for these cases do not change much from the original paper except for the fact that they are now applied to sequences of terms instead of individual terms with said sequences as argument lists and the addition of α , which remains unchanged in these steps. Therefore we will refer to the original proof in [29].

The projection rule is not used, due to the fact that θ is non-erasing on \mathcal{X} , and therefore only basic transformation steps make up the derivation.

Step 2: In order to show that the derivation we just built actually terminates, we construct a complexity measure. This complexity measure is a 6-tuple $\langle m_1, m_2, m_3, m_4, m_5, m_6 \rangle$, where m_4 is a multiset of natural numbers, and the other components are integers. The tuples are ordered lexicographically, using the standard ordering on integers and the Dershowitz-Manna ordering on multisets. Given a configuration $\langle \Delta; \sigma; \alpha \rangle$, a set of variables \mathcal{Y} , and a substitution δ , the components of the measure m_1, \dots, m_6 are defined as follows:

$$\begin{aligned}
 m_1 &= \text{the number of distinct variables in } \Delta, \\
 m_2 &= \text{Dif}(\mathcal{Y}, \delta, \sigma), \\
 m_3 &= \text{the number of symbols in } \Delta, \\
 m_4 &= \{\{\text{size}(\ulcorner t_1, \dots, t_m \urcorner) \mid \ulcorner s_1, \dots, s_n \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner t_1, \dots, t_m \urcorner \in \Gamma\}\}, \\
 m_5 &= \text{the number of equations in } \Delta \text{ of the form } s \simeq_{\mathcal{R}, \lambda}^? x, \text{ where } s \notin \mathcal{V}_I, \\
 m_6 &= \text{the number of equations in } \Delta \text{ that have the form} \\
 &\quad \ulcorner s, s_1, \dots, s_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \bar{x}, t_1, \dots, t_m \urcorner, \text{ where } s \notin \mathcal{V}_S.
 \end{aligned}$$

For the ordering on complexity measures to be well-founded, all of their number components have to be naturals. This is clear for all but m_2 : By definition of Dif, m_2 can also be negative, depending on the substitutions given. However, we know that since $\sigma_1 = \varepsilon$ and because no new variables are introduced, for each σ_i it holds that $\text{ran}(\sigma_i) \subseteq \mathcal{X}$. Therefore we know that $\text{Dif}(\mathcal{X}, \theta, \sigma_i) > 0$, since $\sigma_i \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$. For m_4 , the multiset ordering is well-founded. Hence, if we take \mathcal{X} in the role of \mathcal{Y} and θ in the role of δ , we get a well-founded ordering on complexity measures.

The table below shows that each step strictly reduces the complexity measure:

	m_1	m_2	m_3	m_4	m_5	m_6
T- U_{sim} :	\geq	\geq	$>$			
O1- U_{sim} :	\geq	\geq	\geq	\geq	$>$	
O2- U_{sim} :	\geq	\geq	\geq	\geq	\geq	$>$
IVE- U_{sim} :	$>$					
HR- U_{sim} :	\geq	\geq	$>$			
DEC- U_{sim} :	\geq	\geq	\geq	$>$		
SVE1- U_{sim} :	\geq	\geq	$>$			
SVE2- U_{sim} :	$>$					
W1- U_{sim} :	\geq	$>$				
W2- U_{sim} :	\geq	$>$				

This means that our constructed derivation terminates successfully.

Let $\langle \Gamma_k; \sigma_k; \alpha_k \rangle$ be the final configuration in our construct. We know that $\Gamma_k = \emptyset$, because otherwise another step would be possible, additionally $\sigma_k \preceq_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$ and $\alpha_k \geq \beta$ holds. This finished the proof of the lemma. \square

Lemma 3.8

Let Γ be an (\mathcal{R}, λ) -unification problem, $\mathcal{X} = \mathcal{V}(\Gamma)$ the set of variables occurring in Γ , $\theta \in \mathcal{U}(\Gamma, \mathcal{R}, \lambda)_\beta$ a solution of the problem with similarity degree $\beta \geq \lambda$ and S a selection strategy.

If θ is erasing on \mathcal{X} , then there exists a derivation of the form

$$\langle \Gamma_0; \sigma_0; \alpha_0 \rangle \Longrightarrow_P \langle \Gamma_1; \sigma_1; \alpha_1 \rangle \Longrightarrow_{BT}^+ \langle \emptyset; \sigma_n; \alpha_n \rangle,$$

where $\Gamma_0 = \Gamma$, $\sigma_0 = \varepsilon$ and $\alpha_0 = 1$, such that $\sigma_n \preceq_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$ and $\alpha_n \geq \beta$.

Proof. Assume $\theta = \{\bar{x}_1 \mapsto \ulcorner, \dots, \bar{x}_k \mapsto \ulcorner, \dots\}$ with $\bar{x}_1, \dots, \bar{x}_k \in \mathcal{X}$, $k > 0$, being the variables mapped to the empty sequence. We perform a step with the projection rule $\langle \Gamma_0; \sigma_0; \alpha_0 \rangle \Longrightarrow_P \langle \Gamma_1; \sigma_1; \alpha_1 \rangle$ with $\sigma_1 = \{\bar{x}_1 \mapsto \ulcorner, \dots, \bar{x}_k \mapsto \ulcorner\}$, $\Gamma_1 = \Gamma_0 \sigma_1$ and $\alpha_1 = \alpha_0$. By defining a substitution φ as $\varphi = \theta|_{\text{dom}(\theta) \setminus \text{dom}(\sigma_1)}$ and due to the fact that σ_1 is idempotent, i.e., $\sigma_1 \sigma_1 = \sigma_1$, we can see that $\theta = \sigma_1 \varphi = \sigma_1 \sigma_1 \varphi \in \mathcal{U}(\Gamma_0, \mathcal{R}, \lambda)_\beta$ iff $\sigma_1 \varphi \in \mathcal{U}(\Gamma_0 \sigma_1, \mathcal{R}, \lambda)$ iff $\theta \in \mathcal{U}(\Gamma_1, \mathcal{R}, \lambda)$. Since $\Gamma_1 = \Gamma_0 \sigma_1$ we know that θ is now non-erasing on $\mathcal{V}(\Gamma_1)$.

From lemma 3.7 we know that there exists a derivation of the form $\langle \Delta_1; \delta_1; \gamma_1 \rangle \Longrightarrow_{BT}^+ \langle \emptyset; \delta_n; \gamma_n \rangle$ with $\Delta_1 = \Gamma_1$, $\delta_1 = \varepsilon$ and $\gamma_1 = 1$ such that $\gamma_n \geq \beta$ and $\delta_n \preceq_{\mathcal{R}, \lambda}^{\mathcal{X}_1} \theta$ with $\mathcal{X}_1 = \mathcal{V}(\Gamma_1)$. Therefore, there exists a substitution ψ such that $\delta_n \psi \simeq_{\mathcal{R}, \lambda}^{\mathcal{X}_1} \theta$ and ψ is non-erasing on \mathcal{X}_1 . From $\text{vran}(\delta_n) \subseteq \mathcal{X}_1$, we know that $\sigma_1 \delta_n = \sigma_1 \cup \delta_n$ and from $\text{dom}(\delta_1) \subseteq \mathcal{X}_1$ due the fact that no rule introduces new variables, we can assume that ψ is also non-erasing on \mathcal{X} . Hence, $\sigma_1 \delta_n \psi \simeq_{\mathcal{R}, \lambda}^{\mathcal{X}_1} \theta$. If we have $\langle \Delta_i; \delta_i; \gamma_i \rangle \Longrightarrow_R \langle \Delta_{i+1}; \delta_{i+1}; \gamma_{i+1} \rangle$ for $i \geq 1$ with R being a basic

transformation rule, then we also have $\langle \Delta_i; \sigma_1 \delta_i; \gamma_i \rangle \Longrightarrow_R \langle \Delta_{i+1}; \sigma_1 \delta_{i+1}; \gamma_{i+1} \rangle$ with the same rule R . By simply choosing $\Gamma_i = \Delta_i$, $\sigma_i = \sigma_1 \delta_i$ and $\alpha_i = \gamma_i$ for $1 < i \leq n$ we can construct the derivation $\langle \Gamma; \epsilon; 1 \rangle \Longrightarrow_P \langle \Gamma_1; \sigma_1; \alpha_1 \rangle \Longrightarrow_{BT} \cdots \Longrightarrow_{BT} \langle \emptyset; \sigma_n; \alpha_n \rangle$ which satisfies $\sigma_n \lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}} \theta$ and $\alpha_n \geq \beta$. \square

Theorem 3.9 (Completeness of \mathcal{U}_{sim})

Let \mathcal{R} be a similarity relation, λ a cut value for \mathcal{R} , and Γ an (\mathcal{R}, λ) -unification problem. Then $Sol_{\mathcal{U}_{sim}}(\Gamma)$ is a complete set of (\mathcal{R}, λ) -unifiers of Γ for $var(\Gamma)$.

Proof. This follows from Theorem 3.5 and Lemmata 3.7 and 3.8. \square

Note that $Sol_{\mathcal{U}_{sim}}(\Gamma)$ in general, is not minimal, because it may contain $\lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}}$ -comparable substitutions, where $\mathcal{X} = var(\Gamma)$. However, this set does not contain $\lesssim_{\mathcal{R}, \lambda}^{\mathcal{X}}$ -comparable substitutions. This result is analogous to the one in [29], and therefore we do not go into further details here.

3.2 Fuzzy Unranked Unification with Proximity Relations

This section uses the algorithm for class-based unification with proximity relations introduced by Temur Kutsia and Cleo Pau in [35] to modify the unranked unification to work with proximity. To do this we need some modifications to X-terms and some further notions taken from [35] and [43]:

A *name*, denoted by the bold-faced letters $\mathbf{n}, \mathbf{m}, \mathbf{k}$, is a symbol, designed to be a placeholder for a constant or an unranked function symbol. If it stands for a constant, its arity is defined as 0. Otherwise it is unranked. They themselves belong neither to the set of variables of the language nor to the set of function symbols: They form a separate set, which we denote by \mathcal{N} and assume that it is countable. A *neighborhood* is either a finite set of constants, or a finite set of unranked function symbols, or a name. It will be denoted by \mathbf{f}, \mathbf{g} , or \mathbf{h} . The symbol Nb stands for the set of all neighborhoods.

Within this section, the definition of extended terms is slightly different from the one introduced in Chapter 2. Whereas there the set of X-terms was defined over \mathcal{F} and \mathcal{V} as $\mathcal{T}_X(\mathcal{F}, \mathcal{V})$, here it is defined over \mathcal{F} , \mathcal{N} , and \mathcal{V} and is denoted as $\mathcal{T}_X(\mathcal{F}, \mathcal{N}, \mathcal{V})$, with $\mathbf{t} \in \mathcal{T}_X(\mathcal{F}, \mathcal{N}, \mathcal{V})$ being defined as $\mathbf{t} := x \mid \bar{x} \mid \mathbf{n}(\mathbf{t}_1, \dots, \mathbf{t}_n) \mid \mathbf{f}(\mathbf{t}_1, \dots, \mathbf{t}_n)$. We will continue to use $\mathbf{s}, \mathbf{t}, \mathbf{r}$ for X-terms over neighborhoods. Other concepts we have defined on X-terms in Chapter 2 can be applied on neighborhood X-terms equivalently.

In the case when an X-term has only singleton neighborhoods, meaning the neighborhood is either a name or a single function symbol, we will refer to such an X-term as *singleton X-term*

or *SX-term*. Even though not entirely correct, we will use the similarity of this definition with regular terms to present terms as SX-terms, and therefore function symbols as singleton neighborhoods.

It is possible to convert an X-term to a set of SX-terms:

$$\begin{aligned}
 \text{singl}(x) &:= \{x\}, \\
 \text{singl}(\bar{x}) &:= \{\bar{x}\}, \\
 \text{singl}(\mathbf{n}(\mathbf{t}_1, \dots, \mathbf{t}_n)) &:= \{\mathbf{n}(\mathbf{s}_1, \dots, \mathbf{s}_n) \mid \mathbf{s}_i \in \text{singl}(\mathbf{t}_i), 1 \leq i \leq n\}, \\
 \text{singl}(\mathbf{f}(\mathbf{t}_1, \dots, \mathbf{t}_n)) &:= \{f(\mathbf{s}_1, \dots, \mathbf{s}_n) \mid f \in \mathbf{f}, \mathbf{s}_i \in \text{singl}(\mathbf{t}_i), 1 \leq i \leq n\}, \text{ where } \mathbf{f} \notin \mathcal{N}.
 \end{aligned}$$

This concept is also applicable to X-substitutions:

$$\text{singl}(\boldsymbol{\mu}) := \{\theta \mid \mathbf{x}\theta \in \text{singl}(\mathbf{x}\boldsymbol{\mu}) \text{ for all } \mathbf{x} \in \mathcal{V}\}.$$

A finite mapping $\Phi : \mathcal{N} \rightarrow Nb \setminus \mathcal{N}$ associating names to non-name neighborhoods is called a *name-neighborhood mapping*. A name $\mathbf{n} \in \text{dom}(\Phi)$ is mapped to a finite set of constants if its arity is 0, or to a finite set of unranked function symbols if it is unranked. Applying a name-neighborhood mapping Φ to an X-term \mathbf{t} will result in an X-term $\Phi(\mathbf{t})$ where each name \mathbf{n} is replaced by $\Phi(\mathbf{n})$. Using a name-neighborhood mapping on a sequence of X-terms applies it to each of the X-terms. Applying Φ to a set of unification problems Γ means applying it to both sides of each of the unification problems.

For finite sets of function symbols \mathbf{f} and \mathbf{g} , we denote by $\mathcal{R}(\mathbf{f}, \mathbf{g})$ the number $\min\{\mathcal{R}(f, g) \mid f \in \mathbf{f}, g \in \mathbf{g}\}$.

The following notions are taken from [35]:

Definition 3.6 (Neighborhood equation)

Given a proximity relation \mathcal{R} , a cut value λ and two neighborhoods \mathbf{f} and \mathbf{g} , an (\mathcal{R}, λ) -neighborhood equation is written as $\mathbf{f} \approx_{\mathcal{R}, \lambda}^? \mathbf{g}$, where the question mark denotes that this equation is still to be solved.

We say that a name-neighborhood mapping Φ is a solution to an (\mathcal{R}, λ) -neighborhood equation $\mathbf{f} \approx_{\mathcal{R}, \lambda}^? \mathbf{g}$, if $\mathcal{R}(\Phi(\mathbf{f}), \Phi(\mathbf{g})) \geq \lambda$.

A finite set of (\mathcal{R}, λ) -equations is called an (\mathcal{R}, λ) -constraint. If a name-neighborhood mapping Φ is an (\mathcal{R}, λ) -solution for all (\mathcal{R}, λ) -equations in a constraint C , then Φ is called an (\mathcal{R}, λ) -solution of C .

Definition 3.7 (X-Unification)

Let \mathcal{R} be a proximity relation and λ a cut value. An (\mathcal{R}, λ) -X-unification problem Γ is a finite set of (\mathcal{R}, λ) -X-equations $\tilde{\mathbf{s}} \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{t}}$.

An (\mathcal{R}, λ) -solution of an (\mathcal{R}, λ) -X-equation $\tilde{\mathbf{s}} \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{t}}$ is a pair (Φ, μ) with Φ being a name-neighborhood mapping and μ being an X-substitution, such that $\mathcal{R}(\Phi(\tilde{\mathbf{s}}\mu), \Phi(\tilde{\mathbf{t}}\mu)) \geq \lambda$.

If (Φ, μ) is an (\mathcal{R}, λ) -solution of all equations in Γ , then we say that (Φ, μ) is an (\mathcal{R}, λ) -solution or an (\mathcal{R}, λ) -X-unifier of Γ .

The same concepts are defined for SX-terms, in particular. Our unification problems will actually use only SX-terms. Therefore, the notions relevant to those problems are formulated for SX-terms below.

Definition 3.8 (Occurrence Cycle)

A set of SX-equations $\{\mathbf{x} \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{t}}\} \cup \Gamma$ contains an occurrence cycle for the variable \mathbf{x} if $\tilde{\mathbf{t}} \notin \mathcal{V}$ and there exist SX-terms pairs $(\mathbf{x}_1, \tilde{\mathbf{r}}_1), \dots, (\mathbf{x}_n, \tilde{\mathbf{r}}_n)$ with $\mathbf{x}_1 = \mathbf{x}$ and $\tilde{\mathbf{r}}_1 = \tilde{\mathbf{t}}$, such that for each $1 \leq i \leq n$, Γ contains an equation $\mathbf{x}_i \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{r}}_i$ or $\tilde{\mathbf{r}}_i \simeq_{\mathcal{R}, \lambda}^? \mathbf{x}_1$, at least one $\tilde{\mathbf{r}}_i$ is not a variable, and $\mathbf{x}_{i+1} \in \mathcal{V}(\tilde{\mathbf{r}}_i)$ where $\mathbf{x}_{n+1} = \mathbf{x}_1$.

Lemma 3.10

A set of SX-equations Γ has no non-erasing (\mathcal{R}, λ) -solution for any \mathcal{R} and λ , if it contains an occurrence cycle for some variable.

Proof. Due to the fact that neighborhoods with different number of arguments can not be (\mathcal{R}, λ) -close, an SX-term can never be (\mathcal{R}, λ) -close to its own subterms. If we are not allowed to eliminate sequence variables, this means that there is no (\mathcal{R}, λ) -solution for an equation containing an occurrence cycle. \square

Example 3.2. Unification problems with occurrence cycles can have erasing solutions. For instance, the problem $\{\ulcorner \bar{x}, \bar{y} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner f(\bar{x}), a \urcorner\}$ can be solved by substitution $\{\bar{x} \mapsto \ulcorner \urcorner, \bar{y} \mapsto \ulcorner f(), a \urcorner\}$ for any \mathcal{R} and λ .

In this section we will not use (\mathcal{R}, λ) -instantiation relation $\lesssim_{\mathcal{R}, \lambda}$, as it would no longer be a quasi order with our proximity relation \mathcal{R} . Instead we use syntactically instantiation relation \preceq . It means that we will be talking about syntactic minimal complete sets of unifiers, which, therefore, may contain $\lesssim_{\mathcal{R}, \lambda}$ -comparable substitutions.

Similar to $\mathfrak{U}_{\text{sim}}$ we also allow for equations between sequences.

Unranked unification with proximity relations will be split into two parts, *pre-unification* and *constraint solving*, just as in [35] and [43]. This means the procedure goes along the following path:

- As the pre-unification algorithm works on SX-equations. We will take the equation between terms of the initial unification problem and treat it as an SX-equation. The pre-unification algorithm then either transforms it until we are left with a set V of equations between variables (a *variables-only system*), a neighborhood constraint C , and an X-substitution μ over $\mathcal{T}_X(\emptyset, \mathcal{N}, \mathcal{V})$ with $\text{dom}(\mu) \cap \mathcal{V}(V) = \emptyset$ or it fails.
- In order to solve the neighborhood constraint C under consideration of μ we will need the constraint solving algorithm. When successful, the algorithm creates a finite set of name-neighborhood mappings $\mathcal{M} = \{\Phi_1, \dots, \Phi_n\}$ where each maps all the names occurring in μ to sets of function symbols. However, if the procedure fails, then the initial unification problem is not solvable.
- The pair $(\Phi_i, \mu\nu)$ solves the initial unification problem, meaning $\Phi_i(\mu\nu)$ is an X-unifier, for each $\Phi_i \in \mathcal{M}$ and each X-unifier ν of V . From them, we can extract our set of solutions as $\text{singl}(\Phi_1(\mu\nu)) \cup \dots \cup \text{singl}(\Phi_n(\mu\nu))$.

The Pre-Unification algorithm builds its neighborhood constraint by creating new copies of each term as they are dealt with in the variable elimination rules. This is done via a *renaming function* ρ , which takes (SX-)terms and returns a term which only contains names and variables.

$$\begin{aligned}
 \rho(x) &:= x_i && \text{with } x_i \text{ being a fresh individual variable,} \\
 \rho(\bar{x}) &:= \bar{x}_i && \text{with } \bar{x}_i \text{ being a fresh sequence variable,} \\
 \rho(\mathbf{n}(s_1, \dots, s_n)) &:= \mathbf{n}_i(\rho(s_1), \dots, \rho(s_n)) && \text{with } \mathbf{n}_i \text{ being a fresh name,} \\
 \rho(f(s_1, \dots, s_n)) &:= \mathbf{n}_i(\rho(s_1), \dots, \rho(s_n)) && \text{with } \mathbf{n}_i \text{ being a fresh name.}
 \end{aligned}$$

Each occurrence of a name or a function symbol is replaced by a new name, and each occurrence of a variable is replaced by a fresh variable of the same kind.

Example 3.3. If we take the term $f(x, a, \mathbf{n}(x), \bar{x})$ (treated as an SX-term), and apply ρ to it, we get

$$\rho(f(x, a, \mathbf{n}(x), \bar{x})) = \mathbf{n}_1(x_1, \mathbf{n}_2, \mathbf{n}_3(x_2), \bar{x}_1),$$

where $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3$ are new names and x_1, x_2, \bar{x}_1 are fresh variables.

Given two sequences of terms \tilde{s} and \tilde{t} , the proximity-based pre-unification algorithm starts with the configuration $\langle \{\tilde{s} \simeq_{\mathcal{R}, \lambda} \tilde{t}\}; \emptyset; \varepsilon \rangle$, where \tilde{s} and \tilde{t} are viewed as SX-terms $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$. The rules that transform configurations are given below.

(\mathcal{R}, λ) -Pre-Unification

P-U_{prox}: **Projection**

$$\langle \Gamma; C; \mu \rangle \Longrightarrow \langle \Gamma\theta; C; \mu\theta \rangle,$$

with $\theta \neq \varepsilon$, $\text{dom}(\theta) \subseteq \mathcal{V}_S(\Gamma)$ and $\text{ran}(\theta) = \emptyset$.

T-U_{prox}: **Trivial**

$$\langle \{\bar{s} \simeq_{\mathcal{R}, \lambda}^? \tilde{s}\} \cup \Gamma; C; \mu \rangle \Longrightarrow \langle \Gamma; C; \mu \rangle.$$

O1-U_{prox}: **Orient 1**

$$\langle \{s \simeq_{\mathcal{R}, \lambda}^? x\} \cup \Gamma; C; \mu \rangle \Longrightarrow \langle \{x \simeq_{\mathcal{R}, \lambda}^? s\} \cup \Gamma; C; \mu \rangle,$$

if $s \notin \mathcal{V}$.

O2-U_{prox}: **Orient 2**

$$\langle \{\lceil s, s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \bar{x}, t_1, \dots, t_m \rceil\} \cup \Gamma; C; \mu \rangle \Longrightarrow$$

$$\langle \{\lceil \bar{x}, t_1, \dots, t_m \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil s, s_1, \dots, s_n \rceil\} \cup \Gamma; C; \mu \rangle,$$

if $s \notin \mathcal{V}_S$.

HR-U_{prox}: **Head Removal**

$$\langle \{f(s_1, \dots, s_n) \simeq_{\mathcal{R}, \lambda}^? g(t_1, \dots, t_m)\} \cup \Gamma; C; \mu \rangle \Longrightarrow$$

$$\langle \{\lceil s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t_1, \dots, t_m \rceil\} \cup \Gamma; C \cup \{f \simeq_{\mathcal{R}, \lambda}^? g\}; \mu \rangle.$$

DEC-U_{prox}: **Decomposition**

$$\langle \{\lceil s, s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t, t_1, \dots, t_m \rceil\} \cup \Gamma; C; \mu \rangle$$

$$\Longrightarrow \langle \{s \simeq_{\mathcal{R}, \lambda}^? t, \lceil s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t_1, \dots, t_m \rceil\} \cup \Gamma; C; \mu \rangle,$$

if $s, t \notin \mathcal{V}_S$ and $\lceil s, s_1, \dots, s_n \rceil \neq \lceil t, t_1, \dots, t_m \rceil$ with $n + m > 0$.

IVE-U_{prox}: **Individual Variable Elimination**

$$\langle \{x \simeq_{\mathcal{R}, \lambda}^? t\} \cup \Gamma; C; \mu \rangle \Longrightarrow \langle \{t' \simeq_{\mathcal{R}, \lambda}^? t\} \cup \Gamma\theta; C; \mu\theta \rangle,$$

if there is no occurrence cycle for x in $\{x \simeq_{\mathcal{R}, \lambda}^? t\} \cup \Gamma$, $t \notin \mathcal{V}$, $\theta = \{x \mapsto t'\}$ and $t' = \rho(t)$.

SVS-U_{prox}: **Sequence Variable Separation**

$$\langle \{\lceil \bar{x}, s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil y, t_1, \dots, t_m \rceil\} \cup \Gamma; C; \mu \rangle \Longrightarrow$$

$$\langle \{\bar{x} \simeq_{\mathcal{R}, \lambda}^? y, \lceil s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t_1, \dots, t_m \rceil\} \cup \Gamma; C; \mu \rangle$$

for $\bar{x} \neq y$ and $n + m > 0$.

SVE1-U_{prox}: **Sequence Variable Elimination 1**

$$\langle \{\lceil \bar{x}, s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \bar{x}, t_1, \dots, t_m \rceil\} \cup \Gamma; C; \mu \rangle \Longrightarrow$$

$$\langle \{\lceil s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t_1, \dots, t_m \rceil\} \cup \Gamma; C; \mu \rangle.$$

SVE2-U_{prox}: **Sequence Variable Elimination 2**

$$\langle \{\lceil \bar{x}, s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t, t_1, \dots, t_m \rceil\} \cup \Gamma; C; \mu \rangle \Longrightarrow$$

$$\langle \{t' \simeq_{\mathcal{R}, \lambda}^? t, \lceil s_1 \theta, \dots, s_n \theta \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t_1 \theta, \dots, t_m \theta \rceil\} \cup \Gamma\theta; C; \mu\theta \rangle,$$

if there is no occurrence cycle for \bar{x} in $\{\lceil \bar{x}, s_1, \dots, s_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil t, t_1, \dots, t_m \rceil\} \cup \Gamma$, $t \notin \mathcal{V}$, $\theta = \{\bar{x} \mapsto t'\}$ and $t' = \rho(t)$.

W1-U_{prox}: Widening 1

$$\langle \{\lceil \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \rceil\} \cup \Gamma; C; \boldsymbol{\mu} \rangle \implies$$

$$\langle \{\mathbf{t}' \simeq_{\mathcal{R}, \lambda}^? \mathbf{t}, \lceil \bar{x}', \mathbf{s}_1 \boldsymbol{\theta}, \dots, \mathbf{s}_n \boldsymbol{\theta} \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \mathbf{t}_1 \boldsymbol{\theta}, \dots, \mathbf{t}_m \boldsymbol{\theta} \rceil\} \cup \Gamma \boldsymbol{\theta}; C; \boldsymbol{\mu} \boldsymbol{\theta} \rangle,$$

if there is no occurrence cycle for \bar{x} in $\{\lceil \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \rceil\} \cup \Gamma$, $n+m > 0$,
 $\boldsymbol{\theta} = \{\bar{x} \mapsto \lceil \mathbf{t}', \bar{x}' \rceil\}$ with $\mathbf{t}' = \rho(\mathbf{t})$ and \bar{x}' being a fresh variable.

W2-U_{prox}: Widening 2

$$\langle \{\lceil \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \bar{y}, \mathbf{t}_1, \dots, \mathbf{t}_m \rceil\} \cup \Gamma; C; \boldsymbol{\mu} \rangle \implies$$

$$\langle \{\lceil \bar{x} \simeq_{\mathcal{R}, \lambda}^? \bar{x}', \mathbf{s}_1 \boldsymbol{\theta}, \dots, \mathbf{s}_n \boldsymbol{\theta} \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \bar{y}', \mathbf{t}_1 \boldsymbol{\theta}, \dots, \mathbf{t}_m \boldsymbol{\theta} \rceil\} \cup \Gamma \boldsymbol{\theta}; C; \boldsymbol{\mu} \boldsymbol{\theta} \rangle,$$

with $n+m > 0$, $\boldsymbol{\theta} = \{\bar{y} \mapsto \lceil \bar{x}', \bar{y}' \rceil\}$ and \bar{x}' and \bar{y}' being fresh variables.

IVOCC-U_{prox}: Individual Variable Occurrence Cycle Check

$$\langle \{x \simeq_{\mathcal{R}, \lambda}^? \mathbf{t}\} \cup \Gamma; C; \boldsymbol{\mu} \rangle \implies \perp,$$

if there is an occurrence cycle for x in $\{x \simeq_{\mathcal{R}, \lambda}^? \mathbf{t}\} \cup \Gamma$ and $x \neq t$.

SVOCC-U_{prox}: Sequence Variable Occurrence Cycle Check

$$\langle \{\lceil \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \rceil\} \cup \Gamma; C; \boldsymbol{\mu} \rangle \implies \perp,$$

if there is an occurrence cycle for \bar{x} in $\{\lceil \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \rceil\} \cup \Gamma$ and
 $\bar{x} \neq \mathbf{t}$.

E1-U_{sim}: Empty 1

$$\langle \{\lceil \lceil \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_n \rceil \rceil\} \cup \Gamma'; C; \boldsymbol{\mu} \rangle \implies \perp$$

with $\mathbf{t} \notin \mathcal{V}_S$.

E2-U_{sim}: Empty 2

$$\langle \{\lceil \lceil \mathbf{s}, \mathbf{s}_1, \dots, \mathbf{s}_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \lceil \rceil \rceil \rceil\} \cup \Gamma'; C; \boldsymbol{\mu} \rangle \implies \perp$$

with $\mathbf{s} \notin \mathcal{V}_S$.

Just as with \mathcal{U} , termination is not guaranteed, unless some additional restrictions are imposed on the unification problem.

Definition 3.9

A solution of an (\mathcal{R}, λ) -configuration $\langle \Gamma; C; \boldsymbol{\mu} \rangle$ is a mapping-substitution pair $(\Phi, \boldsymbol{\nu})$, which fulfills the following conditions:

- $(\Phi, \boldsymbol{\nu})$ is an (\mathcal{R}, λ) -solution of Γ ;
- Φ is an (\mathcal{R}, λ) -solution of C ;
- for each $\mathbf{x} \in \text{dom}(\boldsymbol{\mu})$, we have $\mathbf{x}\boldsymbol{\nu} = \mathbf{x}\boldsymbol{\mu}\boldsymbol{\nu}$.

Next, we want to prove soundness and completeness for Pre-Unification:

Lemma 3.11

Let \mathcal{C} be a configuration $\langle \Gamma_1; C_1; \mu_1 \rangle$.

1. If $\mathcal{C} \Longrightarrow \perp$ by (IVOCC- U_{prox}) or (SVOCC- U_{prox}) rules, then \mathcal{C} does not have a non-erasing solution.
2. If $\mathcal{C} \Longrightarrow \perp$ by (E1- U_{prox}) or (E2- U_{prox}) rules, then \mathcal{C} does not have a solution.
3. If $\mathcal{C} \Longrightarrow \langle \Gamma_2; C_2; \mu_2 \rangle$ is a step performed by a non-failing pre-unification rule, then every solution of $\langle \Gamma_2; C_2; \mu_2 \rangle$ is a solution of \mathcal{C} .

Proof.

1. Occurrence cycles can be solved with non-erasing substitutions.
2. Obvious.
3. We will show the proof of some of the non-trivial rules:

(HR- U_{prox}) We have

$$\begin{aligned} \Gamma_1 &= \{\mathbf{f}(\mathbf{s}_1, \dots, \mathbf{s}_n) \simeq_{\mathcal{R}, \lambda}^? \mathbf{g}(\mathbf{t}_1, \dots, \mathbf{t}_m)\} \cup \Gamma'_1, \\ \Gamma_2 &= \{\ulcorner \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner\} \cup \Gamma'_1, \\ C_2 &= C_1 \cup \{\mathbf{f} \approx_{\mathcal{R}, \lambda}^? \mathbf{g}\}, \\ \mu_2 &= \mu_1. \end{aligned}$$

Since the question whether \mathbf{f} and \mathbf{g} are (\mathcal{R}, λ) -close only moves from Γ to C , the configurations $\langle \Gamma_1; C_1; \mu_1 \rangle$ and $\langle \Gamma_2; C_2; \mu_2 \rangle$ have the same set of solutions.

(SVE2- U_{prox}) We have

$$\begin{aligned} \Gamma_1 &= \{\ulcorner \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_2 \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner\} \cup \Gamma'_1, \\ \Gamma_2 &= \{\mathbf{t}' \simeq_{\mathcal{R}, \lambda}^? \mathbf{t}, \ulcorner \mathbf{s}_1 \boldsymbol{\theta}, \dots, \mathbf{s}_n \boldsymbol{\theta} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}_1 \boldsymbol{\theta}, \dots, \mathbf{t}_m \boldsymbol{\theta} \urcorner\} \cup \Gamma'_1 \boldsymbol{\theta}, \\ C_2 &= C_1, \\ \mu_2 &= \mu_1 \boldsymbol{\theta}, \end{aligned}$$

with $\boldsymbol{\theta}_2 = \{\bar{x} \mapsto \mathbf{t}'\}$, $\mathbf{t}' = \rho(\mathbf{t})$, $\mathbf{t} \notin \mathcal{V}_S$ and there is no occurrence cycle for \bar{x} in Γ_1 . Let (Φ, \mathbf{v}) be a solution of $\langle \Gamma_2; C_2; \mu_2 \rangle$. From the fact that μ_2 contains $\boldsymbol{\theta}$, since $\bar{x} \notin \text{dom}(\mu_1)$, we know that $\Phi(\bar{x}\mathbf{v}) = \Phi(\bar{x}\mu_2\mathbf{v}) = \Phi(\mathbf{t}'\mathbf{v})$. Additionally we know from \mathbf{v} solving $\mathbf{t}' \simeq_{\mathcal{R}, \lambda}^? \mathbf{t}$ and $\ulcorner \mathbf{s}_1 \boldsymbol{\theta}, \dots, \mathbf{s}_n \boldsymbol{\theta} \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}_1 \boldsymbol{\theta}, \dots, \mathbf{t}_m \boldsymbol{\theta} \urcorner$, that $\mathcal{R}(\Phi(\mathbf{t}'\mathbf{v}), \Phi(\mathbf{t}\mathbf{v})) \geq \lambda$ and therefore it holds that $\mathcal{R}(\Phi(\bar{x}\mathbf{v}), \Phi(\mathbf{t}\mathbf{v})) \geq \lambda$, meaning that (Φ, \mathbf{v}) is a solution for the equation $\bar{x} \simeq_{\mathcal{R}, \lambda}^? \mathbf{t}'$ and therefore also for $\ulcorner \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_2 \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner$.

For any equation $eq \in \Gamma'_1$ we know that $eq\mathbf{v} = eq\boldsymbol{\theta}\nu$ holds as we already know that $\bar{x}\mathbf{v} = \mathbf{t}'\mathbf{v} = \bar{x}\boldsymbol{\theta}\nu$, and for those variables $\mathbf{v} \neq \bar{x}$ it is clear that $\mathbf{v}\mathbf{v} = \mathbf{v}\boldsymbol{\theta}\nu$. Therefore we can say that (Φ, \mathbf{v}) is a solution for Γ'_1 .

For any $\mathbf{v} \in \text{dom}(\boldsymbol{\mu}_1)$ we have $\mathbf{v}\mathbf{v} = \mathbf{v}\boldsymbol{\mu}_2\mathbf{v} = \mathbf{v}\boldsymbol{\mu}_1\boldsymbol{\theta}\nu = \mathbf{v}\boldsymbol{\mu}_1\mathbf{v}$. Hence, (Φ, \mathbf{v}) is a solution of $\langle \Gamma_1; C; \boldsymbol{\mu}_1 \rangle$.

□

Lemma 3.12 (Soundness of Pre-Unification)

Let \tilde{s} and \tilde{t} be two sequences of terms such that our pre-unification algorithm constructs the derivation from their SX-forms $\langle \{\tilde{s} \simeq_{\mathcal{R}, \lambda}^? \tilde{t}\}; \emptyset; \varepsilon \rangle \Longrightarrow^* \langle V; C; \boldsymbol{\mu} \rangle$, with V being a variables-only constraint, and let (Φ, \mathbf{v}) be an (\mathcal{R}, λ) -solution of the configuration $\langle V; C; \boldsymbol{\mu} \rangle$. The (\mathcal{R}, λ) -X-unifier of $\{\tilde{s} \simeq_{\mathcal{R}, \lambda}^? \tilde{t}\}$ is the X-substitution $\Phi(\boldsymbol{\mu}\nu)$.

Proof. The proof idea remains the same as in [43]: We have to show that $(\Phi, \boldsymbol{\mu}\nu)$ is an (\mathcal{R}, λ) -solution of $\langle V; C; \boldsymbol{\mu} \rangle$. Knowing that $\text{dom}(\boldsymbol{\mu}) \cap \mathcal{V}(V) = \emptyset$, we get that $(\Phi, \boldsymbol{\mu}\nu)$ is a solution of V , since $V\boldsymbol{\mu} = V$ and (Φ, \mathbf{v}) being an (\mathcal{R}, λ) -solution of $\langle V; C; \boldsymbol{\mu} \rangle$. From $\boldsymbol{\mu}$ being idempotent we can say that $x\boldsymbol{\mu}\nu = x\boldsymbol{\mu}\boldsymbol{\mu}\nu$. This means that $(\Phi, \boldsymbol{\mu}\nu)$ is an (\mathcal{R}, λ) -solution of $\langle V; C; \boldsymbol{\mu} \rangle$.

Using Lemma 3.11 we get from $\langle \{\tilde{s} \simeq_{\mathcal{R}, \lambda}^? \tilde{t}\}; \emptyset; \varepsilon \rangle \Longrightarrow^* \langle V; C; \boldsymbol{\mu} \rangle$ that $(\Phi, \boldsymbol{\mu}\nu)$ is an (\mathcal{R}, λ) -solution to $\langle \{\tilde{s} \simeq_{\mathcal{R}, \lambda}^? \tilde{t}\}; \emptyset; \varepsilon \rangle$.

We know that $\Phi(\tilde{s}\boldsymbol{\mu}\nu) = \tilde{s}\Phi(\boldsymbol{\mu}\nu)$ and $\Phi(\tilde{t}\boldsymbol{\mu}\nu) = \tilde{t}\Phi(\boldsymbol{\mu}\nu)$ since \tilde{s} and \tilde{t} do not contain any names. Combining this with the fact that $\Phi(\tilde{s}\boldsymbol{\mu}\nu) = \Phi(\tilde{t}\boldsymbol{\mu}\nu)$ we get that $\tilde{s}\Phi(\boldsymbol{\mu}\nu) = \tilde{t}\Phi(\boldsymbol{\mu}\nu)$, i.e. $\Phi(\boldsymbol{\mu}\nu)$ is an (\mathcal{R}, λ) -X-solution of $\{\tilde{s} \simeq_{\mathcal{R}, \lambda}^? \tilde{t}\}$. □

Corollary 3.13

Let \tilde{s} and \tilde{t} be two sequences of terms, such that the pre-unification algorithm constructs $\langle \{\tilde{s} \simeq_{\mathcal{R}, \lambda}^? \tilde{t}\}; \emptyset; \varepsilon \rangle \Longrightarrow^* \langle V; C; \boldsymbol{\mu} \rangle$. Let Φ be an (\mathcal{R}, λ) -solution of C and let (Φ, \mathbf{v}) be an (\mathcal{R}, λ) -solution of $\langle V; C; \boldsymbol{\mu} \rangle$ with \mathbf{v} being name-free. Then every substitution in $\text{singl}(\Phi(\boldsymbol{\mu}\nu))$ is an (\mathcal{R}, λ) -unifier of \tilde{s} and \tilde{t} .

Proof. This follows directly from Lemma 3.12 and the way singl is defined. Since \mathbf{v} does not contain any names and since we know from the construction of C that Φ maps all names occurring in C to sets of function symbols, we get that $\text{singl}(\Phi(\boldsymbol{\mu}\nu))$ is a set of substitutions, not SX-substitutions. □

We take the following proposition from [43] as is and prove it for our modified rules.

Proposition 3.14

Let a substitution σ be an (\mathcal{R}, λ) -unifier of two sequences of terms \tilde{s} and \tilde{t} with $\mathcal{X} = \mathcal{V}(\tilde{s}) \cup \mathcal{V}(\tilde{t})$ being the set of variables occurring in \tilde{s} and \tilde{t} . Consider a maximal derivation constructed by the Pre-Unification algorithm that starts from the SX-versions of \tilde{s} and \tilde{t} . Assume $\langle \Gamma_1; C_1; \mu_1 \rangle \Longrightarrow_{BT} \langle \Gamma_2; C_2; \mu_2 \rangle$ is a step in this derivation. If (Φ_1, θ_1) is an idempotent and non-erasing (\mathcal{R}, λ) -solution of $\langle \Gamma_1; C_1; \mu_1 \rangle$ with $\sigma|_{\mathcal{X}} = \theta_1|_{\mathcal{X}}$ for $\theta_1 \in \text{subs}(\Phi_1(\theta_1))$, then $\langle \Gamma_2; C_2; \mu_2 \rangle$ has an idempotent and non-erasing (\mathcal{R}, λ) -solution (Φ_2, θ_2) with $\sigma|_{\mathcal{X}} = \theta_2|_{\mathcal{X}}$ for $\theta_2 \in \text{subs}(\Phi_2(\theta_2))$.

Proof. We assume that $\text{dom}(\sigma) \subseteq \mathcal{X}$ and prove each of the rules separately:

(TRI- U_{prox}) We take $\Phi_1 = \Phi_2$ and $\theta_2 = \theta_1$.

(O1- U_{prox}) Straightforward.

(O2- U_{prox}) Straightforward.

(HR- U_{prox}) We have

- $\Gamma_1 = \{\mathbf{f}(\mathbf{s}_1, \dots, \mathbf{s}_n) \simeq_{\mathcal{R}, \lambda}^? \mathbf{g}(\mathbf{t}_1, \dots, \mathbf{t}_m)\} \cup \Gamma'_1,$
- $\Gamma_2 = \{\ulcorner \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner\} \cup \Gamma'_1,$
- $C_2 = C_1 \cup \{\mathbf{f} \approx_{\mathcal{R}, \lambda}^? \mathbf{g}\},$
- $\mu_1 = \mu_1.$

It is clear, that since (Φ_1, θ_1) is an (\mathcal{R}, λ) -solution of $\langle \Gamma_1; C_1; \mu_1 \rangle$, it is also an (\mathcal{R}, λ) -solution of $\langle \Gamma_2; C_2; \mu_2 \rangle$. Therefore we choose $\Phi_2 = \Phi_1$ and $\mu_2 = \mu_1$.

(DEC- U_{prox}) We have

- $\Gamma_1 = \{\ulcorner \mathbf{s}, \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner\} \cup \Gamma'_1$ with $\mathbf{s}, \mathbf{t} \notin \mathcal{V}_S$ and $\ulcorner \mathbf{s}, \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \neq \ulcorner \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner$ and $n + m > 0,$
- $\Gamma_2 = \{\mathbf{s} \simeq_{\mathcal{R}, \lambda}^? \mathbf{t}, \ulcorner \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner\} \cup \Gamma'_1,$
- $C_2 = C_1,$
- $\mu_2 = \mu_1.$

Again, since we know that (Φ_1, θ_1) is an (\mathcal{R}, λ) -solution of $\langle \Gamma; C_1; \mu_1 \rangle$, it is also an (\mathcal{R}, λ) -solution of $\langle \Gamma_2; C_2; \mu_2 \rangle$ and we take $\Phi_2 = \Phi_1$ and $\theta_2 = \theta_1$.

(IVE- U_{prox}) See [43].

(SVS- U_{prox}) Similar to (DEC- U_{prox}) it is clear that we use $\Phi_2 = \Phi_1$ and $\theta_2 = \theta_1$.

(SVE1-U_{prox}) We have

- $\Gamma_1 = \{\ulcorner \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \bar{x}, \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner\} \cup \Gamma'_1,$
- $\Gamma_2 = \{\ulcorner \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner\} \cup \Gamma'_1,$
- $C_2 = C_1,$
- $\mu_2 = \mu_1.$

As this rule is quite similar to (TRI-U_{prox}) it is clear that (Φ_1, θ_1) is a solution to $\langle \Gamma_2; C_2; \mu_2 \rangle$ and therefore we take $\Phi_2 = \Phi_1$ and $\theta_2 = \theta_1$.

(SVE2-U_{prox}) We have

- $\Gamma_1 = \{\ulcorner \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner\} \cup \Gamma'_1,$
- $\Gamma_2 = \{\ulcorner \mathbf{t}' \simeq_{\mathcal{R}, \lambda}^? \mathbf{t}, \mathbf{s}_1 \theta, \dots, \mathbf{s}_n \theta \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}_1 \theta, \dots, \mathbf{t}_m \theta \urcorner\} \cup \Gamma'_1 \theta,$
- $C_2 = C_1,$
- $\mu_2 = \mu_1 \theta.$

This rule can be done analogously to (IVE-U_{prox}).

(W1-U_{prox}) We have

- $\Gamma_1 = \{\ulcorner \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner\} \cup \Gamma'_1,$ where there is no occurrence cycle for \bar{x} in Γ_1 and $n + m > 0,$
- $\Gamma_2 = \{\ulcorner \mathbf{t}' \simeq_{\mathcal{R}, \lambda}^? \mathbf{t}, \ulcorner \bar{x}', \mathbf{s}_1 \theta, \dots, \mathbf{s}_n \theta \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}_1 \theta, \dots, \mathbf{t}_m \theta \urcorner\} \cup \Gamma'_1 \theta,$
- $C_2 = C_1,$
- $\mu_2 = \mu_1 \theta$ with $\theta = \{\bar{x} \mapsto \ulcorner \mathbf{t}', \bar{x}' \urcorner\}, \mathbf{t}' = \rho(\mathbf{t})$ and \bar{x}' being a new variable.

From (Φ_1, θ_1) being an (\mathcal{R}, λ) -solution of $\langle \Gamma_1; C_1; \mu_1 \rangle$, we know that

$$\Phi_1(\ulcorner \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \theta_1) \simeq_{\mathcal{R}, \lambda}^? \Phi_1(\ulcorner \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner \theta_1).$$

We start by constructing Φ_2 and θ_2 from Φ_1 and θ_1 respectively. Since (Φ_1, θ_1) is a solution of Γ_1 and $\ulcorner \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \urcorner \simeq_{\mathcal{R}, \lambda}^? \ulcorner \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \urcorner \in \Gamma_1$, we know that $\bar{x} \in \text{dom}(\theta_1)$. Since $\mathbf{t}' = \rho(\mathbf{t})$, we know that each symbol occurring in \mathbf{t}' is either a name or variable. Let $\tilde{\mathbf{r}} = \Phi_1(\bar{x} \theta_1) = \ulcorner \mathbf{r}_1, \dots, \mathbf{r}_k \urcorner$, then $\text{pos}(\mathbf{t}') \subseteq \text{pos}(\mathbf{r}_1)$ since \mathbf{t}' has to be unifiable to the first term in $\tilde{\mathbf{r}}$. We will denote set of positions where a name (variable) occurs in \mathbf{t}' with $\text{pos}_{\mathcal{N}}(\mathbf{t}')$ ($\text{pos}_{\mathcal{V}}(\mathbf{t}')$). Let

$$\Phi' := \{\mathbf{n} \mapsto \mathbf{f} \mid \mathbf{n} = \mathbf{t}'|_p, \mathbf{f} = \mathbf{r}_1|_p, p \in \text{pos}_{\mathcal{N}}(\mathbf{t}')\},$$

$$\theta' := [\mathbf{v}_1 \mapsto \mathbf{s}_1; \dots; \mathbf{v}_l \mapsto \mathbf{s}_l], \mathbf{v}_i = \mathbf{t}'|_{p_i}, \mathbf{s}_i = \mathbf{r}_1|_{p_i}, \{p_1, \dots, p_n\} = \text{pos}_V(\mathbf{t}');$$

and set

$$\begin{aligned} \Phi_2 &:= \Phi_1 \cup \Phi', \\ \theta_2 &:= [\text{rep}(\theta_1, \{\bar{x} \mapsto \bar{x}\theta_1\}) \rightsquigarrow \{\bar{x} \mapsto \ulcorner \mathbf{t}', \mathbf{q}_2, \dots, \mathbf{q}_k \urcorner\}]; \theta'], \\ \bar{x}\theta_1 &= \tilde{\mathbf{q}} = \ulcorner \mathbf{q}_1, \dots, \mathbf{q}_k \urcorner. \end{aligned}$$

Since $\text{dom}(\Phi_1) \cap \text{dom}(\Phi') = \emptyset$ we get that Φ_2 is well-defined. The property of being idempotent carries over to θ_2 , since we assume θ_1 to be idempotent and therefore the variables occurring in the range of θ' are not a part of the domain of θ_1 .

Having constructed Φ_2 and θ_2 , we now have to prove the following:

Proving that (Φ_2, θ_2) is a solution of Γ_2 : It holds that $\Phi_2(\mathbf{y}\theta_2) = \Phi_1(\mathbf{y}\theta_1)$ for all $\mathbf{y} \in \text{dom}(\theta_1) \setminus \{\bar{x}\}$ from the way we designed Φ_2 and θ_2 . We also have to show the same for \bar{x} and those variables in $\text{dom}(\theta_2) \setminus \text{dom}(\theta_1)$, i.e. $\{\mathbf{v}_1, \dots, \mathbf{v}_l\} \in \text{dom}(\theta')$. This can we done in one go as we get

$$\Phi_2(\bar{x}\theta_2) = \Phi_2(\ulcorner \mathbf{t}', \mathbf{q}_2, \dots, \mathbf{q}_k \urcorner) = \tilde{\mathbf{r}} = \Phi_1(\bar{x}\theta_1).$$

Since the only difference between Γ_1 and Γ_2 is the replacement of \bar{x} with $\ulcorner \mathbf{t}', \bar{x}' \urcorner$, it holds that (Φ_2, θ_2) is a solution of Γ_2 .

Proving that Φ_2 is a solution of C_2 : Since we have $\Phi_2 = \Phi_1 \cup \Phi'$ and $C_2 = C_1$ this holds.

Proving $\mathbf{y}\theta_2 = \mathbf{y}\mu_2\theta_2$ for all $\mathbf{y} \in \text{dom}(\mu_2)$: Let $\mathbf{y} \in \text{dom}(\mu_2)$ with $\mathbf{y} \neq \bar{x}$. We get

$$\begin{aligned} \mathbf{y}\theta_2 &= \mathbf{y}[\text{rep}(\theta_1, \{\bar{x} \mapsto \bar{x}\theta_1\}) \rightsquigarrow \{\bar{x} \mapsto \ulcorner \mathbf{t}', \mathbf{q}_2, \dots, \mathbf{q}_k \urcorner\}]; \theta'] \\ &= \mathbf{y}[\text{rep}(\theta_1, \{\bar{x} \mapsto \bar{x}\theta_1\}) \rightsquigarrow \{\bar{x} \mapsto \ulcorner \mathbf{t}'\theta', \mathbf{q}_2, \dots, \mathbf{q}_k \urcorner\}], \end{aligned}$$

since θ' only changes the variables occurring in \mathbf{t}' . On the other side we have

$$\begin{aligned} \mathbf{y}\mu_2\theta_2 &= \mathbf{y}\mu_1\{\bar{x} \mapsto \ulcorner \mathbf{t}', \bar{x}' \urcorner\}[\text{rep}(\theta_1, \{\bar{x} \mapsto \bar{x}\theta_1\}) \rightsquigarrow \{\bar{x} \mapsto \ulcorner \mathbf{t}', \mathbf{q}_2, \dots, \mathbf{q}_k \urcorner\}]; \theta'] \\ &= \mathbf{y}\mu_1\{\bar{x} \mapsto \ulcorner \mathbf{t}', \bar{x}' \urcorner\} \text{rep}(\theta_1, \{\bar{x} \mapsto \bar{x}\theta_1\}) \rightsquigarrow \{\bar{x} \mapsto \ulcorner \mathbf{t}', \mathbf{q}_2, \dots, \mathbf{q}_k \urcorner\} \theta' \\ &= \mathbf{y}\mu_1[\text{rep}(\theta_1, \{\bar{x} \mapsto \bar{x}\theta_1\}) \rightsquigarrow \{\bar{x} \mapsto \ulcorner \mathbf{t}'\theta', \mathbf{q}_2, \dots, \mathbf{q}_k \urcorner\}]. \end{aligned}$$

This is again due to θ' only changing the variables occurring in \mathbf{t}' . We know that $\mathbf{y}\theta_1 = \mathbf{y}\mu_1\theta_1$ and since the only difference between $\mathbf{y}\theta_2$ and $\mathbf{y}\theta_1$ as well as the only difference

between $\mathbf{y}\mu_2\theta_2$ and $\mathbf{y}\mu_1\theta_1$ is the replacement of occurrences of $\bar{x}\theta_1$ by $\lceil \mathbf{t}'\theta', \mathbf{q}_2, \dots, \mathbf{q}_k \rceil$, we get that $\mathbf{y}\theta_2 = \mathbf{y}\mu_2\theta_2$.

Let $\mathbf{y} = \bar{x}$. We get

$$\begin{aligned}
 \bar{x}\theta_2 &= \bar{x}[\text{rep}(\theta_1, \{\bar{x} \mapsto \bar{x}\theta_1\}) \rightsquigarrow \{\bar{x} \mapsto \lceil \mathbf{t}', \mathbf{q}_2, \dots, \mathbf{q}_k \rceil\}]; \theta' \\
 &= \bar{x}[\text{rep}(\theta_1, \{\bar{x} \mapsto \bar{x}\theta_1\}) \rightsquigarrow \{\bar{x} \mapsto \lceil \mathbf{t}'\theta', \mathbf{q}_2, \dots, \mathbf{q}_k \rceil\}] \\
 &= \lceil \mathbf{t}'\theta, \mathbf{q}_2, \dots, \mathbf{q}_k \rceil. \\
 \bar{x}\mu_2\theta_2 &= \bar{x}\{\bar{x} \mapsto \lceil \mathbf{t}', \bar{x}' \rceil\}[\text{rep}(\theta_1, \{\bar{x} \mapsto \bar{x}\theta_1\}) \rightsquigarrow \{\bar{x} \mapsto \lceil \mathbf{t}', \mathbf{q}_2, \dots, \mathbf{q}_k \rceil\}]; \theta' \\
 &= \bar{x}\{\bar{x} \mapsto \lceil \mathbf{t}', \bar{x}' \rceil\} \text{rep}(\theta_1, \{\bar{x} \mapsto \bar{x}\theta_1\}) \rightsquigarrow \{\bar{x} \mapsto \lceil \mathbf{t}', \mathbf{q}_2, \dots, \mathbf{q}_k \rceil\} \theta' \\
 &= \bar{x}\{\bar{x} \mapsto \lceil \mathbf{t}'\theta', \mathbf{q}_2, \dots, \mathbf{q}_k \rceil\} \\
 &= \lceil \mathbf{t}'\theta', \mathbf{q}_2, \dots, \mathbf{q}_k \rceil.
 \end{aligned}$$

It is easy to see that $\bar{x}\theta_2 = \bar{x}\mu_2\theta_2$.

Proving that $\theta_2|_{\mathcal{X}} = \sigma|_{\mathcal{X}}$ for some $\theta_2 \in \text{subs}(\Phi_2(\theta_2))$: Since $\Phi_2(\mathbf{y}\theta_2) = \Phi_1(\mathbf{y}\theta_1)$ for all $\mathbf{y} \in \text{dom}(\theta_1)$ and none of the newly introduced variables in $\text{dom}(\theta')$ occurring in \mathcal{X} , we get that $\theta_2|_{\mathcal{X}} = \theta_1|_{\mathcal{X}} = \sigma|_{\mathcal{X}}$.

(W2-U_{prox}) We have

- $\Gamma_1 = \{\lceil \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \bar{y}, \mathbf{t}_1, \dots, \mathbf{t}_m \rceil\} \cup \Gamma'_1$ with $n + m > 0$,
- $\Gamma_2 = \{\bar{x} \simeq_{\mathcal{R}, \lambda}^? \bar{x}', \lceil \mathbf{s}_1\theta, \dots, \mathbf{s}_n\theta \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \bar{y}', \mathbf{t}_1\theta, \dots, \mathbf{t}_m\theta \rceil\} \cup \Gamma'_1\theta$,
- $C_2 = C_1$,
- $\mu_2 = \mu_1\theta$ with $\theta = \{\bar{y} \mapsto \lceil \bar{x}', \bar{y}' \rceil\}$, and \bar{x}' and \bar{y}' being new variables.

This rule can be shown in a similar way to (W1-U_{prox}). In contrast to (W1-U_{prox}) however, we take $\Phi_2 = \Phi_1$ as there are no new name symbols being introduced and $\theta_2 = [\text{rep}(\theta_1, \{\bar{y} \mapsto \bar{y}\theta_1\}) \rightsquigarrow \{\bar{y} \mapsto \lceil \bar{x}', \mathbf{q}_2, \dots, \mathbf{q}_k \rceil\}]; \theta']$ for $\theta' = \{\bar{x}' \mapsto \mathbf{r}_1\}$, $\Phi_1(\bar{y}\theta_1) = \lceil \mathbf{r}_1, \dots, \mathbf{r}_k \rceil$ and $\bar{y}\theta_1 = \lceil \mathbf{q}_1, \dots, \mathbf{q}_k \rceil$.

□

Lemma 3.15

Let \mathcal{R} be a proximity relation, λ a cut value, σ an (\mathcal{R}, λ) -unifier of two sequences of terms \tilde{s} and \tilde{t} which is non-erasing on \mathcal{X} with $\mathcal{X} = \mathcal{V}(\tilde{s}) \cup \mathcal{V}(\tilde{t})$. Starting with the configuration for their SX-variants $\langle \{\tilde{s} \simeq_{\mathcal{R}, \lambda}^? \tilde{t}\}; \emptyset; \varepsilon \rangle$, there exists a derivation with only basic transformations which ends in $\langle V; C; \mu \rangle$ with V being a variable-only constraint such that for an (\mathcal{R}, λ) -solution ν of V and an (\mathcal{R}, λ) -solution Φ of C , it holds that $(\mu\nu)|_{\mathcal{X}} = \sigma|_{\mathcal{X}}$ for $\mu \in \text{subs}(\Phi(\mu))$.

Proof. Since we know that there exists a solution σ of $\tilde{s} \simeq_{\mathcal{R},\lambda}^? \tilde{t}$, we know from soundness of Pre-Unification that there must exist a derivation which does not end in \perp . Since after applying the rules we can only be left with an variables-only-configuration, we know that the successful derivation ends in $\langle V; C; \mu \rangle$ with V being a variable-only constraint.

Without loss of generality we assume that σ is idempotent and we show first, how we construct the derivation and then show, that the derivation terminates successfully:

We start the derivation with $\langle \Gamma_0; C_0; \mu_0 \rangle$ with $\Gamma_0 = \{\tilde{s} \simeq_{\mathcal{R},\lambda}^? \tilde{t}\}$, $C_0 = \emptyset$ and $\mu_0 = \varepsilon$, and taking $\Phi_0 = \emptyset$ and $\theta_0 = \sigma$, it holds that (Φ_0, θ_0) is an (\mathcal{R}, λ) -solution of $\langle \Gamma_0; C_0; \mu_0 \rangle$ since (Φ_0, θ_0) is a solution of Γ_0 , Φ_0 is a solution of C_0 , for all variables \mathbf{x} it holds that $\mathbf{x}\mu_0\theta_0 = \mathbf{x}\varepsilon\theta_0 = \mathbf{x}\theta_0$ and choosing $\theta_0 = \sigma \in \text{subs}(\Phi_0(\theta_0))$ we get that $\sigma|_{\mathcal{X}} = \theta_0|_{\mathcal{X}}$.

The conditions for the propositions are met and we can apply it n times to reach our final configuration of $\langle V; C_n; \mu_n \rangle$ for which we can construct Φ_n and θ_n such that (Φ_n, θ_n) is a solution of $\langle V; C_n; \mu_n \rangle$, Φ_n solves C_n , $\mathbf{x}\theta_n = \mathbf{x}\mu_n\theta_n$ for all $x \in \text{dom}(\theta_n)$ and $\theta_n|_{\mathcal{X}} = \sigma|_{\mathcal{X}}$ for a substitution $\theta_n \in \text{subs}(\Phi_n(\theta_n))$.

Given an X-substitution \mathbf{v} which is a solution of V , we can represent θ_n as $\theta_n = \mu_n\mathbf{v}$ since $\text{vran}(\mu_n) \subseteq \mathcal{V}(V)$. Using that $\text{subs}(\Phi_n(\theta_n)) = \text{subs}(\Phi_n(\mu_n\mathbf{v}))$, we can select $\mu \in \text{subs}(\Phi_n(\mu_n))$ and $\nu \in \text{subs}(\Phi_n(\mathbf{v}))$ such that $\theta_n = \mu\nu$. Therefore we have $\sigma|_{\mathcal{X}} = \theta_n|_{\mathcal{X}} = \mu\nu|_{\mathcal{X}}$.

We also have to show that the derivation we constructed can actually terminates successfully. In order to do that we need to introduce a directed acyclic graph (dag), on which we will define a complexity measure.

Throughout Pre-Unification we will visually represent the used variables and their connection through a variable dependency graph G , a dag. For each variable in the original unification problem we will have a vertex representing said variable. If for example our problem Γ has the variable set $\mathcal{V}(\Gamma) = \{x, y, z, \bar{x}\}$ then we have $\text{Vert}(G) = \{V_x, V_y, V_z, V_{\bar{x}}\}$. Each of these vertices has a finite set of variables assigned to it which we will call *copies*, in which we will store copies of the variables. At the start of the algorithm we would have $\text{copies}(V_x, G) = \{x\}$, $\text{copies}(V_y, G) = \{y\}$, $\text{copies}(V_z, G) = \{z\}$, $\text{copies}(V_{\bar{x}}, G) = \{\bar{x}\}$ for our example. Since none of the variables are mapped at the beginning, the vertices are not connected and $\text{Edge}(G) = \emptyset$.

Through application of rules edges are added and copy sets are modified.

(IVE-U_{prox}) Given an equation of the form $x \simeq_{\mathcal{R},\lambda}^? \mathbf{t}$, we now have to draw an edge from the vertex to which x belongs to, to the vertices to which the variables in \mathbf{t} belong to. Let $x \in \text{copies}(V_x, G)$ and v_1, \dots, v_n be the variables occurring in \mathbf{t} . Then we add an edge from V_x to V_{v_i} for $v_i \in \text{copies}(V_{v_i}, G)$ for $1 \leq i \leq n$. Since we create a fresh copy \mathbf{t}' of \mathbf{t} , we need to add the copies $\rho(v_1), \dots, \rho(v_n)$ of v_1, \dots, v_n to their respective copy

sets. Since the rule maps x to \mathbf{t}' , we can then delete x from its vertex' copy set, i.e. $\text{copies}(V_x) = \text{copies}(V_x) \setminus \{x\}$.

(SVE- U_{prox}) Given an equation of the form $\lceil \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_m \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \rceil$, the graph G gets modified in the same way as for the individual variable elimination rule but in this case for \bar{x} and \mathbf{t} .

One can see that through these two rules the copy set of the variable to be mapped gets reduced.

We now need an ordering on such graphs. First we can say that we serialize the sequence of vertices in such a way that a vertex comes before those vertices which it is outbound to. We will use this serialization on our set of copy sets, meaning we have a sequence looking like $\lceil \text{copies}(V_1, G), \dots, \text{copies}(V_k, G) \rceil$ with $\lceil V_1, \dots, V_k \rceil$ being the serialization of our vertices. We can say that $G_1 >_{\text{cop}} G_2$ for two graphs G_1 and G_2 , which have the same set of vertices and edges and only the copy sets being different, iff

$$\lceil \text{copies}(V_1, G_1), \dots, \text{copies}(V_k, G_1) \rceil >_{\text{cop}} \lceil \text{copies}(V_1, G_2), \dots, \text{copies}(V_k, G_2) \rceil.$$

The ordering $>_{\text{cop}}$ holds if and only if there exists $1 \leq i \leq k$ such that for all $1 \leq j < i$ it holds that $\text{copies}(V_j, G_1) = \text{copies}(V_j, G_2)$ and $\text{copies}(V_i, G_1) \supset \text{copies}(V_i, G_2)$. This can be viewed as a lexicographic ordering on the copy sets. This ordering on serializations of copy sets is well-founded and therefore the same holds for dags. All the other basic rules do not change the graph.

We can now create our complexity measurement to prove that this derivation terminates. Let $\langle \Gamma'; C'; \gamma \rangle$ be a configuration with respect to an X-substitution δ and a set of variables \mathcal{V} , then the tuple (m_1, m_2, \dots, m_6) is a complexity measure of the configuration with m_1, \dots, m_6 defined as:

$$\begin{aligned}
 m_1 &= \text{Dif}(\mathcal{V}, \delta, \gamma), \\
 m_2 &= \text{the copy set dag}, \\
 m_3 &= \text{the number of symbols in } \Delta, \\
 m_4 &= \{\{\text{size}(\lceil \mathbf{t}_1, \dots, \mathbf{t}_m \rceil) \mid \lceil \mathbf{s}_1, \dots, \mathbf{s}_n \rceil \lesssim_{\mathcal{R}, \lambda}^? \lceil \mathbf{t}_1, \dots, \mathbf{t}_m \rceil \in \Gamma\}\}, \\
 m_5 &= \text{the number of equations in } \Delta \text{ of the form } \mathbf{t} \simeq_{\mathcal{R}, \lambda}^? x, \text{ with } \mathbf{t} \notin \mathcal{V}, \\
 m_6 &= \text{the number of equations in } \Delta \text{ of the form } \lceil \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_m \rceil \simeq_{\mathcal{R}, \lambda}^? \lceil \bar{x}, \mathbf{s}_1, \dots, \mathbf{s}_n \rceil, \\
 &\quad \text{with } \mathbf{t} \notin \mathcal{V}_S.
 \end{aligned}$$

For m_3, m_5, m_6 are natural numbers and therefore, the standard ordering on them is well-founded. For m_4 we will use the well-founded ordering on multi-sets $>_{\text{DM}}$. The ordering $>_{\text{cop}}$ for m_2 is well founded as well. For m_1 we need to show that $\text{Dif}(\mathcal{X}', \mu_i, \sigma) > 0$ for $\mu_i \in \text{subs}(\boldsymbol{\mu}_i)$

and $\mathcal{X}' = \mathcal{X} \cup \mathcal{V}(\mu_i)$ with $0 \leq i \leq n$. This holds because clearly $\text{ran}(\mu_i) \subseteq \mathcal{X}'$. Hence, we can reuse the ordering on natural numbers.

We can now show that each step strictly reduces the complexity measure:

	m_1	m_2	m_3	m_4	m_5	m_6
T-U _{prox}	\geq	\geq	$>$			
O1-U _{prox}	\geq	\geq	\geq	\geq	$>$	
O2-U _{prox}	\geq	\geq	\geq	\geq	\geq	$>$
HR-U _{prox}	\geq	\geq	$>$			
DEC-U _{prox}	\geq	\geq	\geq	$>$		
IVE-U _{prox}	\geq	$>$				
SVS-U _{prox}	\geq	\geq	\geq	$>$		
SVE1-U _{prox}	\geq	\geq	$>$			
SVE2-U _{prox}	\geq	$>$				
W1-U _{prox}	$>$					
W2-U _{prox}	$>$					

This means we have constructed a derivation and shown that it actually terminates. Therefore the lemma holds. \square

Lemma 3.16

Let \mathcal{R} be a proximity relation, λ a cut value, σ an (\mathcal{R}, λ) -unifier of two sequences of terms \tilde{s} and \tilde{t} , and $\mathcal{X} = \mathcal{V}(\tilde{s}) \cup \mathcal{V}(\tilde{t})$. If σ is erasing on \mathcal{X} , then there exists a derivation $\langle \{\tilde{\mathbf{s}} \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{t}}\}; \emptyset; \varepsilon \rangle \Longrightarrow_P \langle \Gamma_1; C_1; \mu_1 \rangle \Longrightarrow_{BT}^+ \langle \Gamma_k; C; \boldsymbol{\mu} \rangle$ with Γ_k being a variables-only constraint; such that for an (\mathcal{R}, λ) -solution ν of Γ_k and an (\mathcal{R}, λ) -solution Φ of C , it holds that $(\mu\nu)|_{\mathcal{X}} = \sigma|_{\mathcal{X}}$ for $\mu \in \text{subs}(\Phi(\boldsymbol{\mu}))$.

Proof. Let $\mu_1 = \{\bar{x}_1 \mapsto \ulcorner, \dots, \bar{x}_n \mapsto \ulcorner\}$ with $\bar{x}_1, \dots, \bar{x}_n \in \mathcal{X}$ being the variables that are mapped to the empty sequence in σ , for $n > 0$. For the first step of the derivation we will use the projection rule (P-U_{prox}) and get $\langle \{\tilde{\mathbf{s}} \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{t}}\}; \emptyset; \varepsilon \rangle \Longrightarrow_P \langle \Gamma_1; C_1; \mu_1 \rangle$ with $\Gamma_1 = \{\tilde{\mathbf{s}} \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{t}}\} \mu_1$ and $C_1 = \emptyset$. We create a substitution $\varphi = \sigma|_{\text{dom}(\sigma) \setminus \text{dom}(\mu_1)}$, such that $\sigma = \mu_1 \varphi = \mu_1 \mu_1 \varphi$. Since σ is an (\mathcal{R}, λ) -unifier of $\{\tilde{\mathbf{s}} \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{t}}\}$, meaning $\mathcal{R}(\tilde{\mathbf{s}}\sigma, \tilde{\mathbf{t}}\sigma) = \mathcal{R}(\tilde{\mathbf{s}}\mu_1\mu_1\varphi, \tilde{\mathbf{t}}\mu_1\mu_1\varphi) \geq \lambda$, we can see that it also holds that $\mu_1\varphi = \sigma$ is an (\mathcal{R}, λ) -unifier of $\{\tilde{\mathbf{s}}\mu_1 \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{t}}\mu_1\} = \Gamma_1$.

Now σ is non-erasing on $\mathcal{X}_1 = \mathcal{V}(\Gamma_1)$ and an (\mathcal{R}, λ) -unifier of Γ_1 . Therefore, by Lemma 3.15, we know that there exists a derivation $\langle \Gamma'_1; \emptyset; \varepsilon \rangle \Longrightarrow_{BT}^+ \langle \Gamma'_k; C'_k; \boldsymbol{\mu}'_k \rangle$ with $\Gamma'_1 = \Gamma_1$, such that for an (\mathcal{R}, λ) -solution ν of the variables-only configuration Γ'_k and an (\mathcal{R}, λ) -solution Φ of C'_k , it holds that $(\mu'\nu)|_{\mathcal{X}_1} = \sigma|_{\mathcal{X}_1}$ for $\mu' \in \text{subs}(\Phi(\boldsymbol{\mu}'_k))$. Since $\text{vran}(\mu') \subseteq \mathcal{X}_1$, we know that $\mu_1\mu' = \mu_1 \cup \mu'$ and due to ν being non-erasing on \mathcal{X} we get that $\mu_1\mu'\nu|_{\mathcal{X}} = \sigma|_{\mathcal{X}}$.

Assuming we have the step $\langle \Gamma'_i; C'_i; \mu'_i \rangle \Longrightarrow_R \langle \Gamma'_{i+1}; C'_{i+1}; \mu'_{i+1} \rangle$ for $i \geq 1$ and R being a basic transformation rule of the Pre-Unification rule set, then we can achieve the step $\langle \Gamma'_i; C'_i; \mu_1 \mu'_i \rangle \Longrightarrow_R \langle \Gamma'_{i+1}; C'_{i+1}; \mu_1 \mu'_{i+1} \rangle$ with the same rule R . Thus we can simply take $\Gamma_i = \Gamma'_i$, $C_i = C'_i$ and $\mu_i = \mu_1 \mu'_i$ for all $1 \leq i \leq k$, so that we get $\langle \{\tilde{\mathbf{s}} \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{t}}\}; \emptyset; \varepsilon \rangle \Longrightarrow_P \langle \Gamma_1; C_1; \mu_1 \rangle \Longrightarrow_{\text{BT}}^+ \langle \Gamma_k; C_k; \mu_k \rangle$. Then it holds that $(\mu\nu)\mu_1|_{\mathcal{X}} = \sigma|_{\mathcal{X}}$ for $\mu \in \text{subs}(\Phi(\mu_k))$. \square

Theorem 3.17 (Completeness of Pre-Unification)

Let \mathcal{R} be a proximity relation, λ a cut value, σ an (\mathcal{R}, λ) -unifier of two sequences of terms $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$, and $\mathcal{X} = \mathcal{V}(\tilde{\mathbf{s}}) \cup \mathcal{V}(\tilde{\mathbf{t}})$. Let $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$ be SX -versions of $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$. Then there exists a derivation $\langle \{\tilde{\mathbf{s}} \simeq_{\mathcal{R}, \lambda}^? \tilde{\mathbf{t}}\}; \emptyset; \varepsilon \rangle \Longrightarrow^+ \langle V; C; \mu \rangle$ with V being a variables-only constraint such that for an (\mathcal{R}, λ) -solution ν of V and an (\mathcal{R}, λ) -solution Φ of C , it holds that $(\mu\nu)|_{\mathcal{X}} = \sigma|_{\mathcal{X}}$ for $\mu \in \text{subs}(\Phi(\mu))$.

Proof. Follows from Lemma 3.15 and Lemma 3.16. \square

The way we combine two name-neighborhood mappings Φ and Ψ is defined as follows:

$$\begin{aligned} \Phi \odot \Psi := & \{ \mathbf{f} \mapsto \Phi(\mathbf{f}) \mid \mathbf{f} \in \text{dom}(\Phi) \setminus \text{dom}(\Psi) \} \cup \\ & \{ \mathbf{f} \mapsto \Psi(\mathbf{f}) \mid \mathbf{f} \in \text{dom}(\Psi) \setminus \text{dom}(\Phi) \} \cup \\ & \{ \mathbf{f} \mapsto \Phi(\mathbf{f}) \cap \Psi(\mathbf{f}) \mid \mathbf{f} \in \text{dom}(\Phi) \cap \text{dom}(\Psi) \}. \end{aligned}$$

In order to solve the neighborhood constraint C we constructed in the Pre-Unification algorithm, we will use the *Constraint Simplification* algorithm CS from as intended in [35]. As all the necessary changes to accommodate sequence variables and flexible arity were already done within the Pre-Unification algorithm $\mathfrak{U}_{\text{prox}}$, we can now leave the Constraint Simplification as is. CS works on configurations of the form $\langle C; \Phi \rangle$, where C is an (\mathcal{R}, λ) -neighborhood constraint and Φ is a name-neighborhood mapping. The algorithm needs a starting configuration of the form $\langle C; \emptyset \rangle$ and returns either \perp in the case that process fails, or $\langle \{ \mathbf{n}_1 \simeq_{\mathcal{R}, \lambda}^? \mathbf{m}_1, \dots, \mathbf{n}_n \simeq_{\mathcal{R}, \lambda}^? \mathbf{m}_n \}; \Phi \rangle$ with Φ being a solution to C and $\mathbf{n}_i, \mathbf{m}_i \notin \text{dom}(\Phi)$ for $1 \leq i \leq n$.

Constraint Simplification Algorithm (CS)

FFS- $\mathfrak{U}_{\text{prox}}$: **Function Symbols**

$$\begin{aligned} \langle \{ f \approx_{\mathcal{R}, \lambda}^? g \} \cup C; \Phi \rangle & \Longrightarrow \langle C; \Phi \rangle, \\ \text{if } \mathcal{R}(f, g) & \geq \lambda. \end{aligned}$$

NFS- $\mathfrak{U}_{\text{prox}}$: **Name vs. Function Symbol**

$$\langle \{ \mathbf{n} \approx_{\mathcal{R}, \lambda}^? g \} \cup C; \Phi \rangle \Longrightarrow \langle C; \Phi \odot \{ \mathbf{n} \mapsto \text{pc}_{\mathcal{R}, \lambda}(g) \} \rangle.$$

FSN- U_{prox} : **Function Symbol vs. Name**

$$\langle \{g \approx_{\mathcal{R},\lambda}^? \mathbf{n}\} \cup C; \Phi \rangle \Longrightarrow \langle \{\mathbf{n} \approx_{\mathcal{R},\lambda}^? g\} \cup C; \Phi \rangle.$$

NN1- U_{prox} : **Name vs. Name 1**

$$\langle \{\mathbf{n} \approx_{\mathcal{R},\lambda}^? \mathbf{m}\} \cup C; \Phi \rangle \Longrightarrow \langle C; \Phi \odot \{\mathbf{n} \mapsto \{f\}, \mathbf{m} \mapsto \mathbf{pc}_{\mathcal{R},\lambda}(f)\} \rangle,$$

where $\mathbf{n} \in \text{dom}(\Phi)$ and $f \in \Phi(\mathbf{n})$.

NN2- U_{prox} : **Name vs. Name 2**

$$\langle \{\mathbf{m} \approx_{\mathcal{R},\lambda}^? \mathbf{n}\} \cup C; \Phi \rangle \Longrightarrow \langle \{\mathbf{n} \approx_{\mathcal{R},\lambda}^? \mathbf{m}\} \cup C; \Phi \rangle,$$

where $\mathbf{m} \notin \text{dom}(\Phi)$ and $\mathbf{n} \in \text{dom}(\Phi)$.

F1- U_{prox} : **Fail 1**

$$\langle \{f \approx_{\mathcal{R},\lambda}^? g\} \cup C; \Phi \rangle \Longrightarrow \perp,$$

if $\mathcal{R}(f, g) < \lambda$.

F2- U_{prox} : **Fail 2**

$$\langle C; \Phi \rangle \Longrightarrow \perp,$$

if there exists $\mathbf{f} \in \text{dom}(\Phi)$ with $\Phi(\mathbf{f}) = \emptyset$.

Branching can occur when applying the rule (NN1) as all elements of $\Phi(\mathbf{f})$ are a possible function symbol. The number of branches is equal to $|\Phi(\mathbf{f})|$.

Since we did not change the *CS* algorithm from [35] we will omit the proof for termination, soundness and completeness as they would not change from the original.

Example 3.4. We will revisit example 3.1 shown for $\mathfrak{U}_{\text{sim}}$ and change the similarity relation to a proximity relation. Let $\Gamma = \{f(\bar{x}), g(a), x, \bar{y}\} \simeq_{\mathcal{R},\lambda}^? g(f(x, \bar{x}), b, a, c)$, our proximity relation \mathcal{R} being defined as $\mathcal{R}(f, g) = 0.9$, $\mathcal{R}(a, b) = 0.7$ and $\mathcal{R}(a, c) = 0.9$, and $\lambda = 0.8$. Several branches are generated during the solution process of this problem, but we only show a single successful one.

Pre-Unification:

$$\begin{array}{l}
 \langle \Gamma; \emptyset; \varepsilon \rangle \\
 \xrightarrow{\text{P-}U_{\text{prox}}} \langle \{f(g(a), x, \bar{y}) \simeq_{\mathcal{R},\lambda}^? g(f(x), b, a, c)\}; \emptyset; \{\bar{x} \mapsto \ulcorner \urcorner\} \rangle \\
 \xrightarrow{\text{HR-}U_{\text{prox}}} \langle \{\ulcorner g(a), x, \bar{y} \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner f(x), b, a, c \urcorner\}; \{f \approx_{\mathcal{R},\lambda}^? g\}; \{\bar{x} \mapsto \ulcorner \urcorner\} \rangle \\
 \xrightarrow{\text{DEC-}U_{\text{prox}}} \langle \{g(a) \simeq_{\mathcal{R},\lambda}^? f(x), \ulcorner x, \bar{y} \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner b, a, c \urcorner\}; \{f \approx_{\mathcal{R},\lambda}^? g\}; \{\bar{x} \mapsto \ulcorner \urcorner\} \rangle \\
 \xrightarrow{\text{HR-}U_{\text{prox}}} \langle \{a \simeq_{\mathcal{R},\lambda}^? x, \ulcorner x, \bar{y} \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner b, a, c \urcorner\}; \{f \approx_{\mathcal{R},\lambda}^? g\}; \{\bar{x} \mapsto \ulcorner \urcorner\} \rangle \\
 \xrightarrow{\text{O1-}U_{\text{prox}}} \langle \{x \simeq_{\mathcal{R},\lambda}^? a, \ulcorner x, \bar{y} \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner b, a, c \urcorner\}; \{f \approx_{\mathcal{R},\lambda}^? g\}; \{\bar{x} \mapsto \ulcorner \urcorner\} \rangle \\
 \xrightarrow{\text{IVE-}U_{\text{prox}}} \langle \mathbf{n}_1 \simeq_{\mathcal{R},\lambda}^? a, \ulcorner \mathbf{n}_1, \bar{y} \urcorner \simeq_{\mathcal{R},\lambda}^? \ulcorner b, a, c \urcorner\}; \{f \approx_{\mathcal{R},\lambda}^? g\}; \{\bar{x} \mapsto \ulcorner \urcorner, x \mapsto \mathbf{n}_1\} \rangle
 \end{array}$$

$$\begin{array}{l}
 \xrightarrow{\text{HR-U}_{\text{prox}}} \langle \{\lceil \neg \simeq_{\mathcal{R},\lambda}^? \lceil \neg, \lceil \mathbf{n}_1, \bar{y} \rceil \simeq_{\mathcal{R},\lambda}^? \lceil b, a, c \rceil \rceil; \{f \approx_{\mathcal{R},\lambda}^? g, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a\}; \\
 \{\bar{x} \mapsto \lceil \neg, x \mapsto \mathbf{n}_1\}\rangle \\
 \xrightarrow{\text{T-U}_{\text{prox}}} \langle \lceil \mathbf{n}_1, \bar{y} \rceil \simeq_{\mathcal{R},\lambda}^? \lceil b, a, c \rceil \rceil; \{f \approx_{\mathcal{R},\lambda}^? g, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a\}; \{\bar{x} \mapsto \lceil \neg, x \mapsto \mathbf{n}_1\}\rangle \\
 \xrightarrow{\text{DEC-U}_{\text{prox}}} \langle \{\mathbf{n}_1 \simeq_{\mathcal{R},\lambda}^? b, \bar{y} \simeq_{\mathcal{R},\lambda}^? \lceil a, c \rceil \rceil; \{f \approx_{\mathcal{R},\lambda}^? g, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a\}; \{\bar{x} \mapsto \lceil \neg, x \mapsto \mathbf{n}_1\}\rangle \\
 \xrightarrow{\text{HR/T-U}_{\text{prox}}} \langle \{\bar{y} \simeq_{\mathcal{R},\lambda}^? \lceil a, c \rceil \rceil; \{f \approx_{\mathcal{R},\lambda}^? g, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? b\}; \{\bar{x} \mapsto \lceil \neg, x \mapsto \mathbf{n}_1\}\rangle \\
 \xrightarrow{\text{W1-U}_{\text{prox}}} \langle \{\mathbf{n}_2 \simeq_{\mathcal{R},\lambda}^? a, \bar{y} \simeq_{\mathcal{R},\lambda}^? c\}; \{f \approx_{\mathcal{R},\lambda}^? g, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? b\}; \\
 \{\bar{x} \mapsto \lceil \neg, x \mapsto \mathbf{n}_1, \bar{y} \mapsto \lceil \mathbf{n}_2, \bar{y}' \rceil \rceil\}\rangle \\
 \xrightarrow{\text{HR/T-U}_{\text{prox}}} \langle \{\bar{y}' \simeq_{\mathcal{R},\lambda}^? c\}; \{f \approx_{\mathcal{R},\lambda}^? g, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? b, \mathbf{n}_2 \approx_{\mathcal{R},\lambda}^? a\}; \\
 \{\bar{x} \mapsto \lceil \neg, x \mapsto \mathbf{n}_1, \bar{y} \mapsto \lceil \mathbf{n}_2, \bar{y}' \rceil \rceil\}\rangle \\
 \xrightarrow{\text{SVE2-U}_{\text{prox}}} \langle \{\mathbf{n}_3 \simeq_{\mathcal{R},\lambda}^? c\}; \{f \approx_{\mathcal{R},\lambda}^? g, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? b, \mathbf{n}_2 \approx_{\mathcal{R},\lambda}^? a\}; \\
 \{\bar{x} \mapsto \lceil \neg, x \mapsto \mathbf{n}_1, \bar{y} \mapsto \lceil \mathbf{n}_2, \mathbf{n}_3 \rceil \rceil\}\rangle \\
 \xrightarrow{\text{HR/T-U}_{\text{prox}}} \langle \emptyset; \{f \approx_{\mathcal{R},\lambda}^? g, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? b, \mathbf{n}_2 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_3 \approx_{\mathcal{R},\lambda}^? c\}; \\
 \{\bar{x} \mapsto \lceil \neg, x \mapsto \mathbf{n}_1, \bar{y} \mapsto \lceil \mathbf{n}_2, \mathbf{n}_3 \rceil \rceil\}\rangle.
 \end{array}$$

Next we take our computed constraint $C = \{f \approx_{\mathcal{R},\lambda}^? g, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? b, \mathbf{n}_2 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_3 \approx_{\mathcal{R},\lambda}^? c\}$ and apply the Constraint Simplification algorithm:

Constraint Simplification:

$$\begin{array}{l}
 \langle \{f \approx_{\mathcal{R},\lambda}^? g, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? b, \mathbf{n}_2 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_3 \approx_{\mathcal{R},\lambda}^? c\}; \emptyset \rangle \\
 \xrightarrow{\text{FFS}} \langle \{\mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? b, \mathbf{n}_2 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_3 \approx_{\mathcal{R},\lambda}^? c\}; \emptyset \rangle \\
 \xrightarrow{\text{NFS}} \langle \{\mathbf{n}_1 \approx_{\mathcal{R},\lambda}^? b, \mathbf{n}_2 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_3 \approx_{\mathcal{R},\lambda}^? c\}; \{\mathbf{n}_1 \mapsto \{b, c\}\}\rangle \\
 \xrightarrow{\text{NFS}} \langle \{\mathbf{n}_2 \approx_{\mathcal{R},\lambda}^? a, \mathbf{n}_3 \approx_{\mathcal{R},\lambda}^? c\}; \{\mathbf{n}_1 \mapsto \{b\}\}\rangle \\
 \xrightarrow{\text{NFS}} \langle \{\mathbf{n}_3 \approx_{\mathcal{R},\lambda}^? c\}; \{\mathbf{n}_1 \mapsto \{b\}, \mathbf{n}_2 \mapsto \{b, c\}\}\rangle \\
 \xrightarrow{\text{NFS}} \langle \emptyset; \{\mathbf{n}_1 \mapsto \{b\}, \mathbf{n}_2 \mapsto \{b, c\}, \mathbf{n}_3 \mapsto \{a, c\}\}\rangle.
 \end{array}$$

Therefore, we get $\Phi = \{\mathbf{n}_1 \mapsto \{b\}, \mathbf{n}_2 \mapsto \{b, c\}, \mathbf{n}_3 \mapsto \{a, c\}\}$ and $\mu = \{\bar{x} \mapsto \lceil \neg, x \mapsto \mathbf{n}_1, \bar{y} \mapsto \lceil \mathbf{n}_2, \mathbf{n}_3 \rceil \rceil\}$ which gives us $\Phi(\mu) = \{\bar{x} \mapsto \lceil \neg, x \mapsto \{b\}, \bar{y} \mapsto \lceil \{b, c\}, \{a, c\} \rceil \rceil\}$. Computed solutions therefore are

$$\begin{aligned}
 \text{singl}(\Phi(\mu)) = & \{\{\bar{x} \mapsto \lceil \neg, x \mapsto b, \bar{y} \mapsto \lceil b, a \rceil \rceil, \\
 & \{\bar{x} \mapsto \lceil \neg, x \mapsto b, \bar{y} \mapsto \lceil b, c \rceil \rceil, \\
 & \{\bar{x} \mapsto \lceil \neg, x \mapsto b, \bar{y} \mapsto \lceil c, a \rceil \rceil, \\
 & \{\bar{x} \mapsto \lceil \neg, x \mapsto b, \bar{y} \mapsto \lceil c, c \rceil \rceil\}.
 \end{aligned}$$

4 Matching

This chapter is based on the procedure to solve matching problems with free function symbols from *Variadic Equational Matching in Associative and Commutative Theories* [10] by Besik Dundua, Temur Kutsia and Mircea Marin. Said procedure will be modified in such a way that it can work with similarity and proximity relations instead of syntactic equality. However, we do not consider the associative and commutative equational theories.

In [10], matching problems are first linearized to get rid of multiple occurrences of the same variable. This is done by introducing variable copies, renaming each occurrence of a variable by a new copy. After that, the *Linear Matching* procedure $L\mathfrak{M}$ is applied to find a preliminary solution for the linearized matching problem. In the next step, in this preliminary solution, the variables are reverted to their original form, followed by using the *Reconstruct Solutions* or *RS* algorithm, designed to check for consistency in the non-linear case, to make sure that the copies of the same variable are mapped to the same expression (a term or a sequence of terms).

We keep this method of handling linear and non-linear matching problems and change the individual rules and steps in $L\mathfrak{M}$ and *RS* such that they can deal with similarity and proximity relations. Below the set of all matchers for the matching problem Γ is denoted by $\mathcal{M}(\Gamma)$.

The original $L\mathfrak{M}$ procedure has the following set of inference rules for syntactic matching of a linear problem with free function symbols, written in our notation:

T-M: **Trivial**

$$\langle \{s \ll^? s\} \cup \Gamma; S \rangle \Longrightarrow \langle \Gamma; S \rangle.$$

IVE-M: **Individual Variable Elimination**

$$\langle \{x \ll^? t\} \cup \Gamma; S \rangle \Longrightarrow \langle \Gamma; S \cup \{x \approx t\} \rangle.$$

DEC-M: **Decomposition**

$$\langle \{f(s, \tilde{s}) \ll^? f(t, \tilde{t})\} \cup \Gamma; S \rangle \Longrightarrow \langle \{s \ll^? t, f(\tilde{s}) \ll^? f(\tilde{t})\} \cup \Gamma; S \rangle,$$

where $s \notin \mathcal{V}_S$.

SVE-M: **Sequence Variable Elimination**

$$\langle \{f(\bar{x}, \tilde{s}) \ll^? f(\tilde{t}_1, \tilde{t}_2)\} \cup \Gamma; S \rangle \Longrightarrow \langle \{f(\tilde{s}) \ll^? f(\tilde{t}_2)\} \cup \Gamma; S \cup \{\bar{x} \approx \tilde{t}_1\} \rangle.$$

This procedure works on pairs $\langle \Gamma; S \rangle$, where Γ is a linear matching problem and S is a set of equations in solved form. The algorithm starts with the initial system of $\langle \Gamma; \emptyset \rangle$ and by applying these rules it either ends in a successful configuration $\langle \emptyset; S \rangle$, with $\Sigma(S) \in \mathcal{M}(\Gamma)$ (where $\Sigma(S)$ is a substitution obtained from a solved form by replacing \approx by \mapsto), or it ends in failure when no further rule can be applied.

The original algorithm *RS* has the following steps to check a preliminary solution S of a linearized matching problem for consistency:

Algorithm 1 RS - Reconstruct Solutions

Input: S_{lin}

Output: S

- 1: $S := S_{\text{lin}}$
 - 2: Reinststate the original names of all individual and sequence variables in S .
 Example: $S = \{v_x^1 \approx t_1, v_x^2 \approx t_2\}$ will be changed to $S = \{x \approx t_1, x \approx t_2\}$.
 If $t_1 = t_2$, then $S = \{x \approx t_1\}$ because S is a set.
 - 3: If S contains two pairs $x \approx t_1$ and $x \approx t_2$ for the same individual variable $x \in \mathcal{V}_I$ with $t_1 \neq t_2$, then STOP with failure.
 - 4: If S contains two pairs $\bar{x} \approx \tilde{t}_1$ and $\bar{x} \approx \tilde{t}_2$ for the same sequence variable $\bar{x} \in \mathcal{V}_S$ with $\tilde{t}_1 \neq \tilde{t}_2$, then STOP with failure.
 - 5: **return** S .
-

It returns a final solution S for the non-linear matching problem Γ with free function symbols. From S , a matcher for Γ can then be extracted as mentioned above: $\Sigma(S) \in \mathcal{M}(\Gamma)$.

4.1 Fuzzy Unranked Matching with Similarity Relations

In order for the unranked linear matching algorithm *L \mathfrak{M}* to work with similarity relations, we need to modify the rules slightly. The way this change is done is influenced by Maria Sessa's [49] and Temur Kutsia's and Cleo Pau's [44] approaches. In contrast to the original *L \mathfrak{M}* , we will now allow the algorithm to work on sequences of terms and therefore add a rule which is responsible for removing the heads of the terms.

T- M_{sim} : **Trivial**

$$\langle \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{s}\} \cup \Gamma; S; \alpha \rangle \Longrightarrow \langle \Gamma; S; \alpha \rangle.$$

IVE- M_{sim} : **Individual Variable Elimination**

$$\langle \{x \lesssim_{\mathcal{R}, \lambda}^? t\} \cup \Gamma; S; \alpha \rangle \Longrightarrow \langle \Gamma; S \cup \{x \approx_{\mathcal{R}, \lambda} t\}; \alpha \rangle.$$

HR- M_{sim} : **Head Removal**

$$\langle \{f(\tilde{s}) \lesssim_{\mathcal{R}, \lambda}^? g(\tilde{t})\} \cup \Gamma; S; \alpha \rangle \Longrightarrow \langle \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma; S; \alpha \wedge \beta \rangle,$$

if $\mathcal{R}(f, g) = \beta \geq \lambda$.

DEC-M_{sim}: Decomposition

$$\langle \{\ulcorner s, \tilde{s} \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner t, \tilde{t} \urcorner\} \cup \Gamma; S; \alpha \rangle \Longrightarrow \langle \{s \lesssim_{\mathcal{R}, \lambda}^? t, \tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma; S; \alpha \wedge \beta \rangle,$$

where $s \notin \mathcal{V}_S$.

SVE-M_{sim}: Sequence Variable Elimination

$$\langle \{\ulcorner \bar{x}, \tilde{s} \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner \tilde{t}_1, \tilde{t}_2 \urcorner\} \cup \Gamma; S; \alpha \rangle \Longrightarrow \langle \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}_2\} \cup \Gamma; S \cup \{\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_1\}; \alpha \rangle.$$

We also add additional rules, to explicitly catch failure cases, as opposed to the original procedure in [10], where failure was defined as the inability to apply any further rules to the system of matching problems.

CLA-M_{sim}: Clash

$$\langle \{f(\tilde{s}) \lesssim_{\mathcal{R}, \lambda}^? g(\tilde{t})\} \cup \Gamma; S; \alpha \rangle \Longrightarrow \perp,$$

if $\mathcal{R}(f, g) < \lambda$.

AD-M_{sim}: Arity Disagreement

$$\langle \{\ulcorner s_1, \dots, s_n \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner t_1, \dots, t_m \urcorner\} \cup \Gamma; S; \alpha \rangle \Longrightarrow \perp,$$

if $n \neq m$, $s_i \notin \mathcal{V}_S$ for all $1 \leq i \leq n$.

E-M_{sim}: Empty

$$\langle \{\ulcorner s, s_1, \dots, s_n \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner \urcorner\} \cup \Gamma; S; \alpha \rangle \Longrightarrow \perp,$$

with $s \notin \mathcal{V}_S$.

The fuzzy *Linear Matching* algorithm $L\mathfrak{M}_{\text{sim}}$ with similarity relations \mathcal{R} and cut values λ , works on a system $\langle \Gamma; S; \alpha \rangle$, where Γ is a linear (\mathcal{R}, λ) -matching problem, S is a set of equations in solved form, and α is the matching degree. The above-listed rules are applied to selected equations of the matching problem to form a derivation. Branching can occur as rule (SVE-M_{sim}) has multiple ways of how it can be set up, depending on how many terms are assigned to the sequence variable.

The initial system or starting configuration will be $\langle \Gamma; \emptyset; 1 \rangle$ and, after applying the rules, either the final configuration of $\langle \emptyset; S; \alpha \rangle$ indicating a successful derivation will be reached, or the process ends in \perp through one of the failure rules meaning that the matching problem Γ cannot be solved.

In the case where the derivation goes from $\langle \Gamma; \emptyset; 1 \rangle \Longrightarrow^+ \langle \emptyset; S; \alpha \rangle$, S is called an (\mathcal{R}, λ) -*answer* of Γ and α is the accompanying matching degree. The set of all answers of Γ under consideration of λ -cut and \mathcal{R} , computed by the algorithm $L\mathfrak{M}_{\text{sim}}$, will be written as $L\mathfrak{M}_{\text{sim}}(\Gamma, \mathcal{R}, \lambda)$.

Definition 4.1

Let \mathcal{R} be a similarity relation and λ a cut value for \mathcal{R} . A substitution σ is called an (\mathcal{R}, λ) -matcher of a (\mathcal{R}, λ) -matching problem Γ if $\deg_{\mathcal{R}}(\Gamma\sigma) \geq \lambda$. The set of all (\mathcal{R}, λ) -matchers of Γ is denoted by $\mathcal{M}(\Gamma, \mathcal{R}, \lambda)$.

If we know that $\text{deg}_{\mathcal{R}}(\Gamma\sigma) = \alpha \geq \lambda$ we can also write $\sigma \in \mathcal{M}(\Gamma, \mathcal{R}, \lambda)_{\alpha}$.

In the case $\Sigma(S) \in \mathcal{M}(\Gamma, \mathcal{R}, \lambda)_{\alpha}$, we can also write that $S \in L\mathfrak{M}_{\text{sim}}(\Gamma, \mathcal{R}, \lambda)_{\alpha}$.

The following proofs regarding $L\mathfrak{M}_{\text{sim}}$ are based on the proofs of the original procedure for syntactic unranked linear matching from [10] and are modified to handle the changes for solving with similarity relations.

Theorem 4.1 (Termination of $L\mathfrak{M}_{\text{sim}}$)

The algorithm $L\mathfrak{M}_{\text{sim}}$ terminates for every input.

Proof. In order to show that the procedure terminates, we create a complexity measure and show that it strictly decreases with each step performed in $L\mathfrak{M}_{\text{sim}}$. We define the complexity measure $\langle m_1, m_2 \rangle$ as follows:

$$m_1 = \text{number of variables in } \Gamma,$$

$$m_2 = \{\{\text{size}(\tilde{t}) \mid \tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t} \in \Gamma\}\}.$$

The complexity measure is ordered by the lexicographic ordering on tuples of integers and is well founded with ordering on both natural numbers and multisets of natural numbers. We can see that each of the non-failing rules of $L\mathfrak{M}_{\text{sim}}$ reduces our measure strictly:

	m_1	m_2
T- $\mathfrak{M}_{\text{sim}}$:	\geq	$>$
HR- $\mathfrak{M}_{\text{sim}}$:	\geq	$>$
DEC- $\mathfrak{M}_{\text{sim}}$:	\geq	$>$
IVE- $\mathfrak{M}_{\text{sim}}$:	$>$	
SVE- $\mathfrak{M}_{\text{sim}}$:	$>$	

The rules (CLA- $\mathfrak{M}_{\text{sim}}$), (AD- $\mathfrak{M}_{\text{prox}}$) and (E- $\mathfrak{M}_{\text{sim}}$) clearly also lead to termination. Therefore $L\mathfrak{M}_{\text{sim}}$ terminates for every input. \square

We also need one further property:

Lemma 4.2

Given a similarity relation \mathcal{R} , a λ -cut and an (\mathcal{R}, λ) -matching problem Γ , if $\mathcal{M}(\Gamma_1, \mathcal{R}, \lambda)_{\alpha} = \mathcal{M}(\Gamma_2, \mathcal{R}, \lambda)_{\alpha}$, then $\mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)_{\alpha} = \mathcal{M}(\Gamma_2\sigma, \mathcal{R}, \lambda)_{\alpha}$ for any substitution σ .

Proof. The proof follows along the same line as for Lemma 3.1. \square

Lemma 4.3

Given a similarity relation \mathcal{R} , a cut value λ and a linear matching problem Γ_1 , if $\langle \Gamma_1; S_1; \alpha_1 \rangle \Longrightarrow \langle \Gamma_2; S_2; \alpha_2 \rangle$ is a step in $L\mathcal{M}_{sim}$ with $S_2 = S_1 \cup S$ and $\alpha_2 = \alpha_1 \wedge \beta$ for some S and β , then $\mathcal{M}(\Gamma_2, \mathcal{R}, \lambda) = \mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)$ with $\sigma = \Sigma(S)$. Additionally, if $\varphi \in \mathcal{M}(\Gamma_2, \mathcal{R}, \lambda)_\gamma$ then $\varphi \in \mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)_{\gamma \wedge \beta}$.

Proof. We only prove this lemma for certain rules. For those not shown, the proof is either similar or simpler.

(IVE- \mathcal{M}_{sim}) We have $\langle \{x \lesssim_{\mathcal{R}, \lambda}^? s\} \cup \Gamma'_1; S_1; \alpha_1 \rangle \Longrightarrow_{IVE-\mathcal{M}_{sim}} \langle \Gamma_2; S_1 \cup S; \alpha_1 \rangle$ with $\Gamma_2 = \Gamma'_1$, $S = \{x \approx_{\mathcal{R}, \lambda} s\}$ and $\beta = 1$.

From S we get $\Sigma(S) = \{x \mapsto s\}$ and therefore we get $\sigma = \{x \mapsto s\}$. We also can see, that $\Gamma_1\sigma = \{x \lesssim_{\mathcal{R}, \lambda}^? s\}\sigma \cup \Gamma'_1\sigma = \{x\sigma \lesssim_{\mathcal{R}, \lambda}^? s\} \cup \Gamma'_1\sigma = \{x\sigma \lesssim_{\mathcal{R}, \lambda}^? s\} \cup \Gamma'_1$, where the last equality stems from the fact that Γ_1 is linear.

“ \subseteq ” let $\varphi \in \mathcal{M}(\Gamma_2, \mathcal{R}, \lambda)_\gamma$, which means that $\deg_{\mathcal{R}}(\Gamma_2\varphi) = \gamma \geq \lambda$.

In order to show that $\varphi \in \mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)_{\gamma \wedge \beta} \subseteq \mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)$ we can show that:

$$\begin{aligned}
 \deg_{\mathcal{R}}(\Gamma_1\sigma\varphi) &= \deg_{\mathcal{R}}(\{x\sigma \lesssim_{\mathcal{R}, \lambda}^? s\}\varphi \cup \Gamma'_1\varphi) \\
 &= \deg_{\mathcal{R}}(\{x\sigma \lesssim_{\mathcal{R}, \lambda}^? s\}\varphi) \wedge \deg_{\mathcal{R}}(\Gamma'_1\varphi) \\
 &= \deg_{\mathcal{R}}(\{s \lesssim_{\mathcal{R}, \lambda}^? s\}\varphi) \wedge \deg_{\mathcal{R}}(\Gamma_2\varphi) \\
 &= 1 \wedge \gamma = \beta \wedge \gamma \geq \lambda.
 \end{aligned}$$

“ \supseteq ” let $\varphi \in \mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)$, i.e. $\deg_{\mathcal{R}}(\Gamma_1\sigma\varphi) \geq \lambda$.

In order to show that $\varphi \in \mathcal{M}(\Gamma_2, \mathcal{R}, \lambda)$, which we can do by proving $\deg_{\mathcal{R}}(\Gamma_2\varphi) \geq \lambda$, we can use our rewrite of $\deg_{\mathcal{R}}(\Gamma_1\sigma\varphi) \geq \lambda$ from the other direction:

$$\begin{aligned}
 \deg_{\mathcal{R}}(\Gamma_1\sigma\varphi) &= \deg_{\mathcal{R}}(\{s \lesssim_{\mathcal{R}, \lambda}^? s\}\varphi) \wedge \deg_{\mathcal{R}}(\Gamma_2\varphi) \\
 &= 1 \wedge \deg_{\mathcal{R}}(\Gamma_2\varphi) \\
 &= \deg_{\mathcal{R}}(\Gamma_2\varphi) \geq \lambda.
 \end{aligned}$$

(SVE- \mathcal{M}_{sim}) We have $\langle \{\lceil \bar{x}, \tilde{s} \rceil \lesssim_{\mathcal{R}, \lambda}^? \lceil \tilde{t}_1, \tilde{t}_2 \rceil\} \cup \Gamma'_1; S_1; \alpha_1 \rangle \Longrightarrow_{SVE-\mathcal{M}_{sim}} \langle \Gamma_2; S_1 \cup S; \alpha_1 \rangle$ with $\Gamma_2 = \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}_2\} \cup \Gamma'_1$, $S = \{\{\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_1\}\}$ and $\beta = 1$.

From S we get $\Sigma(S) = \{\bar{x} \mapsto \tilde{t}_1\}$ and therefore we get $\sigma = \{\bar{x} \mapsto \tilde{t}_1\}$. We also can see that $\Gamma_1\sigma = \{\lceil \bar{x}, \tilde{s} \rceil \lesssim_{\mathcal{R}, \lambda}^? \lceil \tilde{t}_1, \tilde{t}_2 \rceil\}\sigma \cup \Gamma'_1\sigma = \{\lceil \tilde{t}_1, \tilde{s} \rceil \lesssim_{\mathcal{R}, \lambda}^? \lceil \tilde{t}_1, \tilde{t}_2 \rceil\} \cup \Gamma'_1\sigma$.

“ \subseteq ” let $\varphi \in \mathcal{M}(\Gamma_2, \mathcal{R}, \lambda)_\gamma$, which means that $\deg_{\mathcal{R}}(\Gamma_2\varphi) = \gamma \geq \lambda$.

In order to show that $\varphi \in \mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)_{\gamma \wedge \beta} \subseteq \mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)$ we can show that:

$$\begin{aligned}
 \deg_{\mathcal{R}}(\Gamma_1\sigma\varphi) &= \deg_{\mathcal{R}}(\{\ulcorner \tilde{t}_1, \tilde{s}^\top \lesssim_{\mathcal{R}, \lambda}^? \ulcorner \tilde{t}_1, \tilde{t}_2^\top \urcorner \varphi \cup \Gamma'_1\varphi\}) \\
 &= \deg_{\mathcal{R}}(\{\ulcorner \tilde{t}_1, \tilde{s}^\top \lesssim_{\mathcal{R}, \lambda}^? \ulcorner \tilde{t}_1, \tilde{t}_2^\top \urcorner \varphi\}) \wedge \deg_{\mathcal{R}}(\Gamma'_1\varphi) \\
 &= \mathcal{R}(\tilde{t}_1\varphi, \tilde{t}_1\varphi) \wedge \mathcal{R}(\tilde{s}\varphi, \tilde{t}_2\varphi) \wedge \deg_{\mathcal{R}}(\Gamma'_1\varphi) \\
 &= 1 \wedge \deg_{\mathcal{R}}(\{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}_2\}\varphi) \wedge \deg_{\mathcal{R}}(\Gamma'_1\varphi) \\
 &= 1 \wedge \deg_{\mathcal{R}}(\{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}_2\}\varphi \cup \Gamma'_1\varphi) \\
 &= 1 \wedge \deg_{\mathcal{R}}(\Gamma_2\varphi) \\
 &= \beta \wedge \gamma \geq \lambda.
 \end{aligned}$$

“ \supseteq ” works analogously.

□

Lemma 4.4

Given a similarity relation \mathcal{R} , a λ -cut and a linear (\mathcal{R}, λ) -matching problem Γ_1 , if

$$\langle \Gamma_1; S_1; \alpha_1 \rangle \Longrightarrow^+ \langle \Gamma_n; S_n; \alpha_n \rangle$$

is a derivation in $L\mathfrak{M}_{sim}$ with $S_n = S_1 \cup S$ and $\alpha_n = \alpha_1 \wedge \beta$, then $\mathcal{M}(\Gamma_n, \mathcal{R}, \lambda) = \mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)$ with $\sigma = \Sigma(S)$. Additionally, if $\varphi \in \mathcal{M}(\Gamma_n, \mathcal{R}, \lambda)_\gamma$ then $\varphi \in \mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)_{\gamma \wedge \beta}$.

Proof. In order to prove this lemma, we will use Lemma 4.3 and induction on the number of steps n . When $n = 1$, the statement holds due to Lemma 4.3. Let us assume as induction hypothesis that it holds for $n - 1$ steps.

Let $\langle \Gamma_1; S_1\alpha_1 \rangle \Longrightarrow \langle \Gamma_2; S_1 \cup S^1; \alpha_1 \wedge \beta_1 \rangle \Longrightarrow \dots \Longrightarrow \langle \Gamma_n; S_1 \cup S^1 \cup \dots \cup S^{n-1}; \alpha_1 \wedge \beta_1 \wedge \dots \wedge \beta_{n-1} \rangle \Longrightarrow \langle \Gamma_{n+1}; S_1 \cup S^1 \cup \dots \cup S^n; \alpha_1 \wedge \beta_1 \wedge \dots \wedge \beta_n \rangle$ be our derivation with $S = S^1 \cup \dots \cup S^n$ and $\beta = \beta_1 \wedge \dots \wedge \beta_n$. We denote by $\sigma_i = \Sigma(S^i)$ the substitution derived from the solved sets for $i = 1, \dots, n$. From our induction hypothesis, we know that $\mathcal{M}(\Gamma_n, \mathcal{R}, \lambda) = \mathcal{M}(\Gamma_1\sigma_1 \dots \sigma_{n-1}, \mathcal{R}, \lambda)$ with $\sigma_1 \dots \sigma_{n-1} = \Sigma(S^1 \cup \dots \cup S^{n-1})$ and Lemma 4.3 tells us that $\mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda) = \mathcal{M}(\Gamma_n\sigma_n, \mathcal{R}, \lambda)$ with $\sigma_n = \Sigma(S^n)$. In order to show that $\mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda) = \mathcal{M}(\Gamma_1\sigma, \mathcal{R}, \lambda)$ with $\sigma = \Sigma(S)$ we will be using Lemma 4.2 and our induction hypothesis.

$$\begin{aligned}
 \mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda) &= \mathcal{M}(\Gamma_n\sigma_n, \mathcal{R}, \lambda) \\
 &\stackrel{L4.2}{=} \mathcal{M}(\Gamma_n\sigma_1 \dots \sigma_{n-1}\sigma_n, \mathcal{R}, \lambda) \\
 &\stackrel{IH}{=} \mathcal{M}(\Gamma_1\sigma_1 \dots \sigma_{n-1}\sigma_n, \mathcal{R}, \lambda)
 \end{aligned}$$

Since $\sigma_1 \cdots \sigma_{n-1} \sigma_n = \Sigma(S) = \Sigma(S^1 \cup \cdots \cup S^n)$, we have proven this statement. We additionally want to show that if $\varphi \in \mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda)_\gamma$, then $\varphi \in \mathcal{M}(\Gamma_1 \sigma, \mathcal{R}, \lambda)_{\gamma \wedge \beta}$ with $\sigma = \Sigma(S)$. From our induction hypothesis we know that

$$\varphi \in \mathcal{M}(\Gamma_n, \mathcal{R}, \lambda)_\gamma \implies \varphi \in \mathcal{M}(\Gamma_1 \sigma_1 \cdots \sigma_{n-1}, \mathcal{R}, \lambda)_{\gamma \wedge \beta_1 \wedge \cdots \wedge \beta_n}$$

with $\sigma_1 \cdots \sigma_{n-1} = \Sigma(S^1 \cup \cdots \cup S^{n-1})$. So we can show that:

$$\begin{aligned} \varphi \in \mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda)_\gamma &\stackrel{\text{L4.3}}{\implies} \varphi \in \mathcal{M}(\Gamma_n \sigma_n, \mathcal{R}, \lambda)_{\gamma \wedge \beta_n} \\ &\stackrel{\text{L4.2, IH}}{\implies} \varphi \in \mathcal{M}(\Gamma_1 \sigma_1 \cdots \sigma_{n-1} \sigma_n, \mathcal{R}, \lambda)_{\gamma \wedge \beta_n \wedge \beta_1 \wedge \cdots \wedge \beta_{n-1}} \\ &\text{iff } \varphi \in \mathcal{M}(\Gamma_1 \sigma_1 \cdots \sigma_{n-1} \sigma_n, \mathcal{R}, \lambda)_{\gamma \wedge \beta} \end{aligned}$$

with $\sigma_1 \cdots \sigma_{n-1} \sigma_n = \Sigma(S) = \Sigma(S^1 \cup \cdots \cup S^n)$. □

Lemma 4.5

Given a similarity relation \mathcal{R} , a λ -cut and a linear (\mathcal{R}, λ) -matching problem Γ , if $\langle \Gamma; \emptyset; 1 \rangle \implies^+ \langle \emptyset; S; \alpha \rangle$ is a derivation in $L\mathfrak{M}_{sim}$, then $\sigma \in \mathcal{M}(\Gamma, \mathcal{R}, \lambda)_\alpha$ with $\sigma = \Sigma(S)$.

Proof. From Lemma 4.4 we know that $\mathcal{M}(\emptyset, \mathcal{R}, \lambda) = \mathcal{M}(\Gamma \sigma, \mathcal{R}, \lambda)$ for $\sigma = \Sigma(S)$. Since $\varepsilon \in \mathcal{M}(\emptyset, \mathcal{R}, \lambda)_1$ with $\varepsilon = \Sigma(\emptyset)$, it follows that $\varepsilon \in \mathcal{M}(\Gamma \sigma, \mathcal{R}, \lambda)_{1 \wedge \alpha} \iff \sigma \in \mathcal{M}(\Gamma, \mathcal{R}, \lambda)_\alpha$. □

Theorem 4.6 (Soundness of $L\mathfrak{M}_{sim}$)

Let \mathcal{R} be a similarity relation, λ a cut value for \mathcal{R} and Γ a linear (\mathcal{R}, λ) -matching problem. Then $\Sigma(L\mathfrak{M}_{sim}(\Gamma, \mathcal{R}, \lambda)) \subseteq \mathcal{M}(\Gamma, \mathcal{R}, \lambda)$.

Proof. By Lemma 4.5 and the procedure $L\mathfrak{M}_{sim}$. □

For the proof of completeness we will use an approach similar to the proof of completeness of the unification procedure in Chapter 3.1 based on [29].

Theorem 4.7 (Completeness of $L\mathfrak{M}_{sim}$)

Let \mathcal{R} be a similarity relation, λ a cut value for \mathcal{R} and Γ a linear (\mathcal{R}, λ) -matching problem. If $\theta \in \mathcal{M}(\Gamma, \mathcal{R}, \lambda)_\gamma$, then $\theta|_{\mathcal{V}(\Gamma)} \in \Sigma(L\mathfrak{M}_{sim}(\Gamma, \mathcal{R}, \lambda)_\alpha)$ with $\alpha \geq \gamma \geq \lambda$.

Proof. In order to prove this theorem, we will construct a derivation of the form $\langle \Gamma; \emptyset; 1 \rangle \implies^+ \langle \emptyset; S; \alpha \rangle$, using $L\mathfrak{M}_{sim}$, resulting in $\theta|_{\mathcal{V}(\Gamma)} = \Sigma(S)$ and $\alpha \geq \gamma$. Assuming we have already constructed our derivation up to Γ_n , meaning $\langle \Gamma_1; S_1; \alpha_1 \rangle \implies^+ \langle \Gamma_n; S_n; \alpha_n \rangle$, with $\Gamma_1 = \Gamma$, $S_1 = \emptyset$ and $\alpha_1 = 1$, such that $\theta \in \mathcal{M}(\Gamma_n, \mathcal{R}, \lambda)$, $\theta|_{\mathcal{V}(S_n)} = \Sigma(S_n)$ and $\alpha_n \geq \gamma$.

We now have to create the next step $\langle \Gamma_n; S_n; \alpha_n \rangle \Longrightarrow \langle \Gamma_{n+1}; S_{n+1}; \alpha_{n+1} \rangle$, such that $\theta \in \mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda)$, $\theta|_{\mathcal{V}(S_{n+1})} = \Sigma(S_{n+1})$ and $\alpha_{n+1} \geq \gamma$. For this, we will look at a pair $s \lesssim_{\mathcal{R}, \lambda}^? t$ from $\Gamma_n = \{s \lesssim_{\mathcal{R}, \lambda}^? t\} \cup \Gamma'_n$. There are several ways to extend the derivation, depending on that pair:

- 1) $s = t$. In this case the derivation is extended by $\langle \Gamma_n; S_n; \alpha_n \rangle \Longrightarrow_{\text{T-M}_{\text{sim}}} \langle \Gamma'_n; S_n; \alpha_n \rangle$ with $\Gamma_{n+1} = \Gamma'_n$, $S_{n+1} = S_n$ and $\alpha_{n+1} = \alpha_n$. It is easy to see that $\theta \in \mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda)$, $\theta|_{\mathcal{V}(\Gamma_{n+1})} = \Sigma(S_{n+1})$ and $\alpha_{n+1} \geq \gamma$, as the only change has been to reduce the matching problem.
- 2) $s \in \mathcal{V}_I$. Let $s = x$, then we extend the derivation by $\langle \Gamma_n; S_n; \alpha_n \rangle \Longrightarrow_{\text{IVE-M}_{\text{sim}}} \langle \Gamma'_n; S_n \cup \{x \approx_{\mathcal{R}, \lambda} t\}; \alpha_n \rangle$ with $\Gamma_{n+1} = \Gamma'_n$, $S_{n+1} = S_n \cup \{x \approx_{\mathcal{R}, \lambda} t\}$ and $\alpha_{n+1} = \alpha_n$. Since $\theta \in \mathcal{M}(\Gamma, \mathcal{R}, \lambda)$ and we only removed a pair from Γ_n , it holds that $\theta \in \mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda)$. Furthermore, $\theta|_{\mathcal{V}(S_{n+1})} = \theta|_{\mathcal{V}(S_n)} \cup \{x \mapsto t\} = \Sigma(S_{n+1})$ and $\alpha_{n+1} = \alpha_n \geq \gamma$. We know that $\alpha_n \geq \gamma$ and, therefore, we just have to argue that $\beta \geq \gamma$ in order for $\alpha_{n+1} \geq \gamma$ to hold.
- 3) $s = f(\tilde{s})$ and $t = g(\tilde{t})$ with $\mathcal{R}(f, g) = \beta \geq \lambda$. The derivation continues with the step $\langle \Gamma_n; S_n; \alpha_n \rangle \Longrightarrow_{\text{HR-M}_{\text{sim}}} \langle \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma'_n; S_n; \alpha_n \wedge \beta \rangle$ with $\Gamma_{n+1} = \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma'_n$, $S_{n+1} = S_n$ and $\alpha_{n+1} = \alpha_n \wedge \beta$. $\theta \in \mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda)$ since we only removed the heads of the terms. Further it holds that $\theta|_{\mathcal{V}(S_{n+1})} = \Sigma(S_{n+1})$ since $S_{n+1} = S_n$. Since θ is a solution of Γ and must therefore solve the problem pair $f(\tilde{s})$ and $g(\tilde{t})$, γ must take $\mathcal{R}(f, g)$ into consideration and therefore $\gamma \leq \beta$.
- 4) $s = \ulcorner s_0, \tilde{s} \urcorner$ and $t = \ulcorner t_0, \tilde{t} \urcorner$ with $s_1 \notin \mathcal{V}_S$. We perform the next step as

$$\langle \Gamma_n; S_n; \alpha_n \rangle \Longrightarrow_{\text{DEC-M}_{\text{sim}}} \langle \{s_0 \lesssim_{\mathcal{R}, \lambda}^? t_0, \tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma'_n; S_n; \alpha_n \wedge \beta \rangle$$

with $\Gamma_{n+1} = \{s_0 \lesssim_{\mathcal{R}, \lambda}^? t_0, \tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma'_n$, $S_{n+1} = S_n$ and $\alpha_{n+1} = \alpha_n$. We have $\theta \in \mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda)$ because Γ_{n+1} and Γ_n have the same matchers as they have the same set of variables ($\mathcal{V}(\Gamma_{n+1}) = \mathcal{V}(\Gamma_n)$) and since $S_{n+1} = S_n$ it is clear that $\theta|_{\mathcal{V}(S_{n+1})} = \Sigma(S_{n+1})$.

- 5) $s = \ulcorner s_0, \tilde{s} \urcorner$ with $s_0 = \bar{x} \in \mathcal{V}_S$ and $t = \ulcorner \tilde{t}_1, \tilde{t}_2 \urcorner$. The derivation will be extended by $\langle \Gamma_n; S_n; \alpha_n \rangle \Longrightarrow_{\text{SVE-M}_{\text{sim}}} \langle \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}_2\} \cup \Gamma'_n; S_n \cup \{\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_1\}; \alpha_n \rangle$ with $\Gamma_{n+1} = \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}_2\} \cup \Gamma'_n$, $S_{n+1} = S_n \cup \{\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_1\}$ and $\alpha_{n+1} = \alpha_n$. Similar to Case 1) it is clear that $\theta \in \mathcal{M}(\Gamma_{n+1}, \mathcal{R}, \lambda)$, $\theta|_{\mathcal{V}(S_{n+1})} = \Sigma(S_{n+1})$ and $\alpha_{n+1} \geq \beta$. It is also clear that $\alpha \geq \gamma \geq \beta$.

We extend this derivation until we reach the final configuration of $\langle \emptyset; S; \alpha \rangle$ for which we know that $\theta|_{\mathcal{V}(S)} = \Sigma(S)$. Since $\mathcal{V}(S) = \mathcal{V}(\Gamma)$ it holds that $\theta|_{\mathcal{V}(\Gamma)} = \Sigma(S)$ and therefore $\theta|_{\mathcal{V}(\Gamma)} \in \Sigma(L\mathcal{M}(\Gamma, \mathcal{R}, \lambda)_{\text{sim}})$. Also $\alpha \geq \gamma \geq \lambda$. \square

To solve a non-linear matching problem Γ under similarity relations, we also have to rewrite the RS algorithm to RS_{sim} . In general, the way a non-linear (\mathcal{R}, λ) -matching problem Γ is

approached is by first transforming Γ into a linear matching problem Γ_{lin} by renaming all n occurrences of variables $x \in \mathcal{V}(\Gamma)$ as v_x^1, \dots, v_x^n . By doing this we are now left with a linear (\mathcal{R}, λ) -matching problem Γ_{lin} , to which we can apply LM_{sim} to obtain the computed answer S_{lin} and its corresponding similarity degree α_{lin} . In order to get an answer S and degree α to our non-linear problem, we have to revert our variables to their original names and check for consistency of the answer. This is done with the *Reconstruct Solutions* algorithm RS_{sim} , which takes S_{lin} and α_{lin} as inputs and either stops with failure or returns S which solves the non-linear (\mathcal{R}, λ) -matching problem Γ .

Algorithm 2 RS_{sim} - Reconstruct Solutions

Input: $S_{\text{lin}}, \alpha_{\text{lin}}$

Output: S, α

- 1: $S := S_{\text{lin}}, \alpha := \alpha_{\text{lin}}$
 - 2: Reinstatement of the original names of all individual and sequence variables in S .
 Example: $S = \{v_x^1 \approx_{\mathcal{R}, \lambda} t_1, v_x^2 \approx_{\mathcal{R}, \lambda} t_2\} \cup S'$ will be changed to
 $S = \{x \approx_{\mathcal{R}, \lambda} t_1, x \approx_{\mathcal{R}, \lambda} t_2\} \cup S'$.
 If $t_1 = t_2$, then $S = \{x \approx_{\mathcal{R}, \lambda} t_1\}$ because S is a set.
 - 3: If S contains two pairs $x \approx_{\mathcal{R}, \lambda} t_1$ and $x \approx_{\mathcal{R}, \lambda} t_2$ for the same individual variable $x \in \mathcal{V}_I$ with $\mathcal{R}(t_1, t_2) < \lambda$, then STOP with failure.
 - 4: If S contains two pairs $x \approx_{\mathcal{R}, \lambda} t_1$ and $x \approx_{\mathcal{R}, \lambda} t_2$ for the same individual variable $x \in \mathcal{V}_I$ with $\mathcal{R}(t_1, t_2) = \beta \geq \lambda$, then set $S := \{x \approx_{\mathcal{R}, \lambda} t_1\} \cup S'$ or $S := \{x \approx_{\mathcal{R}, \lambda} t_2\} \cup S'$ and $\alpha = \alpha \wedge \beta$. Repeat Step 4.
 - 5: If S contains two pairs $\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_1$ and $\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_2$ for the same sequence variable $\bar{x} \in \mathcal{V}_S$ with $\mathcal{R}(\tilde{t}_1, \tilde{t}_2) < \lambda$, then STOP with failure.
 - 6: If S contains two pairs $\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_1$ and $\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_2$ for the same sequence variable $\bar{x} \in \mathcal{V}_S$ with $\mathcal{R}(\tilde{t}_1, \tilde{t}_2) = \beta \geq \lambda$, then set $S := \{\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_1\} \cup S'$ or $S := \{\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_2\} \cup S'$ and $\alpha = \alpha \wedge \beta$. Repeat Step 6.
 - 7: **return** S and α .
-

Theorem 4.8 (Termination of RS_{sim})

The procedure RS_{sim} terminates for every input.

Proof. Step 1 and Step 2 are only executed once, at the beginning of the algorithm and Step 3 and Step 5 terminate immediately. For steps 4 and 6 we choose an alternative non-deterministically and continue the algorithm. These two steps are applied only a finite number of times, as many as there are individual or sequence variables in S . Therefore RS_{sim} terminates. \square

Theorem 4.9 (Soundness of RS_{sim})

Let \mathcal{R} be a similarity relation, λ a cut value, Γ a non-linear matching problem and $S_{\text{lin}} \in$

$L\mathcal{M}_{sim}(\Gamma_{lin}, \mathcal{R}, \lambda)$ an answer of the linearized problem and α_{lin} its accompanying degree. If the application of RS_{sim} to S_{lin} and α_{lin} gives a pair (S, α) , then $\Sigma(S) \in \mathcal{M}(\Gamma, \mathcal{R}, \lambda)_\alpha$.

Proof. From Theorem 4.6 we know that $\Sigma(L\mathcal{M}_{sim}(\Gamma_{lin}, \mathcal{R}, \lambda)) \subseteq \mathcal{M}(\Gamma_{lin}, \mathcal{R}, \lambda)$ and therefore $S_{lin} \in \mathcal{M}(\Gamma_{lin}, \mathcal{R}, \lambda)$. Applying RS_{sim} to (S_{lin}, α_{lin}) only reinstates the original variable names and merges bindings that correspond to the same variable in the non-linear problem. Steps 3 and 5 ensure that the procedure fails whenever two terms assigned to the same variable are not sufficiently similar. Hence, if RS_{sim} succeeds, all merged bindings are pairwise similar to at least degree λ .

In Steps 4 and 6, whenever two bindings $x \approx_{\mathcal{R}, \lambda} t_1$ and $x \approx_{\mathcal{R}, \lambda} t_2$ (or the corresponding sequence-variable pairs) are merged, the algorithm updates α to $\alpha := \alpha_{lin} \wedge \beta$, with $\beta = \mathcal{R}(t_1, t_2) \geq \lambda$.

After all duplicates are resolved, the resulting set S contains at most one binding for each variable. Therefore, $\Sigma(S)$ defines a proper substitution for Γ . Since all merged terms satisfy the similarity constraint with respect to \mathcal{R} and λ , the substitution $\Sigma(S)$ is a valid matcher for Γ with similarity degree α , i.e., $\Sigma(S) \in \mathcal{M}(\Gamma, \mathcal{R}, \lambda)_\alpha$. \square

Theorem 4.10 (Completeness of RS_{sim})

Let \mathcal{R} be a similarity relation, λ a cut value, Γ a non-linear matching problem and $\theta \in \mathcal{M}(\Gamma, \mathcal{R}, \lambda)_\gamma$, where $\gamma \geq \lambda$. Let $S_{lin} \in L\mathcal{M}_{sim}(\Gamma_{lin}, \mathcal{R}, \lambda)$ with degree α_{lin} . Then there exists a derivation performed by RS_{sim} starting from S_{lin} and α_{lin} that computes S and α such that $\Sigma(S) = \theta|_{V(\Gamma)}$ and $\alpha \geq \gamma$.

Proof. Linearizing Γ and θ yields a linear matcher θ_{lin} for Γ_{lin} . From Theorem 4.7 we get that the linear algorithm produces an answer $S_{lin} \in L\mathcal{M}_{sim}(\Gamma_{lin}, \mathcal{R}, \lambda)$ with accompanying degree α_{lin} satisfying $\alpha_{lin} \geq \gamma$, and furthermore that θ_{lin} is an instance of S_{lin} .

After Step 2, every variable is renamed to its original name. This means that S contains, for each original individual variable x and each sequence variable \bar{x} , sets of bindings $\{x \approx_{\mathcal{R}, \lambda} t_1, x \approx_{\mathcal{R}, \lambda} t_2, \dots, x \approx_{\mathcal{R}, \lambda} t_n\}$, and $\{\bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_1, \bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_2, \dots, \bar{x} \approx_{\mathcal{R}, \lambda} \tilde{t}_n\}$, where each t_i is the term and each \tilde{t}_i is the sequence of terms assigned to the corresponding occurrence in the linear solution. Because θ_{lin} is an instance of S_{lin} and θ is its projection, each t_i and each \tilde{t}_i is \mathcal{R} -similar to $x\theta$ and $\bar{x}\theta$ respectively to degree at least γ .

For any pairs t_i, t_j and \tilde{t}_i, \tilde{t}_j arising from occurrences of the same original individual variable x and sequence variable \bar{x} we have $\mathcal{R}(t_i, t_j) \geq \gamma \geq \lambda$ and $\mathcal{R}(\tilde{t}_i, \tilde{t}_j) \geq \gamma \geq \lambda$, from the fact that θ is a solution of degree γ . Therefore the tests in Step 3 and Step 5 are not triggered along the branch consistent with θ .

When RS_{sim} encounters two bindings for the same variable, it can choose either binding (nondeterministically) in Step 3 (analogously for Step 6) and set $\alpha := \alpha \wedge \beta$ where $\beta = \mathcal{R}(t_i, t_j) \geq \gamma$ (or $\beta = \mathcal{R}(\tilde{t}_i, \tilde{t}_j) \geq \gamma$). To reconstruct θ , at each such choice we select the binding that agrees with θ .

Each application of steps 4 and 6 strictly reduces the number of duplicate bindings for some original variable. Since the number of occurrences is finite, after finitely many merges no duplicates remain and the algorithm terminates, returning a set S with at most one binding per original variable and a final degree α .

By the choices in steps 4 and 6, the binding left in S for each individual variable x and sequence variable \bar{x} coincides with $x\theta$ and $\bar{x}\theta$, respectively, to degree at least γ . Therefore, the substitution $\Sigma(S)$ equals θ restricted to the variables of Γ , i.e. $\Sigma(S) = \theta|_{\mathcal{V}(\Gamma)}$. Also, the final computed degree α satisfies $\alpha \geq \gamma$. \square

Example 4.1. Let $\Gamma = \{f(x, g(a, \bar{x}, b, x), h(\bar{x}, \bar{y}), \bar{y}) \lesssim_{\mathcal{R}, \lambda}^? g(b, h(c, b, a), g(a, b), c, a)\}$ with a similarity relation \mathcal{R} : $\mathcal{R}(f, g) = 0.9$, $\mathcal{R}(g, h) = 0.8$, $\mathcal{R}(h, f) = 0.8$, $\mathcal{R}(a, b) = 0.7$, $\mathcal{R}(b, c) = 0.7$, $\mathcal{R}(c, a) = 0.8$ and a cut value $\lambda = 0.6$.

The linearized version of Γ would be

$$\Gamma_{\text{lin}} = \{f(v_x^1, g(a, \bar{y}_x^1, b, v_x^2), h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2) \lesssim_{\mathcal{R}, \lambda}^? g(b, h(c, b, a), g(a, b), c, a)\}.$$

Now we use $L\mathcal{M}_{\text{sim}}$ to find a solution for Γ_{lin} :

$$\begin{aligned}
 & \langle \Gamma_{\text{lin}}; \emptyset; 1 \rangle \\
 \xrightarrow{\text{HR-M}_{\text{sim}}} & \langle \ulcorner v_x^1, g(a, \bar{y}_x^1, b, v_x^2), h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner b, h(c, b, a), g(a, b), c, a \urcorner \rangle; \\
 & \emptyset; 0.9 \rangle \\
 \xrightarrow{\text{DEC-M}_{\text{sim}}} & \langle \{v_x^1 \lesssim_{\mathcal{R}, \lambda}^? b, \ulcorner g(a, \bar{v}_x^1, b, v_x^2), h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner h(c, b, a), g(a, b), c, a \urcorner \rangle; \\
 & \emptyset; 0.9 \rangle \\
 \xrightarrow{\text{IVE-M}_{\text{sim}}} & \langle \ulcorner g(a, \bar{v}_x^1, b, v_x^2), h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner h(c, b, a), g(a, b), c, a \urcorner \rangle; \\
 & \{v_x^1 \lesssim_{\mathcal{R}, \lambda}^? b\}; 0.9 \rangle \\
 \xrightarrow{\text{DEC-M}_{\text{sim}}} & \langle \{g(a, \bar{v}_x^1, b, v_x^2) \lesssim_{\mathcal{R}, \lambda}^? h(c, b, a), \ulcorner h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner g(a, b), c, a \urcorner \rangle; \\
 & \{v_x^1 \lesssim_{\mathcal{R}, \lambda}^? b\}; 0.9 \rangle \\
 \xrightarrow{\text{HR-M}_{\text{sim}}} & \langle \ulcorner a, \bar{v}_x^1, b, v_x^2 \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner c, b, a \urcorner, \ulcorner h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner g(a, b), c, a \urcorner \rangle; \\
 & \{v_x^1 \lesssim_{\mathcal{R}, \lambda}^? b\}; 0.8 \rangle \\
 \xrightarrow{\text{DEC-M}_{\text{sim}}} & \langle \{a \lesssim_{\mathcal{R}, \lambda}^? c, \ulcorner \bar{v}_x^1, b, v_x^2 \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner b, a \urcorner, \ulcorner h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner g(a, b), c, a \urcorner \rangle; \\
 & \{v_x^1 \lesssim_{\mathcal{R}, \lambda}^? b\}; 0.8 \rangle
 \end{aligned}$$

$$\begin{aligned}
 & \xrightarrow{\text{HR-M}_{\text{sim}}} \langle \{\ulcorner \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner \lrcorner, \lrcorner \bar{v}_x^1, b, v_x^2 \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner ba \lrcorner, \lrcorner h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner g(a, b), c, a \lrcorner \lrcorner \}; \\
 & \quad \{v_x^1 \lesssim_{\mathcal{R},\lambda}^? b\}; 0.8 \rangle \\
 & \xrightarrow{\text{T-M}_{\text{sim}}} \langle \{\lrcorner \bar{v}_x^1, b, v_x^2 \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner b, a \lrcorner, \lrcorner h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner g(a, b), c, a \lrcorner \lrcorner \}; \\
 & \quad \{v_x^1 \lesssim_{\mathcal{R},\lambda}^? b\}; 0.8 \rangle \\
 & \xrightarrow{\text{SVE-M}_{\text{sim}}} \langle \{\lrcorner b, v_x^2 \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner b, a \lrcorner, \lrcorner h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner g(a, b), c, a \lrcorner \lrcorner \}; \\
 & \quad \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner \}; 0.8 \rangle \\
 & \xrightarrow{\text{DEC-M}_{\text{sim}}} \langle \{b \lesssim_{\mathcal{R},\lambda}^? b, v_x^2 \lesssim_{\mathcal{R},\lambda}^? a, \lrcorner h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner g(a, b), c, a \lrcorner \lrcorner \}; \\
 & \quad \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner \}; 0.8 \rangle \\
 & \xrightarrow{\text{T-M}_{\text{sim}}} \langle \{v_x^2 \lesssim_{\mathcal{R},\lambda}^? a, \lrcorner h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner g(a, b), c, a \lrcorner \lrcorner \}; \\
 & \quad \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner \}; 0.8 \rangle \\
 & \xrightarrow{\text{IVE-M}_{\text{sim}}} \langle \{\lrcorner h(\bar{v}_x^2, \bar{v}_y^1), \bar{v}_y^2 \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner g(a, b), c, a \lrcorner \lrcorner \}; \\
 & \quad \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, v_x^2 \approx_{\mathcal{R},\lambda} a\}; 0.8 \rangle \\
 & \xrightarrow{\text{DEC-M}_{\text{sim}}} \langle \{h(\bar{v}_x^2, \bar{v}_y^1) \lesssim_{\mathcal{R},\lambda}^? g(a, b), \bar{v}_y^2 \lesssim_{\mathcal{R},\lambda}^? \lrcorner c, a \lrcorner \lrcorner \}; \\
 & \quad \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, v_x^2 \approx_{\mathcal{R},\lambda} a\}; 0.8 \rangle \\
 & \xrightarrow{\text{HR-M}_{\text{sim}}} \langle \{\lrcorner \bar{v}_x^2, \bar{v}_y^1 \lrcorner \lesssim_{\mathcal{R},\lambda}^? \lrcorner a, b \lrcorner, \bar{v}_y^2 \lesssim_{\mathcal{R},\lambda}^? \lrcorner c, a \lrcorner \lrcorner \}; \\
 & \quad \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, v_x^2 \approx_{\mathcal{R},\lambda} a\}; 0.8 \rangle \\
 & \xrightarrow{\text{SVE-M}_{\text{sim}}} \langle \{\bar{v}_y^1 \lesssim_{\mathcal{R},\lambda}^? \lrcorner a, b \lrcorner, \bar{v}_y^2 \lesssim_{\mathcal{R},\lambda}^? \lrcorner c, a \lrcorner \lrcorner \}; \\
 & \quad \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, v_x^2 \approx_{\mathcal{R},\lambda} a, \bar{v}_x^2 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner \}; 0.8 \rangle \\
 & \xrightarrow{\text{SVE-M}_{\text{sim}}} \langle \{\bar{v}_y^2 \lesssim_{\mathcal{R},\lambda}^? \lrcorner c, a \lrcorner \lrcorner \}; \\
 & \quad \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, v_x^2 \approx_{\mathcal{R},\lambda} a, \bar{v}_x^2 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, \bar{v}_y^1 \approx_{\mathcal{R},\lambda} \lrcorner a, b \lrcorner \}; 0.8 \rangle \\
 & \xrightarrow{\text{SVE-M}_{\text{sim}}} \langle \{\emptyset; \\
 & \quad \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, v_x^2 \approx_{\mathcal{R},\lambda} a, \bar{v}_x^2 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, \bar{v}_y^1 \approx_{\mathcal{R},\lambda} \lrcorner a, b \lrcorner, \bar{v}_y^2 \approx_{\mathcal{R},\lambda} \lrcorner c, a \lrcorner \lrcorner \}; \\
 & \quad 0.8 \rangle
 \end{aligned}$$

Note that this set of solved equations $S_{\text{lin}}^1 = \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, v_x^2 \approx_{\mathcal{R},\lambda} a, \bar{v}_x^2 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, \bar{v}_y^1 \approx_{\mathcal{R},\lambda} \lrcorner a, b \lrcorner, \bar{v}_y^2 \approx_{\mathcal{R},\lambda} \lrcorner c, a \lrcorner \lrcorner\}$ is only one of the possible branches splitting at the (SVE-M_{sim}) rules. The other branches, which do not result in failure deliver the following sets of solved equations:

$$\begin{aligned}
 S_{\text{lin}}^2 &= \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, v_x^2 \approx_{\mathcal{R},\lambda} a, \bar{v}_x^2 \approx_{\mathcal{R},\lambda} \lrcorner a \lrcorner, \bar{v}_y^1 \approx_{\mathcal{R},\lambda} \lrcorner b \lrcorner, \bar{v}_y^2 \approx_{\mathcal{R},\lambda} \lrcorner c, a \lrcorner \lrcorner\} \\
 S_{\text{lin}}^3 &= \{v_x^1 \approx_{\mathcal{R},\lambda} b, \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, v_x^2 \approx_{\mathcal{R},\lambda} a, \bar{v}_x^2 \approx_{\mathcal{R},\lambda} \lrcorner a, b \lrcorner, \bar{v}_y^1 \approx_{\mathcal{R},\lambda} \lrcorner \lrcorner, \bar{v}_y^2 \approx_{\mathcal{R},\lambda} \lrcorner c, a \lrcorner \lrcorner\}
 \end{aligned}$$

In order to compute solutions for the non-linear problem Γ we feed these linear solutions into the RS_{sim} algorithm:

Step 1: $S^1 := S_{\text{lin}}^1$, $\alpha = \alpha_{\text{lin}}$

Step 2: $S^1 = \{x \approx_{\mathcal{R},\lambda} b, \bar{x} \approx_{\mathcal{R},\lambda} \ulcorner, x \approx_{\mathcal{R},\lambda} a, \bar{y} \approx_{\mathcal{R},\lambda} \lrcorner a, b \urcorner, \bar{y} \approx_{\mathcal{R},\lambda} \lrcorner c, a \urcorner\}$, $\alpha = 0.8$

Step 4: $S^1 = \{x \approx_{\mathcal{R},\lambda} b, \bar{x} \approx_{\mathcal{R},\lambda} \ulcorner, \bar{y} \approx_{\mathcal{R},\lambda} \lrcorner a, b \urcorner, \bar{y} \approx_{\mathcal{R},\lambda} \lrcorner c, a \urcorner\}$, $\alpha = 0.7$

Step 6: $S^1 = \{x \approx_{\mathcal{R},\lambda} b, \bar{x} \approx_{\mathcal{R},\lambda} \ulcorner, \bar{y} \approx_{\mathcal{R},\lambda} \lrcorner a, b \urcorner\}$, $\alpha = 0.7$

Therefore, one solution for Γ would be $S^1 = \{x \approx_{\mathcal{R},\lambda} b, \bar{x} \approx_{\mathcal{R},\lambda} \ulcorner, \bar{y} \approx_{\mathcal{R},\lambda} \lrcorner a, b \urcorner\}$, with a similarity degree of $\alpha = 0.7$. Through branching at steps 4 and 6 we get also different versions of S^1 , e.g. with $x \approx_{\mathcal{R},\lambda} a$ and $\bar{y} \approx_{\mathcal{R},\lambda} \lrcorner c, a \urcorner$.

Next we input S_{lin}^2 into RS_{sim} :

Step 1: $S^2 := S_{\text{lin}}^2$, $\alpha = \alpha_{\text{lin}}$

Step 2: $S^2 = \{x \approx_{\mathcal{R},\lambda} b, \bar{x} \approx_{\mathcal{R},\lambda} \ulcorner, x \approx_{\mathcal{R},\lambda} a, \bar{x} \approx_{\mathcal{R},\lambda} \lrcorner a \urcorner, \bar{y} \approx_{\mathcal{R},\lambda} \lrcorner b \urcorner, \bar{y} \approx_{\mathcal{R},\lambda} \lrcorner c, a \urcorner\}$,
 $\alpha = 0.8$

Step 4: $S^2 = \{x \approx_{\mathcal{R},\lambda} a, \bar{x} \approx_{\mathcal{R},\lambda} \ulcorner, \bar{x} \approx_{\mathcal{R},\lambda} \lrcorner a \urcorner, \bar{y} \approx_{\mathcal{R},\lambda} \lrcorner b \urcorner, \bar{y} \approx_{\mathcal{R},\lambda} \lrcorner c, a \urcorner\}$, $\alpha = 0.7$

Step 5: FAILURE ($\mathcal{R}(\ulcorner, \lrcorner a \urcorner) = 0 < \lambda$)

The same thing would also happen for S_{lin}^3 .

4.2 Fuzzy Unranked Matching with Proximity Relations

This section combines the algorithm of [10] for syntactic unranked matching and the one from *Matching and Generalization Modulo Proximity and Tolerance Relations* by Temur Kutsia and Cleo Pau [34] for ranked matching with proximity relations to create a procedure to match unranked terms under consideration of a proximity relation \mathcal{R} and a cut value λ . Similar to the original unranked matching in [10] and our modification for similarity relations in Section 4.1, this algorithm will also be split for linear and non-linear problems. However, in contrast to the algorithm for solving with similarity relations, we will not be actively tracking the degree. We will again work on sequences of terms and therefore have a rule removing the heads of terms.

We get the following set of rules:

T- M_{prox} : **Trivial**

$$\langle \{\bar{s} \lesssim_{\mathcal{R},\lambda}^? \bar{s}\} \cup \Gamma; S \rangle \Longrightarrow \langle \Gamma; S \rangle.$$

HR- M_{prox} : **Head Removal**

$$\langle \{f(\bar{s}) \lesssim_{\mathcal{R},\lambda}^? g(\bar{t})\} \cup \Gamma; S \rangle \Longrightarrow \langle \{\bar{s} \lesssim_{\mathcal{R},\lambda}^? \bar{t}\} \cup \Gamma; S \rangle,$$

if $\mathcal{R}(f, g) \geq \lambda$.

DEC-M_{prox}: Decomposition

$$\langle \{\ulcorner s, \tilde{s} \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner t, \tilde{t} \urcorner\} \cup \Gamma; S \rangle \Longrightarrow \langle \{s \lesssim_{\mathcal{R}, \lambda}^? t, \tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma; S \rangle,$$

where $s \notin \mathcal{V}_S$.

IVE-M_{prox}: Individual Variable Elimination

$$\langle \{x \lesssim_{\mathcal{R}, \lambda}^? s\} \cup \Gamma; S \rangle \Longrightarrow \langle \Gamma; S \cup \{x \approx_{\mathcal{R}, \lambda} \mathbf{pc}_{\mathcal{R}, \lambda}(s)\} \rangle,$$

with $s \notin \mathcal{V}_S$.

SVE-M_{prox}: Sequence Variable Elimination

$$\langle \{\ulcorner \bar{x}, \tilde{s} \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner \tilde{t}_1, \tilde{t}_2 \urcorner\} \cup \Gamma; S \rangle \Longrightarrow \langle \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}_2\} \cup \Gamma; S \cup \{\bar{x} \approx_{\mathcal{R}, \lambda} \mathbf{pc}_{\mathcal{R}, \lambda}(\tilde{t}_1)\} \rangle.$$

CLA-M_{prox}: Clash

$$\langle \{f(\tilde{s}) \lesssim_{\mathcal{R}, \lambda}^? g(\tilde{t})\} \cup \Gamma; S \rangle \Longrightarrow \perp,$$

if $\mathcal{R}(f, g) < \lambda$.

AD-M_{prox}: Arity Disagreement

$$\langle \{\ulcorner s_1, \dots, s_n \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner t_1, \dots, t_m \urcorner\} \cup \Gamma; S \rangle \Longrightarrow \perp$$

if $n \neq m$; $s_i \notin \mathcal{V}_S$ for $1 \leq i \leq n$ and $\mathcal{R}(f, g) \geq \lambda$.

E-M_{prox}: Empty

$$\langle \{\ulcorner s, \tilde{s} \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner \urcorner\} \cup \Gamma; S \rangle \Longrightarrow \perp,$$

with $s \notin \mathcal{V}_S$

This procedure $L\mathcal{M}_{\text{prox}}$ works on pairs of the shape $\langle \Gamma; S \rangle$, with Γ being a linear set of matching problems and S being the store of equations in solved form. Given the initial configuration of $\langle \{s \lesssim_{\mathcal{R}, \lambda}^? t\}; \emptyset \rangle$, a proximity relation \mathcal{R} and a cut value λ , the algorithm applies the rules until either the state $\langle \emptyset; S \rangle$ is reached, meaning that the matching was a success, or it ends with one of the failure rules.

Again branching can occur as rule (SVE-M_{prox}) has multiple ways of how it can be set up. From our computed (\mathcal{R}, λ) -answer S we get an X-substitution $\{\sigma\} = \Sigma(S)$ from which we can extract the set of (\mathcal{R}, λ) -matchers as $\{\sigma \mid \sigma \in \text{subs}(\sigma)\}$.

For the rest of this section, if not explicitly stated otherwise \mathcal{R} will refer to a given proximity relation and $\lambda \in (0, 1]$ to a cut value. An explanation on how to compute the approximation degree will be given at the end of this section.

Definition 4.2

A substitution σ is an (\mathcal{R}, λ) -solution of $\langle \Gamma; S \rangle$ if and only if for all $x \approx \mathbf{t} \in S$ and $\bar{x} \approx \tilde{\mathbf{t}} \in S$ we have $x\sigma \in \text{terms}(\mathbf{t})$ and $\bar{x}\sigma \in \text{terms}(\tilde{\mathbf{t}})$; and σ is an (\mathcal{R}, λ) -matcher of Γ .

Definition 4.3

A substitution σ is called a relevant matcher of a term s to a term t , if it is a matcher for s

to t and $\text{dom}(\sigma) \subseteq \mathcal{V}(s)$. A relevant matcher for sequences of terms and X -terms is defined analogously.

Lemma 4.11

The algorithm $L\mathfrak{M}_{\text{prox}}$ terminates for every input.

Proof. To prove the termination of the procedure we create a complexity measure $\langle m_1, m_2 \rangle$ ordered by the lexicographic ordering on tuples and show that each step reduces it strictly. We define

$$m_1 = \text{the number of variables in } \Gamma,$$

$$m_2 = \{\{\text{size}(\tilde{t}) \mid \tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t} \in \Gamma\}\}.$$

which gives us a well-founded complexity measure as $m_1, m_2 \in \mathbb{N}$.

Now we can show that each of the rules in $L\mathfrak{M}_{\text{prox}}$ strictly decreases $\langle m_1, m_2 \rangle$:

	m_1	m_2
T- M_{prox}	\geq	$>$
HR- M_{prox}	\geq	$>$
DEC- M_{prox}	\geq	$>$
IVE- M_{prox}	$>$	
SVE- M_{prox}	$>$	

Since the rules (CLA- M_{prox}), (AD- M_{prox}) and (E- M_{prox}) all stop the procedure immediately, the algorithm terminates for all inputs. \square

Lemma 4.12

Given a proximity relation \mathcal{R} and a cut value λ , if $\langle \Gamma_1; S_1 \rangle \Longrightarrow \langle \Gamma_2; S_2 \rangle$ is a step in $L\mathfrak{M}_{\text{prox}}$ with $S_2 = S_1 \cup S$ for some S , then σ is an (\mathcal{R}, λ) -solution of $\langle \Gamma_1; S_1 \rangle$ if and only if it is an (\mathcal{R}, λ) -solution of $\langle \Gamma_2; S_2 \rangle$.

Proof. Only the proofs for a few of the most interesting rules will be shown:

(HR- M_{prox}) We have $\langle \{f(\tilde{s}) \lesssim_{\mathcal{R}, \lambda}^? g(\tilde{t})\} \cup \Gamma'_1; S_1 \rangle \Longrightarrow \langle \Gamma_2; S_2 \rangle$ with $\Gamma_2 = \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma'_1$, $S_2 = S_1$ and $\mathcal{R}(f, g) \geq \lambda$.

“ \subseteq ” We know that σ is an (\mathcal{R}, λ) -solution of $\langle \Gamma_1; S_1 \rangle$ and want to show that σ is an (\mathcal{R}, λ) -solution of $\langle \Gamma_2; S_2 \rangle$. This means we have to show that for all $x \approx \mathbf{q} \in S_2$ and for all $\bar{x} \approx \tilde{\mathbf{q}} \in S_2$ it holds that $x\sigma \in \text{terms}(\mathbf{q})$ and $\bar{x}\sigma \in \text{terms}(\tilde{\mathbf{q}})$; and σ is an (\mathcal{R}, λ) -matcher of Γ_2 .

Since $S_2 = S_1$ we already know from σ being an (\mathcal{R}, λ) -solution of $\langle \Gamma_1; S_1 \rangle$ that for all $x \approx \mathbf{q} \in S_2$ and for all $\bar{x} \approx \tilde{\mathbf{q}} \in S_2$ it holds that $x\sigma \in \text{terms}(\mathbf{q})$ and $\bar{x}\sigma \in \text{terms}(\tilde{\mathbf{q}})$.

In order to prove that σ is an (\mathcal{R}, λ) -matcher of Γ_2 , it has to be shown that for all $\tilde{r} \lesssim_{\mathcal{R}, \lambda}^? \tilde{q} \in \Gamma_2$ it holds that $\tilde{r}\sigma \simeq_{\mathcal{R}, \lambda} \tilde{q}$. It is already known that σ is an (\mathcal{R}, λ) -matcher for all the equations in Γ'_1 , leaving only $\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}$ to show. Given that

$$\begin{aligned} f(\tilde{s})\sigma \simeq_{\mathcal{R}, \lambda} g(\tilde{t}) &\text{ iff } \mathcal{R}(f(\tilde{s})\sigma, g(\tilde{t})) \geq \lambda \\ &\text{ iff } \mathcal{R}(f, g) \wedge \mathcal{R}(\tilde{s}\sigma, \tilde{t}) \geq \lambda \end{aligned}$$

it is clear that $\mathcal{R}(\tilde{s}\sigma, \tilde{t}) \geq \lambda$, which means that $\tilde{s}\sigma \simeq_{\mathcal{R}, \lambda} \tilde{t}$.

“ \supseteq ” We know that σ is an (\mathcal{R}, λ) -solution of $\langle \Gamma_2; S_2 \rangle$ and want to show that σ is an (\mathcal{R}, λ) -solution of $\langle \Gamma_1; S_1 \rangle$. Again since $S_1 = S_2$ it is already given that $x\sigma \in \text{terms}(\mathbf{q})$ and $\bar{x}\sigma \in \text{terms}(\tilde{\mathbf{q}})$ holds for all $x \approx \mathbf{q} \in S_1$ and for all $\bar{x} \approx \tilde{\mathbf{q}} \in S_1$.

For the proof that σ is an (\mathcal{R}, λ) -matcher of $\langle \Gamma_1; S_1 \rangle$ we only have to show that $\mathcal{R}(f(\tilde{s})\sigma, g(\tilde{t})) \geq \lambda$ holds. Since we know that $\tilde{s}\sigma \simeq_{\mathcal{R}, \lambda} \tilde{t}$ and $\mathcal{R}(f, g) \geq \lambda$, this follows immediately.

(DEC-M_{prox}) We have $\{\ulcorner s, \tilde{s} \urcorner \lesssim_{\mathcal{R}, \lambda}^? \ulcorner t, \tilde{t} \urcorner\} \cup \Gamma'_1; S_1 \implies \langle \Gamma_2; S_2 \rangle$ with $\Gamma_2 = \{s \lesssim_{\mathcal{R}, \lambda}^? t, \tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma'_1$ and $S = \emptyset$, which means that $S_2 = S_1$.

“ \subseteq ” Since $S_2 = S_1$ we already know from σ being an (\mathcal{R}, λ) -solution of $\langle \Gamma_1; S_1 \rangle$ that for all $x \approx \mathbf{q} \in S_2$ and for all $\bar{x} \approx \tilde{\mathbf{q}} \in S_2$ it holds that $x\sigma \in \text{terms}(\mathbf{q})$ and $\bar{x}\sigma \in \text{terms}(\tilde{\mathbf{q}})$.

In order to prove that σ is an (\mathcal{R}, λ) -matcher of Γ_2 , it has to be shown that for all $\tilde{r} \lesssim_{\mathcal{R}, \lambda}^? \tilde{q} \in \Gamma_2$ it holds that $\tilde{r}\sigma \simeq_{\mathcal{R}, \lambda} \tilde{q}$. It is already known that σ is an (\mathcal{R}, λ) -matcher for all the equations in Γ'_1 , leaving only $s \lesssim_{\mathcal{R}, \lambda}^? t$ and $\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}$ to show. Given that

$$\begin{aligned} \ulcorner s, \tilde{s} \urcorner \sigma \simeq_{\mathcal{R}, \lambda} \ulcorner t, \tilde{t} \urcorner &\text{ iff } \mathcal{R}(\ulcorner s, \tilde{s} \urcorner \sigma, \ulcorner t, \tilde{t} \urcorner) \geq \lambda \\ &\text{ iff } \mathcal{R}(s\sigma, t) \wedge \mathcal{R}(\tilde{s}\sigma, \tilde{t}) \geq \lambda \end{aligned}$$

it is clear that $\mathcal{R}(s\sigma, t) \geq \lambda$ and $\mathcal{R}(\tilde{s}\sigma, \tilde{t}) \geq \lambda$, which means that $s\sigma \simeq_{\mathcal{R}, \lambda} t$ and $\tilde{s}\sigma \simeq_{\mathcal{R}, \lambda} \tilde{t}$.

“ \supseteq ” We know that σ is an (\mathcal{R}, λ) -solution of $\langle \Gamma_2; S_2 \rangle$ and want to show that σ is an (\mathcal{R}, λ) -solution of $\langle \Gamma_1; S_1 \rangle$. Again since $S_1 = S_2$ it is already given that $x\sigma \in \text{terms}(\mathbf{q})$ and $\bar{x}\sigma \in \text{terms}(\tilde{\mathbf{q}})$ holds for all $x \approx \mathbf{q} \in S_1$ and for all $\bar{x} \approx \tilde{\mathbf{q}} \in S_1$.

The proof that σ is an (\mathcal{R}, λ) -matcher of $\langle \Gamma_1; S_1 \rangle$ can be done analogously to the other direction.

(IVE- M_{prox}) We have $\langle \{x \lesssim_{\mathcal{R}, \lambda}^? s\} \cup \Gamma'_1; S_1 \rangle \Longrightarrow \langle \Gamma_2; S_2 \rangle$ with $\Gamma_2 = \Gamma'_1$ and $S = \{x \approx_{\mathcal{R}, \lambda} \mathbf{pc}_{\mathcal{R}, \lambda}(s)\}$.

“ \subseteq ” Since $\Gamma_2 = \Gamma'_1$, σ being an (\mathcal{R}, λ) -solution of Γ_2 is already given by it being a solution to Γ_1 and therefore Γ'_1 .

It is already known that for all $x \approx \mathbf{q} \in S_1$ and for all $\bar{x} \approx \tilde{\mathbf{q}} \in S_1$, $x\sigma \in \text{terms}(\mathbf{q})$ and $\bar{x}\sigma \in \text{terms}(\tilde{\mathbf{q}})$. This leaves us to show for $x \approx_{\mathcal{R}, \lambda} \mathbf{pc}_{\mathcal{R}, \lambda}(s)$ that $x\sigma \in \text{terms}(\mathbf{pc}_{\mathcal{R}, \lambda}(s))$. In order to prove $x\sigma \in \text{terms}(\mathbf{pc}_{\mathcal{R}, \lambda}(s))$ it suffices to show that $x\sigma \simeq_{\mathcal{R}, \lambda} s$, which we know from the fact that σ is an (\mathcal{R}, λ) -solution of Γ_1 .

“ \supseteq ” We know that σ is an (\mathcal{R}, λ) -solution of $\langle \Gamma_2; S_2 \rangle$ and want to show that σ is an (\mathcal{R}, λ) -solution of $\langle \Gamma_1; S_1 \rangle$. For this direction we already know for all $x \approx \mathbf{q} \in S_2$ and for all $\bar{x} \approx \tilde{\mathbf{q}} \in S_2$ that $x\sigma \in \text{terms}(\mathbf{q})$ and $\bar{x}\sigma \in \text{terms}(\tilde{\mathbf{q}})$ holds as $S_2 = S_1 \cup \{x \approx_{\mathcal{R}, \lambda} \mathbf{pc}_{\mathcal{R}, \lambda}(s)\}$.

Proving that σ is an (\mathcal{R}, λ) -matcher of Γ_1 can be reduced to showing that for $x \lesssim_{\mathcal{R}, \lambda}^? s$ the condition $x\sigma \simeq_{\mathcal{R}, \lambda} s$ holds. Since for $x \approx_{\mathcal{R}, \lambda} s \in S_2$ we already know that $x\sigma \in \text{terms}(\mathbf{pc}_{\mathcal{R}, \lambda}(s))$ we are already done with this part. □

Theorem 4.13 (Soundness of $L\mathfrak{M}_{\text{prox}}$)

Let \mathcal{R} be a proximity relation, λ a cut value and $\Gamma = \{s \lesssim_{\mathcal{R}, \lambda}^? t\}$ a linear matching problem. If the procedure $L\mathfrak{M}_{\text{prox}}$ constructs a derivation of the form $\langle \Gamma; \emptyset \rangle \Longrightarrow^+ \langle \emptyset; S \rangle$, then $\sigma \in \text{subs}(\sigma)$ is a relevant matcher of s to t with $\sigma = \Sigma(S)$.

Proof. We know that $\sigma \in \text{subs}(\sigma)$ with $\sigma = \Sigma(S)$ if and only if σ is an (\mathcal{R}, λ) -solution of $\langle \emptyset; S \rangle$. Given the construction of $\langle \Gamma; \emptyset \rangle \Longrightarrow^+ \langle \emptyset; S \rangle$ we can use induction on Lemma 4.12 to show that σ is an (\mathcal{R}, λ) -matcher of s to t . Seeing as no new variables are created and added to S while applying the rules of $L\mathfrak{M}_{\text{prox}}$ and therefore σ is a relevant (\mathcal{R}, λ) -matcher of s to t . □

Theorem 4.14 (Completeness of $L\mathfrak{M}_{\text{prox}}$)

Let \mathcal{R} be a proximity relation, λ a cut value and $\Gamma = \{s \lesssim_{\mathcal{R}, \lambda}^? t\}$ a linear matching problem. If σ is a relevant (\mathcal{R}, λ) -matcher of s to t and the procedure $L\mathfrak{M}_{\text{prox}}$ constructs a derivation of the form $\langle \Gamma; \emptyset \rangle \Longrightarrow^+ \langle \emptyset; S \rangle$, then $\sigma \in \text{subs}(\sigma)$ with $\sigma = \Sigma(S)$.

Proof. Similarly to the proof of Theorem 4.7, we will first show how to construct the derivation via induction:

For our induction hypothesis we say that we have already executed the procedure steps up until $\langle \Gamma_n; S_n \rangle$ and it holds that σ is an (\mathcal{R}, λ) -matcher of Γ_n and $\sigma|_{\mathcal{V}(S_n)} \in \text{subs}(\sigma_n)$ with $\sigma_n = \Sigma(S_n)$.

Next we prove that we can make the step $\langle \Gamma_n; S_n \rangle \Longrightarrow_R \langle \Gamma_{n+1}; S_{n+1} \rangle$, such that σ is an (\mathcal{R}, λ) -matcher of Γ_{n+1} and $\sigma|_{\mathcal{V}(S_{n+1})} \in \text{subs}(\sigma_{n+1})$ with $\sigma_{n+1} = \Sigma(S_{n+1})$. Looking at a pair $s' \lesssim_{\mathcal{R}, \lambda}^? t' \in \Gamma_n$, we get the following options for rule R for the next step of the derivation:

- 1) We chose R as T-M_{prox} if $s' = t'$. This means we get $\langle \Gamma_n; S_n \rangle \Longrightarrow_{\text{T-M}_{\text{prox}}} \langle \Gamma'_n; S_n \rangle$ with $\Gamma_{n+1} = \Gamma'_n$, $S_{n+1} = S_n$. Due to only removing a trivial pairing from the problem set, it is clear that the properties of σ being an (\mathcal{R}, λ) -matcher of Γ_{n+1} and $\sigma|_{\mathcal{V}(S_{n+1})} \in \text{subs}(\sigma_{n+1})$ with $\sigma_{n+1} = \Sigma(S_{n+1})$ hold.
- 2) If $s' = f(\tilde{s})$ and $t' = g(\tilde{t})$ with $\mathcal{R}(f, g) \geq \lambda$, we use HR-M_{prox} and get $\langle \Gamma_n; S_n \rangle \Longrightarrow_{\text{HR-M}_{\text{prox}}} \langle \{s' \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma'_n; S_n \rangle$ with $\Gamma_{n+1} = \{s' \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma'_n$ and $S_{n+1} = S_n$. σ is still an (\mathcal{R}, λ) -matcher of Γ_{n+1} as the set of variables occurring in Γ_n and Γ_{n+1} are equal. It holds that $\sigma|_{\mathcal{V}(S_{n+1})} = \sigma|_{\mathcal{V}(S_n)} \in \text{subs}(\sigma_n)$, since $S_{n+1} = S_n$.
- 3) If $s' = \ulcorner s'_0, \tilde{s}^\urcorner$ and $t' = \ulcorner t'_0, \tilde{t}^\urcorner$ with $s_1 \notin \mathcal{V}_S$, we use the DEC-M_{prox} rule and get $\langle \Gamma_n; S_n \rangle \Longrightarrow_{\text{DEC-M}_{\text{prox}}} \langle \{s'_0 \lesssim_{\mathcal{R}, \lambda}^? t'_0, \tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma'_n; S_n \rangle$ with $\Gamma_{n+1} = \{s'_0 \lesssim_{\mathcal{R}, \lambda}^? t'_0, \tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}\} \cup \Gamma'_n$ and $S_{n+1} = S_n$. Since $\mathcal{V}(\Gamma_n) = \mathcal{V}(\Gamma_{n+1})$, we know that σ is still an (\mathcal{R}, λ) -matcher of Γ_{n+1} . Due to $S_{n+1} = S_n$ we know that $\sigma|_{\mathcal{V}(S_{n+1})} = \sigma|_{\mathcal{V}(S_n)} \in \text{subs}(\sigma_n)$.
- 4) In the case that $s' = x \in \mathcal{V}_I$ we have to use IVE-M_{prox} for R . We extend the derivation by $\langle \Gamma_n; S_n \rangle \Longrightarrow_{\text{IVE-M}_{\text{prox}}} \langle \Gamma'_n; S_n \cup \{x \approx_{\mathcal{R}, \lambda} \mathbf{pc}_{\mathcal{R}, \lambda}(t')\} \rangle$ with $\Gamma_{n+1} = \Gamma'_n$ and $S_{n+1} = S_n \cup \{x \approx_{\mathcal{R}, \lambda} \mathbf{pc}_{\mathcal{R}, \lambda}(t')\}$. The substitution σ is still an (\mathcal{R}, λ) -matcher of Γ_{n+1} since we only removed a pair from Γ_n , and it holds that $\sigma|_{\mathcal{V}(S_{n+1})} = \sigma|_{\mathcal{V}(\Gamma_n)} \cup \{x \mapsto \mathbf{pc}_{\mathcal{R}, \lambda}(t')\} \in \text{subs}(\sigma_{n+1})$.
- 5) For $s' = \ulcorner \bar{x}, \tilde{s}^\urcorner$ and $t' = \ulcorner \tilde{t}_1, \tilde{t}_2^\urcorner$ we use SVE-M_{prox} for R . This gives us $\langle \Gamma_n; S_n \rangle \Longrightarrow_{\text{SVE-M}_{\text{prox}}} \langle \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}_2\} \cup \Gamma'_n; S_n \cup \{\bar{x} \mapsto \mathbf{pc}_{\mathcal{R}, \lambda}(\tilde{t}_1)\} \rangle$ with $\Gamma_{n+1} = \{\tilde{s} \lesssim_{\mathcal{R}, \lambda}^? \tilde{t}_2\} \cup \Gamma'_n$ and $S_{n+1} = S_n \cup \{\bar{x} \mapsto \mathbf{pc}_{\mathcal{R}, \lambda}(\tilde{t}_1)\}$. It is clear that σ is an (\mathcal{R}, λ) -matcher of Γ_{n+1} and $\sigma|_{\mathcal{V}(S_{n+1})} \in \text{subs}(\sigma_{n+1})$.

From the fact that σ is a relevant (\mathcal{R}, λ) -matcher of Γ , we know that we do not need to apply any of the terminating rules. Therefore we can simply continue to extend this derivation until we hit the final configuration of $\langle \emptyset; S \rangle$, for which we know that σ is an (\mathcal{R}, λ) -solution. By the way we have computed S , we get that $\sigma \in \text{subs}(\sigma)$ with $\sigma = \Sigma(S)$. \square

In order to be able to solve non-linear matching problems with proximity relations, we will again modify the *Reconstruct Solutions* algorithm from [10]:

Algorithm 3 RS_{prox} - Reconstruct Solutions

Input: S_{lin}

Output: S

- 1: $S := S_{\text{lin}}$
 - 2: Reinststate the original names of all individual and function variables in S .
 Example: $S = \{v_x^1 \approx_{\mathcal{R},\lambda} \mathbf{t}_1, v_x^2 \approx_{\mathcal{R},\lambda} \mathbf{t}_2\} \cup S'$ will be changed to
 $S = \{x \approx_{\mathcal{R},\lambda} \mathbf{t}_1, x \approx_{\mathcal{R},\lambda} \mathbf{t}_2\} \cup S'$.
 If $\mathbf{t}_1 = \mathbf{t}_2$, then $S = \{x \approx_{\mathcal{R},\lambda} \mathbf{t}_1\}$ because S is a set.
 - 3: If S contains two pairs $x \approx_{\mathcal{R},\lambda} \mathbf{t}_1$ and $x \approx_{\mathcal{R},\lambda} \mathbf{t}_2$ for the same individual variable $x \in \mathcal{V}_I$ with $\mathbf{s} = \mathbf{t}_1 \sqcap \mathbf{t}_2 = \emptyset$, then STOP with failure.
 - 4: If S contains two pairs $x \approx_{\mathcal{R},\lambda} \mathbf{t}_1$ and $x \approx_{\mathcal{R},\lambda} \mathbf{t}_2$ for the same individual variable $x \in \mathcal{V}_I$ with $\mathbf{s} = \mathbf{t}_1 \sqcap \mathbf{t}_2 \neq \emptyset$, then set $S := \{x \approx_{\mathcal{R},\lambda} \mathbf{s}\} \cup S'$. Repeat Step 4.
 - 5: If S contains two pairs $\bar{x} \approx_{\mathcal{R},\lambda} \tilde{\mathbf{r}}$ and $\bar{x} \approx_{\mathcal{R},\lambda} \tilde{\mathbf{t}}$ for the same sequence variable $\bar{x} \in \mathcal{V}_S$ with $\mathbf{s} = \tilde{\mathbf{r}} \sqcap \tilde{\mathbf{t}} = \emptyset$, then STOP with failure.
 - 6: If S contains two pairs $\bar{x} \approx_{\mathcal{R},\lambda} \tilde{\mathbf{r}}$ and $\bar{x} \approx_{\mathcal{R},\lambda} \tilde{\mathbf{t}}$ for the same sequence variable $\bar{x} \in \mathcal{V}_S$ with $\tilde{\mathbf{s}} = \tilde{\mathbf{r}} \sqcap \tilde{\mathbf{t}} \neq \emptyset$, then set $S := \{\bar{x} \approx_{\mathcal{R},\lambda} \tilde{\mathbf{s}}\} \cup S'$. Repeat Step 6.
 - 7: **return** S .
-

Reminder: all the right hand sides of the solved equations in S are X-terms, obtained from proximity classes and their intersections.

Theorem 4.15 (Termination of RS_{prox})

The procedure RS_{prox} terminates for every input.

Proof. Same as for Theorem 4.1 steps 1 and 2 are only executed once at the start and steps 3 and 5 lead to immediate termination. Step 4 and Step 6 are only executed a finite number of times, depending on the variables occurring in S . Therefore RS_{prox} terminates. \square

Theorem 4.16 (Soundness of RS_{prox})

Let \mathcal{R} be a proximity relation, λ a cut value, $\Gamma = \{s \hat{\approx}_{\mathcal{R},\lambda}^? t\}$ a non-linear matching problem and $S_{\text{lin}} \in L\mathcal{M}_{\text{prox}}(\Gamma_{\text{lin}}, \mathcal{R}, \lambda)$ an answer of the linearized problem. If the application of RS_{prox} to S_{lin} constructs S , then $\sigma \in \text{subs}(\Sigma(S))$ is a relevant matcher of s to t .

Proof. The proof is similar to that of Theorem 4.9. \square

Theorem 4.17 (Completeness of RS_{prox})

Let \mathcal{R} be a proximity relation, λ a cut value, $\Gamma = \{s \stackrel{?}{\sim}_{\mathcal{R},\lambda} t\}$ a non-linear matching problem and σ a relevant (\mathcal{R}, λ) -matcher of s to t . Let $S_{\text{lin}} \in L\mathcal{M}_{\text{prox}}(\Gamma_{\text{lin}}, \mathcal{R}, \lambda)$. Then there exists a derivation performed by RS_{prox} starting from S_{lin} that computes S such that $\sigma \in \text{subs}(\Sigma(S))$.

Proof. The proof is similar to that of Theorem 4.10. \square

If we want to know the exact proximity degree of our (\mathcal{R}, λ) -matchers, we can change the procedure slightly: The linear matching algorithm would then work on triples $\langle \Gamma; S; \alpha \rangle$, with α collecting the degrees between function symbols and being initialized by 1. Instead of regular proximity classes, we would use graded proximity classes, as introduced in Section 2.4. We have to slightly change the rule $\text{HR-M}_{\text{prox}}$:

HR-M_{prox}: Head Removal

$$\langle \{f(\tilde{s}) \stackrel{?}{\sim}_{\mathcal{R},\lambda} g(\tilde{t})\} \cup \Gamma; S; \alpha \rangle \Longrightarrow \langle \{\tilde{s} \stackrel{?}{\sim}_{\mathcal{R},\lambda} \tilde{t}\} \cup \Gamma; S; \alpha \wedge \beta \rangle,$$

if $\mathcal{R}(f, g) = \beta \geq \lambda$

All of the other rules simply carry the degree along.

Let $\langle \emptyset; S; \alpha \rangle$ be the final configuration of the modified $L\mathcal{M}_{\text{prox}}$. For the graded substitution $\langle \sigma, \alpha_\sigma \rangle \in \text{subs}(\Sigma(S))$ we can compute the actual degree of the (\mathcal{R}, λ) -matcher σ by taking the minimum of the degree between the function symbols α , computed by $L\mathcal{M}_{\text{prox}}$, and α_σ , i.e. $\alpha \wedge \alpha_\sigma$ is the approximation degree of the (\mathcal{R}, λ) -matcher σ .

In the case that we have to first use the RS_{prox} algorithm to get the solutions for the non-linear problem, RS_{prox} is used on the graded proximity classes and the function symbol degree α , which was computed by $L\mathcal{M}_{\text{prox}}$, gets combined with the degrees returned by RS_{prox} .

The following example is slightly modified from [34]:

Example 4.2. Let $\Gamma = \{f(x, x, \bar{x}) \stackrel{?}{\sim}_{\mathcal{R},\lambda} g(h(a), p(b), c, d)\}$ be a matching problem with a proximity relation \mathcal{R} defined as $\mathcal{R}(f, g) = 0.8$, $\mathcal{R}(h, p) = 0.8$, $\mathcal{R}(a, b) = 0.9$, $\mathcal{R}(h, d) = 0.7$, $\mathcal{R}(p, c) = 0.6$, $\mathcal{R}(p, d) = 0.8$ and a cut value $\lambda = 0.6$.

The linearized version of Γ would be $\Gamma_{\text{lin}} = \{f(v_x^1, v_x^2, \bar{v}_x^1) \stackrel{?}{\sim}_{\mathcal{R},\lambda} g(h(a), p(b), c, d)\}$.

Now we use $L\mathcal{M}_{\text{prox}}$ to find a solution for Γ_{lin} :

$$\begin{array}{l} \langle \Gamma_{\text{lin}}; \emptyset; 1 \rangle \\ \xrightarrow{\text{HR-M}_{\text{prox}}} \langle \{\ulcorner v_x^1, v_x^2, \bar{v}_x^1 \urcorner \stackrel{?}{\sim}_{\mathcal{R},\lambda} \ulcorner h(a), p(b), c, d \urcorner\}; \emptyset; 0.8 \rangle \end{array}$$

$$\begin{aligned}
 \text{DEC-M}_{\text{prox}} &\xrightarrow{\implies} \langle \{v_x^1 \lesssim_{\mathcal{R},\lambda}^? h(a), \ulcorner v_x^2, \bar{v}_x^1 \urcorner \lesssim \ulcorner p(b), c, d \urcorner\}; \emptyset; 0.8 \rangle \\
 \text{IVE-M}_{\text{prox}} &\xrightarrow{\implies} \langle \{\ulcorner v_x^2, \bar{v}_x^1 \urcorner \lesssim \ulcorner p(b), c, d \urcorner\}; \\
 &\quad \{v_x^1 \approx_{\mathcal{R},\lambda} \{\langle h, 1 \rangle, \langle p, 0.8 \rangle\}(\{\langle a, 1 \rangle, \langle b, 0.9 \rangle\})\}; 0.8 \rangle \\
 \text{DEC-M}_{\text{prox}} &\xrightarrow{\implies} \langle \{v_x^2 \lesssim_{\mathcal{R},\lambda}^? p(b), \bar{v}_x^1 \lesssim_{\mathcal{R},\lambda}^? \ulcorner c, d \urcorner\}; \\
 &\quad \{v_x^1 \approx_{\mathcal{R},\lambda} \{\langle h, 1 \rangle, \langle p, 0.8 \rangle\}(\{\langle a, 1 \rangle, \langle b, 0.9 \rangle\})\}; 0.8 \rangle \\
 \text{IVE-M}_{\text{prox}} &\xrightarrow{\implies} \langle \{\bar{v}_x^1 \lesssim_{\mathcal{R},\lambda}^? \ulcorner c, d \urcorner\}; \\
 &\quad \{v_x^1 \approx_{\mathcal{R},\lambda} \{\langle h, 1 \rangle, \langle p, 0.8 \rangle\}(\{\langle a, 1 \rangle, \langle b, 0.9 \rangle\}), \\
 &\quad v_x^2 \approx_{\mathcal{R},\lambda} \{\langle p, 1 \rangle, \langle h, 0.8 \rangle, \langle c, 0.6 \rangle, \langle d, 0.8 \rangle\}(\{\langle b, 1 \rangle, \langle a, 0.9 \rangle\})\}; 0.8 \rangle \\
 \text{SVE-M}_{\text{prox}} &\xrightarrow{\implies} \langle \{\ulcorner \urcorner \lesssim_{\mathcal{R},\lambda}^? \ulcorner \urcorner\}; \\
 &\quad \{v_x^1 \approx_{\mathcal{R},\lambda} \{\langle h, 1 \rangle, \langle p, 0.8 \rangle\}(\{\langle a, 1 \rangle, \langle b, 0.9 \rangle\}), \\
 &\quad v_x^2 \approx_{\mathcal{R},\lambda} \{\langle p, 1 \rangle, \langle h, 0.8 \rangle, \langle c, 0.6 \rangle, \langle d, 0.8 \rangle\}(\{\langle b, 1 \rangle, \langle a, 0.9 \rangle\}), \\
 &\quad \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \ulcorner \{\langle c, 1 \rangle, \langle p, 0.6 \rangle\}, \{\langle d, 1 \rangle, \langle h, 0.7 \rangle, \langle p, 0.8 \rangle\} \urcorner\}; 0.8 \rangle \\
 \text{T-M}_{\text{prox}} &\xrightarrow{\implies} \langle \emptyset; \\
 &\quad \{v_x^1 \approx_{\mathcal{R},\lambda} \{\langle h, 1 \rangle, \langle p, 0.8 \rangle\}(\{\langle a, 1 \rangle, \langle b, 0.9 \rangle\}), \\
 &\quad v_x^2 \approx_{\mathcal{R},\lambda} \{\langle p, 1 \rangle, \langle h, 0.8 \rangle, \langle c, 0.6 \rangle, \langle d, 0.8 \rangle\}(\{\langle b, 1 \rangle, \langle a, 0.9 \rangle\}), \\
 &\quad \bar{v}_x^1 \approx_{\mathcal{R},\lambda} \ulcorner \{\langle c, 1 \rangle, \langle p, 0.6 \rangle\}, \{\langle d, 1 \rangle, \langle h, 0.7 \rangle, \langle p, 0.8 \rangle\} \urcorner\}; 0.8 \rangle
 \end{aligned}$$

Next we use RS_{prox} to transform the solved set S_{lin} we just computed with $L\mathcal{M}_{\text{prox}}$ to a solution of Γ :

Step 1: $S := S_{\text{lin}}$

Step 2: $S = \{x \approx_{\mathcal{R},\lambda} \{\langle h, 1 \rangle, \langle p, 0.8 \rangle\}(\{\langle a, 1 \rangle, \langle b, 0.9 \rangle\}),$
 $x \approx_{\mathcal{R},\lambda} \{\langle p, 1 \rangle, \langle h, 0.8 \rangle, \langle c, 0.6 \rangle, \langle d, 0.8 \rangle\}(\{\langle b, 1 \rangle, \langle a, 0.9 \rangle\}),$
 $\bar{x} \approx_{\mathcal{R},\lambda} \ulcorner \{\langle c, 1 \rangle, \langle p, 0.6 \rangle\}, \{\langle d, 1 \rangle, \langle h, 0.7 \rangle, \langle p, 0.8 \rangle\} \urcorner\}$

Step 4: $S = \{x \approx_{\mathcal{R},\lambda} \{\langle h, 0.8 \rangle, \langle p, 0.8 \rangle\}(\{\langle a, 0.9 \rangle, \langle b, 0.9 \rangle\}),$
 $\bar{x} \approx_{\mathcal{R},\lambda} \ulcorner \{\langle c, 1 \rangle, \langle p, 0.6 \rangle\}, \{\langle d, 1 \rangle, \langle h, 0.7 \rangle, \langle p, 0.8 \rangle\} \urcorner\}$

Here we can see the computed solutions in a compact format in

$$\begin{aligned}
 S &= \{x \approx_{\mathcal{R},\lambda} \{\langle h, 0.8 \rangle, \langle p, 0.8 \rangle\}(\{\langle a, 0.9 \rangle, \langle b, 0.9 \rangle\}), \\
 &\quad \bar{x} \approx_{\mathcal{R},\lambda} \ulcorner \{\langle c, 1 \rangle, \langle p, 0.6 \rangle\}, \{\langle d, 1 \rangle, \langle h, 0.7 \rangle, \langle p, 0.8 \rangle\} \urcorner\}
 \end{aligned}$$

from which we can get the following (\mathcal{R}, λ) -matchers of Γ with their proximity degrees:

$$\sigma_1 = \{x \mapsto h(a), \bar{x} \mapsto \ulcorner c, d \urcorner\}, \quad \alpha_1 = 0.8 \wedge 0.9 \wedge 1 \wedge 1 \wedge 0.8 = 0.8$$

$\sigma_2 = \{x \mapsto p(a), \bar{x} \mapsto \lceil c, d \rceil\},$	$\alpha_2 = 0.8 \wedge 0.9 \wedge 1 \wedge 1 \wedge 0.8 = 0.8$
$\sigma_3 = \{x \mapsto h(b), \bar{x} \mapsto \lceil c, d \rceil\},$	$\alpha_3 = 0.8 \wedge 0.9 \wedge 1 \wedge 1 \wedge 0.8 = 0.8$
$\sigma_4 = \{x \mapsto p(b), \bar{x} \mapsto \lceil c, d \rceil\},$	$\alpha_4 = 0.8 \wedge 0.9 \wedge 1 \wedge 1 \wedge 0.8 = 0.8$
$\sigma_5 = \{x \mapsto h(a), \bar{x} \mapsto \lceil c, h \rceil\},$	$\alpha_5 = 0.8 \wedge 0.9 \wedge 1 \wedge 0.7 \wedge 0.8 = 0.7$
$\sigma_6 = \{x \mapsto p(a), \bar{x} \mapsto \lceil c, h \rceil\},$	$\alpha_6 = 0.8 \wedge 0.9 \wedge 1 \wedge 0.7 \wedge 0.8 = 0.7$
$\sigma_7 = \{x \mapsto h(b), \bar{x} \mapsto \lceil c, h \rceil\},$	$\alpha_7 = 0.8 \wedge 0.9 \wedge 1 \wedge 0.7 \wedge 0.8 = 0.7$
$\sigma_8 = \{x \mapsto p(b), \bar{x} \mapsto \lceil c, h \rceil\},$	$\alpha_8 = 0.8 \wedge 0.9 \wedge 1 \wedge 0.7 \wedge 0.8 = 0.7$
$\sigma_9 = \{x \mapsto h(a), \bar{x} \mapsto \lceil c, p \rceil\},$	$\alpha_9 = 0.8 \wedge 0.9 \wedge 1 \wedge 0.8 \wedge 0.8 = 0.8$
$\sigma_{10} = \{x \mapsto p(a), \bar{x} \mapsto \lceil c, p \rceil\},$	$\alpha_{10} = 0.8 \wedge 0.9 \wedge 1 \wedge 0.8 \wedge 0.8 = 0.8$
$\sigma_{11} = \{x \mapsto h(b), \bar{x} \mapsto \lceil c, p \rceil\},$	$\alpha_{11} = 0.8 \wedge 0.9 \wedge 1 \wedge 0.8 \wedge 0.8 = 0.8$
$\sigma_{12} = \{x \mapsto p(b), \bar{x} \mapsto \lceil c, p \rceil\},$	$\alpha_{12} = 0.8 \wedge 0.9 \wedge 1 \wedge 0.8 \wedge 0.8 = 0.8$
$\sigma_{13} = \{x \mapsto h(a), \bar{x} \mapsto \lceil p, d \rceil\},$	$\alpha_{13} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 1 \wedge 0.8 = 0.6$
$\sigma_{14} = \{x \mapsto p(a), \bar{x} \mapsto \lceil p, d \rceil\},$	$\alpha_{14} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 1 \wedge 0.8 = 0.6$
$\sigma_{15} = \{x \mapsto h(b), \bar{x} \mapsto \lceil p, d \rceil\},$	$\alpha_{15} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 1 \wedge 0.8 = 0.6$
$\sigma_{16} = \{x \mapsto p(b), \bar{x} \mapsto \lceil p, d \rceil\},$	$\alpha_{16} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 1 \wedge 0.8 = 0.6$
$\sigma_{17} = \{x \mapsto h(a), \bar{x} \mapsto \lceil p, h \rceil\},$	$\alpha_{17} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 0.7 \wedge 0.8 = 0.6$
$\sigma_{18} = \{x \mapsto p(a), \bar{x} \mapsto \lceil p, h \rceil\},$	$\alpha_{18} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 0.7 \wedge 0.8 = 0.6$
$\sigma_{19} = \{x \mapsto h(b), \bar{x} \mapsto \lceil p, h \rceil\},$	$\alpha_{19} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 0.7 \wedge 0.8 = 0.6$
$\sigma_{20} = \{x \mapsto p(b), \bar{x} \mapsto \lceil p, h \rceil\},$	$\alpha_{20} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 0.7 \wedge 0.8 = 0.6$
$\sigma_{21} = \{x \mapsto h(a), \bar{x} \mapsto \lceil p, p \rceil\},$	$\alpha_{21} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 0.8 \wedge 0.8 = 0.6$
$\sigma_{22} = \{x \mapsto p(a), \bar{x} \mapsto \lceil p, p \rceil\},$	$\alpha_{22} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 0.8 \wedge 0.8 = 0.6$
$\sigma_{23} = \{x \mapsto h(b), \bar{x} \mapsto \lceil p, p \rceil\},$	$\alpha_{23} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 0.8 \wedge 0.8 = 0.6$
$\sigma_{24} = \{x \mapsto p(b), \bar{x} \mapsto \lceil p, p \rceil\},$	$\alpha_{24} = 0.8 \wedge 0.9 \wedge 0.6 \wedge 0.8 \wedge 0.8 = 0.6$

5 Anti-Unification

In this chapter, we will focus on a procedure to solve unranked anti-unification problems. The starting point is the paper by Temur Kutsia, Jordi Levy, and Mateu Villaret in *Anti-unification for Unranked Terms and Hedges* [30]. The paper, as reflected in its title, uses the term *hedges* in place of *sequences*; however, the two notions are essentially the same. To remain consistent with our naming convention, we will simply replace the term *hedge* with *sequence*.

As we did in the previous chapters, we first recall the existing crisp method (in this case, the generalization algorithms from [30]), and then show how it can be extended to deal with proximity and similarity relations.

The standard procedure \mathfrak{G} , introduced in [30], is a rule-based algorithm working on systems $\langle \Gamma; S; \sigma \rangle$, where Γ is a set of syntactic unranked anti-unification triple (AUTs) of the form $\{\bar{v}_1 : \tilde{s}_1 \triangleq^? \tilde{q}_1, \dots, \bar{v}_n : \tilde{s}_n \triangleq^? \tilde{t}_n\}$, with each generalization variable \bar{v}_i being unique and the sequences to be generalized being variable disjoint, S is a store of AUT's in solved form and σ is the anti-unifier computed so far. In Γ (which is called a syntactic unranked anti-unification problem), it is assumed that the sides of AUTs do not share variables. Starting the algorithm with the triple $\langle \{\bar{v} : \tilde{s} \triangleq^? \tilde{t}\}; \emptyset; \varepsilon \rangle$ and applying the rules defined in [30] as they fit, the procedure ends in $\langle \emptyset; S; \sigma \rangle$ with σ being an anti-unifier for Γ and $\bar{v}\sigma$ being a generalization for both \tilde{s} and \tilde{t} . From S we can create two substitutions (ν_1, ν_2) , as described in Section 2.6.3, such that $\bar{v}\sigma\nu_1 = \tilde{s}$ and $\bar{v}\sigma\nu_2 = \tilde{t}$.

However, since \mathfrak{G} is highly nondeterministic, a modified version called *Rigid Generalization* $\mathfrak{G}(\mathcal{R})$ is given in [30]. The idea behind introducing rigidity is to maintain as much structure of the terms as possible, in order to reduce some of the less interesting solutions computed by the standard procedure. Its main task is deciding which common subsequence of two given sequences to retain in their generalization.

The following notions are based on their counterparts from [30]:

Definition 5.1 (Alignment)

Let $\tilde{s} = \lceil s_1, \dots, s_n \rceil$ and $\tilde{t} = \lceil t_1, \dots, t_m \rceil$ be two finite sequences of terms. A sequence of the form $\lceil a_1[i_1, j_1], \dots, a_k[i_k, j_k] \rceil$ is called an alignment between s and t if

- $[i_l, j_l] \in \{1, \dots, n\} \times \{1, \dots, m\}$ for $1 \leq l \leq k$,
- $i_1 < \dots < i_k$ and $j_1 < \dots < j_k$, and
- $a_l = \text{head}(\tilde{s}|_{i_l}) = \text{head}(\tilde{t}|_{j_l})$ for $1 \leq l \leq k$.

Definition 5.2 (Rigidity Function)

A binary function \mathcal{R} that, given two sequences of terms \tilde{s} and \tilde{t} , returns a set of alignments between \tilde{s} and \tilde{t} , is called a rigidity function.

Definition 5.3 (\mathcal{R} -Generalization)

Let \tilde{s} and \tilde{t} be two variable disjoint sequences and \mathcal{R} a rigidity function. An \mathcal{R} -generalization of \tilde{s} and \tilde{t} is a sequence \tilde{r} which generalizes both \tilde{s} and \tilde{t} in such a way that either $\mathcal{R}(\tilde{s}, \tilde{t}) = \emptyset$ and $\tilde{r} \in \mathcal{V}_S$, or there is an alignment $\lceil f_1[i_1, j_1], \dots, f_k[i_k, j_k] \rceil \in \mathcal{R}(\tilde{s}, \tilde{t})$ such that the following hold:

1. no successive sequence variables occur in \tilde{r} ,
2. after removing all sequence variables from \tilde{r} , the remaining sequence would have the form $\lceil f_1(\tilde{r}_1), \dots, f_k(\tilde{r}_k) \rceil$,
3. there exist two sequences \tilde{s}' and \tilde{t}' such that \tilde{r}_l is a generalization of \tilde{s}' and \tilde{t}' , $\tilde{s}|_{i_l} = f_l(\tilde{s}')$ and $\tilde{t}|_{j_l} = f_l(\tilde{t}')$, for $1 \leq l \leq k$.

Definition 5.4 (Minimal complete set of \mathcal{R} -generalizations)

Given a rigidity function \mathcal{R} , a complete set of \mathcal{R} -generalizations of two variable-disjoint sequences \tilde{s} and \tilde{t} is a set \mathcal{G} of sequences that satisfies the properties:

- Each $\tilde{r} \in \mathcal{G}$ is an \mathcal{R} -generalization of both \tilde{s} and \tilde{t} .
- For each \mathcal{R} -generalization \tilde{q} of \tilde{s} and \tilde{t} , there exists $\tilde{r} \in \mathcal{G}$ such that $\tilde{q} \preceq \tilde{r}$.

Moreover, \mathcal{G} is a minimal if no two distinct elements of \mathcal{G} are \preceq -comparable.

The algorithm $\mathfrak{G}(\mathcal{R})$ for solving syntactic generalization problems with regards to a rigidity function \mathcal{R} is based on the set of rules, introduced in [30], with slight modifications to accommodate our notions.

 \mathcal{RD} -A: \mathcal{R} -Rigid Decomposition

$$\begin{aligned}
 & \langle \{\bar{x} : \tilde{s} \triangleq^? \tilde{t}\} \cup \Gamma; S; \sigma \rangle \implies \\
 & \langle \{\bar{z}_k : \tilde{s}_k \triangleq^? \tilde{t}_k \mid 1 \leq k \leq n\} \cup \Gamma; \\
 & \{\bar{y}_0 : \tilde{s}|_0^{i_1} \triangleq \tilde{t}|_0^{j_1}\} \cup \{\bar{y}_k : \tilde{s}|_{i_k}^{i_{k+1}} \triangleq \tilde{t}|_{j_k}^{j_{k+1}} \mid 1 \leq k \leq n-1\} \cup \{\bar{y}_n : \tilde{s}|_{i_n}^{|\tilde{s}|+1} \triangleq \tilde{t}|_{j_n}^{|\tilde{t}|+1}\} \cup S; \\
 & \sigma\{\bar{x} \mapsto \lceil \bar{y}_0, f_1(\bar{z}_1), \bar{y}_1, \dots, \bar{y}_{n-1}, f_n(\bar{z}_n), \bar{y}_n \rceil \}, \\
 & \text{if } \mathcal{R}(\tilde{s}, \tilde{t}) \text{ contains a sequence } \lceil f_1[i_1, j_1], \dots, f_n[i_n, j_n] \rceil \text{ such that for all } 1 \leq k \leq n, \\
 & \tilde{s}|_{i_k} = f_k(\tilde{s}_k), \tilde{t}|_{j_k} = f_k(\tilde{t}_k), \text{ and } \bar{y}_0, \bar{y}_k\text{'s and } \bar{z}_k\text{'s are fresh.}
 \end{aligned}$$

\mathcal{RS} -A: \mathcal{R} -Rigid Solve

$$\langle \{\bar{x} : \tilde{s} \triangleq^? \tilde{t}\} \cup \Gamma; S; \sigma \rangle \Longrightarrow \langle \Gamma; \{\bar{x} : \tilde{s} \triangleq \tilde{t}\} \cup S; \sigma \rangle,$$

if $\mathcal{R}(\tilde{s}, \tilde{t}) = \emptyset$.

 \mathcal{RM} -A: \mathcal{R} -Merge

$$\langle \Gamma; \{\bar{x}_1 : \tilde{s} \triangleq \tilde{t}, \bar{x}_2 : \tilde{s} \triangleq \tilde{t}\} \cup S; \sigma \rangle \Longrightarrow \langle \Gamma; \{\bar{x}_1 : \tilde{s} \triangleq \tilde{t}\} \cup S; \sigma\{\bar{x}_2 \mapsto \bar{x}_1\} \rangle,$$

if $\bar{x}_1 \neq \bar{x}_2$.

 \mathcal{RCS} -A: \mathcal{R} -Rigid Clean Store

$$\langle \Gamma; \{\bar{x} : \ulcorner \triangleq \urcorner \} \cup S; \sigma \rangle \Longrightarrow \langle \Gamma; S; \sigma\{\bar{x} \mapsto \ulcorner \} \rangle.$$

This algorithm works similarly to \mathfrak{G} : To generalize \tilde{s} and \tilde{t} , it starts with the triple $\langle \{\bar{v} : \tilde{s} \triangleq^? \tilde{t}\}; \emptyset; \varepsilon \rangle$, where \bar{v} is a new sequence variable, applies the rules in all possible ways to the selected AUT, and ends in a configuration $\langle \emptyset; S; \sigma \rangle$. The rule \mathcal{RD} -A is the one rule causing branching, as it can be applied to all alignments returned from applying \mathcal{R} to the chosen pair. The algorithm computes a finite complete but not necessarily minimal set of \mathcal{R} -generalizations, which can be further minimized using an unranked matching algorithm.

Since the \mathcal{RD} -A rule can look quite confusing, the following example inspired by Example 4.7 in [30] illustrates how it works:

Example 5.1. Let \mathcal{R} be a rigidity function computing the longest common subsequences. Given the triple $\langle \{\bar{x}_1 : \ulcorner f(a), f(b), g(h(a), h(b)) \urcorner \triangleq^? \ulcorner f(a), g(h(a), h) \urcorner \}; \emptyset; \varepsilon \rangle$ we want to use \mathcal{R} . First we get $\mathcal{R}(\ulcorner f(a), f(b), g(h(a), h(b)) \urcorner, \ulcorner f(a), g(h(a), h) \urcorner) = \{\ulcorner f[1, 1], g[3, 2] \urcorner, \ulcorner f[2, 1], g[3, 2] \urcorner\}$. Using alignment $\ulcorner f[1, 1]g[3, 2] \urcorner$ we get via \mathcal{RD} -A

$$\langle \{\bar{z}_1 : a \triangleq^? a, \bar{z}_2 : \ulcorner h(a), h(b) \urcorner \triangleq^? \ulcorner h(a), h \urcorner\}; \{\bar{y}_1 : f(b) \triangleq^? \ulcorner \}; \{\bar{x}_1 \mapsto \ulcorner f(\bar{z}_1), \bar{y}_1, g(\bar{z}_2) \urcorner\} \rangle,$$

and using alignment $\ulcorner f[2, 1]g[3, 2] \urcorner$ we get

$$\langle \{\bar{z}_1 : b \triangleq^? a, \bar{z}_2 : \ulcorner h(a), h(b) \urcorner \triangleq^? \ulcorner h(a), h \urcorner\}; \{\bar{y}_0 : f(a) \triangleq^? \ulcorner \}; \{\bar{x}_1 \mapsto \ulcorner \bar{y}_0, f(\bar{z}_1), g(\bar{z}_2) \urcorner\} \rangle.$$

A rudimentary way of describing how this rule works is by saying that it keeps the arguments of the subterms which have found an alignment partner in Γ and moves the ‘lonely’ terms into the store S . Since the sequences which we want to generalize are variable disjoint, we can be certain that the alignments returned by the rigidity function \mathcal{R} do not contain variables. By constructing the anti-unifier σ the way we do, we can be sure that Condition 1: *no successive sequence variables occur in \tilde{r}* , is fulfilled.

5.1 Fuzzy Unranked Anti-unification with Similarity Relations

In order to modify the algorithm to work under consideration of a similarity relation, we first have to adapt the notions introduced earlier which are needed to formulate the procedure and prove its properties.

Definition 5.5 ((\mathcal{R}, λ)-Alignment)

Let $\tilde{s} = \lceil s_1, \dots, s_n \rceil$ and $\tilde{t} = \lceil t_1, \dots, t_m \rceil$ be two finite sequences of terms, \mathcal{R} a similarity relation and λ a cut value. A sequence of the form $\lceil a_1[i_1, j_1], \dots, a_k[i_k, j_k] \rceil$ is called an (\mathcal{R}, λ)-alignment between \tilde{s} and \tilde{t} if

- $[i_l, j_l] \in \{1, \dots, n\} \times \{1, \dots, m\}$ for $1 \leq l \leq k$,
- $i_1 < \dots < i_k$ and $j_1 < \dots < j_k$, and
- $a_l \simeq_{\mathcal{R}, \lambda} \text{head}(\tilde{s}|_{i_l})$ and $a_l \simeq_{\mathcal{R}, \lambda} \text{head}(\tilde{t}|_{j_l})$ for $1 \leq l \leq k$.

The degrees of an (\mathcal{R}, λ)-alignment is defined as

$$\text{Deg}(\lceil a_1[i_1, j_1] \dots a_k[i_k, j_k] \rceil, \tilde{s}, \tilde{t}, \mathcal{R}) = \left(\bigwedge_{l=1}^k \mathcal{R}(a_l, \text{head}(\tilde{s}|_{i_l})), \bigwedge_{l=1}^k \mathcal{R}(a_l, \text{head}(\tilde{t}|_{j_l})) \right).$$

Definition 5.6 ((\mathcal{R}, λ)-Rigidity Function)

Let \mathcal{R} be a similarity relation and λ a cut value. A binary function $\mathcal{R}_{\mathcal{R}, \lambda}$ that, given two sequences of terms \tilde{s} and \tilde{t} , returns a set of (\mathcal{R}, λ)-alignments between \tilde{s} and \tilde{t} , is called an (\mathcal{R}, λ)-rigidity function.

Definition 5.7 ($\mathcal{R}_{\mathcal{R}, \lambda}$ -Generalization)

Let \tilde{s} and \tilde{t} be two variable disjoint sequences, \mathcal{R} a rigidity function, \mathcal{R} a similarity relation and λ a cut value. An $\mathcal{R}_{\mathcal{R}, \lambda}$ -generalization of \tilde{s} and \tilde{t} is a sequence \tilde{r} which generalizes both \tilde{s} and \tilde{t} under consideration of $\mathcal{R}_{\mathcal{R}, \lambda}$ in such a way that either $\mathcal{R}_{\mathcal{R}, \lambda}(\tilde{s}, \tilde{t}) = \emptyset$ and $\tilde{r} \in \mathcal{V}_S$, or there is an alignment $\lceil f_1[i_1, j_1], \dots, f_k[i_k, j_k] \rceil \in \mathcal{R}_{\mathcal{R}, \lambda}(\tilde{s}, \tilde{t})$ such that the following hold:

1. no successive sequence variables occur in \tilde{r} ,
2. after removing all sequence variables from \tilde{r} , the remaining sequence would have the form $\lceil f_1(\tilde{r}_1), \dots, f_k(\tilde{r}_k) \rceil$,
3. there exist two sequences \tilde{s}' and \tilde{t}' such that \tilde{r}_l is a generalization of \tilde{s}' and \tilde{t}' , $\tilde{s}|_{i_l} = f_l(\tilde{s}')$ and $\tilde{t}|_{j_l} = f_l(\tilde{t}')$, for $1 \leq l \leq k$.

We also say that \tilde{r} is an $\mathcal{R}_{\mathcal{R}, \lambda, \alpha, \beta}$ -generalization of \tilde{s} and \tilde{t} , if there exist substitutions ν_1 and ν_2 such that $\mathcal{R}(\tilde{r}\nu_1, \tilde{s}) = \alpha \geq \lambda$ and $\mathcal{R}(\tilde{r}\nu_2, \tilde{t}) = \beta \geq \lambda$ (in addition to the other conditions of being an $\mathcal{R}_{\mathcal{R}, \lambda}$ -generalization).

Definition 5.8

Given a similarity relation \mathcal{R} , a cut value λ , and a rigidity function \mathcal{R} , a substitution σ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -anti-unifier of the $\mathcal{R}_{\mathcal{R},\lambda}$ -anti-unification problem Γ , if for each $\bar{v} : \tilde{s} \hat{\simeq}_{\mathcal{R},\lambda}^? \tilde{t} \in \Gamma$, we have $\bar{v}\sigma$ as an $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization of both \tilde{s} and \tilde{t} .

The set of all $\mathcal{R}_{\mathcal{R},\lambda}$ -anti-unifiers of Γ is denoted by $\mathcal{A}(\Gamma, \mathcal{R}_{\mathcal{R},\lambda})$.

Moreover, we write that $\sigma \in \mathcal{A}(\Gamma, \mathcal{R}_{\mathcal{R},\lambda})_{\alpha,\beta}$ if $\bar{v}\sigma$ is an $\mathcal{R}_{\mathcal{R},\lambda,\alpha,\beta}$ -generalization for both \tilde{s} and \tilde{t} for all $\bar{v} : \tilde{s} \hat{\simeq}_{\mathcal{R},\lambda}^? \tilde{t} \in \Gamma$.

Definition 5.9 (Minimal complete set of $\mathcal{R}_{\mathcal{R},\lambda}$ -generalizations for similarity relations)

Given a similarity relation \mathcal{R} , a cut value λ , and a rigidity function \mathcal{R} , a complete set of $\mathcal{R}_{\mathcal{R},\lambda}$ -generalizations of two variable-disjoint sequences \tilde{s} and \tilde{t} is a set \mathcal{G} of sequences that satisfies the properties:

- Each $\tilde{r} \in \mathcal{G}$ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization of both \tilde{s} and \tilde{t} .
- For each $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization \tilde{q} of \tilde{s} and \tilde{t} , there exists $\tilde{r} \in \mathcal{G}$ such that $\tilde{q} \lesssim_{\mathcal{R},\lambda} \tilde{r}$.

Moreover, \mathcal{G} is a minimal if no two distinct elements of \mathcal{G} are $\lesssim_{\mathcal{R},\lambda}$ -comparable.

Another notion that we will need is the notion of *substitutions generated from a set of AUTs*: Let S be a set of AUTs $S = \{\bar{v}_1 : \tilde{s}_1 \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}_1, \dots, \bar{v}_n : \tilde{s}_n \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}_n\}$. Then a pair of substitutions generated from S is defined as

$$\Sigma_{\text{gen}}(S) := (\{\bar{v}_1 \mapsto \tilde{s}_1, \dots, \bar{v}_n \mapsto \tilde{s}_n\}, \{\bar{v}_1 \mapsto \tilde{t}_1, \dots, \bar{v}_n \mapsto \tilde{t}_n\}).$$

Now we are able to rework the rules for rigid generalization given in [30] to work with a similarity relation \mathcal{R} and a cut value λ . This time we are going to keep track of two similarity degrees, one for each side of our anti-unification problem. This means our system takes the form of $\langle \Gamma; S; \sigma; \alpha; \beta \rangle$ with Γ being a set of anti-unification problems of the form $\{\bar{v}_1 : \tilde{s}_1 \hat{\simeq}^? \tilde{q}_1, \dots, \bar{v}_n : \tilde{s}_n \hat{\simeq}^? \tilde{t}_n\}$, with each generalization variable \bar{v}_i being unique and the sequences to be generalized being variable disjoint, S being a store of AUT's in solved form, σ being the anti-unifier computed so far, α being the degree for the left hand sides of the AUT's and β being the degree for the right hand sides of the AUT's.

 $\mathcal{R}_{\mathcal{R},\lambda}$ D-A_{sim}: $\mathcal{R}_{\mathcal{R},\lambda}$ -Rigid Decomposition

$$\begin{aligned} & \langle \{\bar{x} : \tilde{s} \hat{\simeq}_{\mathcal{R},\lambda}^? \tilde{t}\} \cup \Gamma; S; \sigma; \alpha; \beta \rangle \Longrightarrow \\ & \langle \{\bar{z}_k : \tilde{s}_k \hat{\simeq}_{\mathcal{R},\lambda}^? \tilde{t}_k \mid 1 \leq k \leq n\} \cup \Gamma; \\ & \quad \{\bar{y}_0 : \tilde{s}|_0^{i_1} \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}|_0^{j_1}\} \cup \{\bar{y}_k : \tilde{s}|_{i_k}^{i_{k+1}} \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}|_{j_k}^{j_{k+1}} \mid 1 \leq k \leq n-1\} \cup \\ & \quad \{\bar{y}_n : \tilde{s}|_{i_n}^{|\tilde{s}|+1} \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}|_{j_n}^{|\tilde{t}|+1}\} \cup S; \\ & \quad \sigma\{\bar{x} \mapsto \ulcorner \bar{y}_0, f_1(\bar{z}_1), \bar{y}_1, \dots, \bar{y}_{n-1}, f_n(\bar{z}_n), \bar{y}_n \urcorner\}; \alpha \wedge \gamma_1; \beta \wedge \gamma_2 \rangle, \\ & \text{if } \mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t}) \text{ contains a sequence } \ulcorner f_1[i_1, j_1], \dots, f_n[i_n, j_n] \urcorner \text{ such that for all} \end{aligned}$$

$1 \leq k \leq n$, $\tilde{s}|_{i_k} \simeq_{\mathcal{R},\lambda} f_k(\tilde{s}_k)$, $\tilde{t}|_{j_k} \simeq_{\mathcal{R},\lambda} f_k(\tilde{t}_k)$, and \bar{y}_0, \bar{y}_k 's and \bar{z}_k 's are fresh and $\text{Deg}(\ulcorner f_1[i_1, j_1], \dots, f_n[i_n, j_n] \urcorner, \tilde{s}, \tilde{t}, \mathcal{R}) = (\gamma_1, \gamma_2)$ with $\gamma_1 \geq \lambda$ and $\gamma_2 \geq \lambda$.

$\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}}$: $\mathcal{R}_{\mathcal{R},\lambda}$ -**Rigid Solve**

$\langle \{\bar{x} : \tilde{s} \hat{\simeq}_{\mathcal{R},\lambda}^? \tilde{t}\} \cup \Gamma; S; \sigma; \alpha; \beta \rangle \Longrightarrow \langle \Gamma; \{\bar{x} : \tilde{s} \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}\} \cup S; \sigma; \alpha; \beta \rangle$,
 if $\mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t}) = \emptyset$.

$\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}$: $\mathcal{R}_{\mathcal{R},\lambda}$ -**Merge**

$\langle \Gamma; \{\bar{x}_1 : \tilde{s}_1 \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}_1, \bar{x}_2 : \tilde{s}_2 \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}_2\} \cup S; \sigma; \alpha; \beta \rangle \Longrightarrow$
 $\langle \Gamma; \{\bar{x}_1 : \tilde{s}_1 \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}_1\} \cup S; \sigma\{\bar{x}_2 \mapsto \bar{x}_1\}; \alpha \wedge \gamma_1; \beta \wedge \gamma_2 \rangle$,
 if $\bar{x}_1 \neq \bar{x}_2$, $\mathcal{R}(\tilde{s}_1, \tilde{s}_2) = \gamma_1 \geq \lambda$ and $\mathcal{R}(\tilde{t}_1, \tilde{t}_2) = \gamma_2 \geq \lambda$.

$\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$: $\mathcal{R}_{\mathcal{R},\lambda}$ -**Rigid Clean Store**

$\langle \Gamma; \{\bar{x} : \ulcorner \hat{\simeq}_{\mathcal{R},\lambda} \urcorner\} \cup S; \sigma; \alpha; \beta \rangle \Longrightarrow \langle \Gamma; S; \sigma\{\bar{x} \mapsto \ulcorner \urcorner\}; \alpha; \beta \rangle$.

Wanting to generalize the sequences of terms \tilde{s} and \tilde{t} , the algorithm starts with the configuration of $\langle \{\bar{x} : \tilde{s} \hat{\simeq}_{\mathcal{R},\lambda}^? \tilde{t}\}; \emptyset; \varepsilon; 1; 1 \rangle$ with \mathcal{R} being a similarity relation and $\lambda \in (0, 1]$ a cut value. After applying the rules until no longer possible, the procedure ends with $\langle \emptyset; S; \sigma; \alpha; \beta \rangle$, such that $\bar{x}\sigma$ is a generalization of \tilde{s} and \tilde{t} and we can generate substitutions (μ_1, μ_2) from S (i.e., $\Sigma_{\text{gen}}(S) = (\mu_1, \mu_2)$) such that $\mathcal{R}(\bar{x}\sigma\mu_1, \tilde{s}) = \alpha \geq \lambda$ and $\mathcal{R}(\bar{x}\sigma\mu_2, \tilde{t}) = \beta \geq \lambda$. Note that the rule $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}$ can cause branching due to having multiple alignments available.

We will now show a short example on how $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ works:

Example 5.2. Let \mathcal{R} be a similarity relation defined as

$$\begin{aligned} \mathcal{R}(f, g) = 0.9, \quad \mathcal{R}(a, b) = 0.8, \quad \mathcal{R}(p, q) = 0.7, \\ \mathcal{R}(b, c) = 0.9, \quad \mathcal{R}(q, r) = 0.7, \\ \mathcal{R}(a, c) = 0.8, \quad \mathcal{R}(p, r) = 0.9. \end{aligned}$$

Let also $\lambda = 0.7$ be a cut value and \mathcal{R} be an (\mathcal{R}, λ) -rigidity function returning the longest common subsequences. Wanting to generalize

$$\tilde{s} = \ulcorner f(a, p(b, c), r(c, a)), q(p(a, b, c)), p(a, b) \urcorner \text{ and } \tilde{t} = \ulcorner g(b, q(c, a)), q(r(a, b)) \urcorner,$$

we set $\Gamma = \{\bar{x} : \ulcorner f(a, p(b, c), r(c, a)), q(p(a, b, c)), p(a, b) \urcorner \hat{\simeq}_{\mathcal{R},\lambda}^? \ulcorner g(b, q(c, a)), q(r(a, b)) \urcorner\}$. The following shows one branch of the procedure:

$$\begin{aligned} & \langle \Gamma; \emptyset; \varepsilon; 1; 1 \rangle \\ \xrightarrow{\text{D-A}_{\text{sim}}} & \langle \{\bar{z}_1 : \ulcorner a, p(b, c), r(c, a) \urcorner \hat{\simeq}_{\mathcal{R},\lambda}^? \ulcorner b, q(c, a) \urcorner, \bar{z}_2 : p(a, b, c) \hat{\simeq}_{\mathcal{R},\lambda}^? r(a, b)\}; \\ & \{\bar{z}_3 : p(a, b) \hat{\simeq}_{\mathcal{R},\lambda} \ulcorner \urcorner\}; \{\bar{x} \mapsto \ulcorner g(\bar{z}_1), q(\bar{z}_2), \bar{z}_3 \urcorner\}; 0.9; 1 \rangle, \end{aligned}$$

$$\begin{aligned}
 & \text{for } \lceil g[1, 1], q[2, 2] \rceil \in \mathcal{R}_{\mathcal{R}, \lambda}(\tilde{s}, \tilde{t}) \\
 \xrightarrow{\text{D-A}_{\text{sim}}} & \langle \{\bar{y}_1 : \lceil \lceil \hat{\simeq}_{\mathcal{R}, \lambda}^? \rceil \rceil, \bar{y}_3 : \lceil c, a \rceil \hat{\simeq}_{\mathcal{R}, \lambda}^? \lceil c, a \rceil, \bar{z}_2 : p(a, b, c) \hat{\simeq}_{\mathcal{R}, \lambda}^? r(a, b) \rceil\}; \\
 & \quad \{\bar{y}_2 : p(b, c) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil, \bar{z}_3 : p(a, b) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil\}; \\
 & \quad \{\bar{x} \mapsto \lceil g(a(\bar{y}_1), \bar{y}_2, p(\bar{y}_3)), q(\bar{z}_2), \bar{z}_3 \rceil\}; 0.9; 0.7), \\
 & \text{for } \lceil a[1, 1], p[3, 2] \rceil \in \mathcal{R}_{\mathcal{R}, \lambda}(\lceil a, p(b, c), r(c, a) \rceil, \lceil b, q(c, a) \rceil) \\
 \xrightarrow{\text{S-A}_{\text{sim}}} & \langle \{\bar{y}_3 : \lceil c, a \rceil \hat{\simeq}_{\mathcal{R}, \lambda}^? \lceil c, a \rceil, \bar{z}_2 : p(a, b, c) \hat{\simeq}_{\mathcal{R}, \lambda}^? r(a, b) \rceil\}; \\
 & \quad \{\bar{y}_1 : \lceil \lceil \hat{\simeq}_{\mathcal{R}, \lambda}^? \rceil \rceil, \bar{y}_2 : p(b, c) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil, \bar{z}_3 : p(a, b) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil\}; \\
 & \quad \{\bar{x} \mapsto \lceil g(a(\bar{y}_1), \bar{y}_2, p(\bar{y}_3)), q(\bar{z}_2), \bar{z}_3 \rceil\}; 0.9; 0.7) \\
 \xrightarrow{\text{CS-A}_{\text{sim}}} & \langle \{\bar{y}_3 : \lceil c, a \rceil \hat{\simeq}_{\mathcal{R}, \lambda}^? \lceil c, a \rceil, \bar{z}_2 : p(a, b, c) \hat{\simeq}_{\mathcal{R}, \lambda}^? r(a, b) \rceil\}; \\
 & \quad \{\bar{y}_2 : p(b, c) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil, \bar{z}_3 : p(a, b) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil\}; \\
 & \quad \{\bar{x} \mapsto \lceil g(a, \bar{y}_2, p(\bar{y}_3)), q(\bar{z}_2), \bar{z}_3 \rceil\}; 0.9; 0.7) \\
 \xrightarrow{\text{D}^*\text{-A}_{\text{sim}}} & \langle \{\bar{z}_2 : p(a, b, c) \hat{\simeq}_{\mathcal{R}, \lambda}^? r(a, b) \rceil\}; \\
 & \quad \{\bar{y}_2 : p(b, c) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil, \bar{z}_3 : p(a, b) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil\}; \\
 & \quad \{\bar{x} \mapsto \lceil g(a, \bar{y}_2, p(c, a)), q(\bar{z}_2), \bar{z}_3 \rceil\}; 0.9; 0.7), \\
 & \text{for } \lceil c[1, 1], a[2, 2] \rceil \in \mathcal{R}_{\mathcal{R}, \lambda}(\lceil c, a \rceil, \lceil c, a \rceil) \\
 \xrightarrow{\text{D-A}_{\text{sim}}} & \langle \{\bar{v} : \lceil a, b, c \rceil \hat{\simeq}_{\mathcal{R}, \lambda}^? \lceil a, b \rceil\}; \\
 & \quad \{\bar{y}_2 : p(b, c) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil, \bar{z}_3 : p(a, b) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil\}; \\
 & \quad \{\bar{x} \mapsto \lceil g(a, \bar{y}_2, p(c, a)), q(r(\bar{v})), \bar{z}_3 \rceil\}; 0.9; 0.7), \\
 & \text{for } r[1, 1] \in \mathcal{R}_{\mathcal{R}, \lambda}(p(a, b, c), r(a, b)) \\
 \xrightarrow{\text{D-S-CS-A}_{\text{sim}}} & \langle \emptyset; \{\bar{v}_3 : c \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil, \bar{y}_2 : p(b, c) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil, \bar{z}_3 : p(a, b) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil\}; \\
 & \quad \{\bar{x} \mapsto \lceil g(a, \bar{y}_2, p(c, a)), q(r(a, b, \bar{v}_3)), \bar{z}_3 \rceil\}; 0.9; 0.7), \\
 & \text{for } \lceil a[1, 1], b[2, 2] \rceil \in \mathcal{R}_{\mathcal{R}, \lambda}(\lceil a, b, c \rceil, \lceil a, b \rceil) \\
 \xrightarrow{\text{M-A}_{\text{sim}}} & \langle \emptyset; \{\bar{v}_3 : c \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil, \bar{y}_2 : p(b, c) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil\}; \\
 & \quad \{\bar{x} \mapsto \lceil g(a, \bar{y}_2, p(c, a)), q(r(a, b, \bar{v}_3)), \bar{y}_2 \rceil\}; 0.8; 0.7)
 \end{aligned}$$

Note that $\text{D-S-CS-A}_{\text{sim}}$ denotes the contracted sequence of steps of D-A_{sim} , S-A_{sim} and CS-A_{sim} . We use this, since we have already shown how this sequence of steps works and we want to shorten this example in order to improve readability.

Now we can see that $\lceil g(a, \bar{y}_2, p(c, a)), q(r(a, b, \bar{v}_3)), \bar{y}_2 \rceil$ is a generalization of \tilde{s} and \tilde{t} . From

$$S = \{\bar{v}_3 : c \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil, \bar{y}_2 : p(b, c) \hat{\simeq}_{\mathcal{R}, \lambda} \lceil \lceil \rceil\}$$

we can extract $\Sigma_{\text{gen}}(S) = (\mu_1, \mu_2)$ with $\mu_1 = \{\bar{v}_3 \mapsto c, \bar{y}_2 \mapsto p(b, c)\}$ and $\mu_2 = \{\bar{v}_3 \mapsto \ulcorner, \bar{y}_2 \mapsto \urcorner\}$ such that

$$\begin{aligned}
 & \mathcal{R}(\bar{x}\sigma\mu_1, \tilde{s}) \\
 &= \mathcal{R}(\ulcorner g(a, p(b, c), p(c, a)), q(r(a, b, c)), \bar{y}_2 \urcorner, \ulcorner f(a, p(b, c), r(c, a)), q(p(a, b, c)), p(a, b) \urcorner) \\
 &= 0.8
 \end{aligned}$$

and

$$\mathcal{R}(\bar{x}\sigma\mu_2, \tilde{t}) = \mathcal{R}(\ulcorner g(a, \bar{y}_2, p(c, a)), q(r(a, b)) \urcorner, \ulcorner g(b, q(c, a)), q(r(a, b)) \urcorner) = 0.7.$$

Theorem 5.1 (Termination of $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R}, \lambda})$)

The procedure $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R}, \lambda})$ terminates for every input.

Proof. In order to show that the procedure terminates, we create a complexity measurement $(M(\Gamma), M(S))$ with $M(\Gamma) := \{\text{size}(\tilde{s} \hat{\simeq}_{\mathcal{R}, \lambda}^? \tilde{t}) + 1 \mid \bar{x} : \tilde{s} \hat{\simeq}_{\mathcal{R}, \lambda}^? \tilde{t} \in \Gamma\}$ and $M(S) := \{\text{size}(\tilde{s} \hat{\simeq}_{\mathcal{R}, \lambda} \tilde{t}) + 1 \mid \bar{x} : \tilde{s} \hat{\simeq}_{\mathcal{R}, \lambda} \tilde{t} \in S\}$ for a given rigidity function $\mathcal{R}_{\mathcal{R}, \lambda}$, a set of $\mathcal{R}_{\mathcal{R}, \lambda}$ -AUT's Γ , the store S , the $\mathcal{R}_{\mathcal{R}, \lambda}$ -anti-unifier σ and the similarity degrees α and β . Since $M(\Gamma)$ and $M(S)$ are multisets, we will use the Dershowitz-Manna ordering $>_{DM}$. Now we have to check if each step of the procedure strictly reduces the complexity measure.

	$M(\Gamma)$	$M(S)$
$\mathcal{R}_{\mathcal{R}, \lambda}D\text{-}A_{\text{sim}}$	$>$	
$\mathcal{R}_{\mathcal{R}, \lambda}S\text{-}A_{\text{sim}}$	$>$	
$\mathcal{R}_{\mathcal{R}, \lambda}M\text{-}A_{\text{sim}}$	\geq	$>$
$\mathcal{R}_{\mathcal{R}, \lambda}CS\text{-}A_{\text{sim}}$	\geq	$>$

Therefore each transformation chain terminates. Even though the decomposition rule $\mathcal{R}_{\mathcal{R}, \lambda}D\text{-}A_{\text{sim}}$ can cause branching, only a finite amount of branches can split as the only a finite amount of alignments are returned by the rigidity function. \square

Lemma 5.2

Let $\langle \Gamma; S; \sigma; \alpha; \beta \rangle \xRightarrow{R_1} \langle \Gamma_1; S_1; \sigma\theta_1; \alpha_1; \beta_1 \rangle \xRightarrow{R_2} \langle \Gamma_2; S_2; \sigma\theta_1\theta_2; \alpha_2; \beta_2 \rangle$ be a sequence of transformations performed by $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R}, \lambda})$ where $R_1 \in \{\mathcal{R}_{\mathcal{R}, \lambda}M\text{-}A_{\text{sim}}, \mathcal{R}_{\mathcal{R}, \lambda}CS\text{-}A_{\text{sim}}\}$ and $R_2 \in \{\mathcal{R}_{\mathcal{R}, \lambda}D\text{-}A_{\text{sim}}, \mathcal{R}_{\mathcal{R}, \lambda}S\text{-}A_{\text{sim}}\}$. Then there exists a transformation sequence $\langle \Gamma; S; \sigma; \alpha; \beta \rangle \xRightarrow{R_2} \langle \Gamma'_1; S'_1; \sigma\theta_2; \alpha'_1; \beta'_1 \rangle \xRightarrow{R_1} \langle \Gamma'_2; S'_2; \sigma\theta_2\theta_1; \alpha'_2; \beta'_2 \rangle$ such that $\Gamma_2 = \Gamma'_2$, $S_2 = S'_2$, $\sigma\theta_1\theta_2 = \sigma\theta_2\theta_1$, $\alpha_2 = \alpha'_2$ and $\beta_2 = \beta'_2$.

Proof. Aside from the addition of the similarity degree, the proof follows the same structure as in the syntactic case. This means that we already know $\Gamma_2 = \Gamma'_2$, $S_2 = S'_2$, $\sigma\theta_1\theta_2 = \sigma\theta_2\theta_1$ from

[30]. What is left to show is that $\alpha_2 = \alpha'_2$ and $\beta_2 = \beta'_2$. Both R_1 and R_2 have the possibility of changing the degree depending on the specific rule chosen. This means that there are four different paths the degree evolution can take.

We start by looking at the first sequence:

- assuming we use both rules which do not change the degrees, i.e first $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$, then $\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}}$, we get that $\alpha_2 = \alpha_1 = \alpha$ and $\beta_2 = \beta_1 = \beta$,
- if we use rule $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$ first, which does not change the degrees leaving $\alpha_1 = \alpha$ and $\beta_1 = \beta$, followed by rule $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}$ which can change the degrees, we get that $\alpha_2 = \alpha_1 \wedge \gamma_1 = \alpha \wedge \gamma_1$ and $\beta_2 = \beta_1 \wedge \gamma_2 = \beta \wedge \gamma_2$,
- using rule $\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}$, which can change the degrees as $\alpha_1 = \alpha \wedge \gamma_1$ and $\beta_1 = \beta \wedge \gamma_2$, followed by $\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}}$, leaving the degrees untouched, we get $\alpha_2 = \alpha_1 = \alpha \wedge \gamma_1$ and $\beta_2 = \beta_1 = \beta \wedge \gamma_2$,
- finally using both rules which can change the degrees, i.e. $\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}$ and $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}$, we get that $\alpha_1 = \alpha \wedge \gamma_1^{\text{M}}$ and therefore $\alpha_2 = \alpha_1 \wedge \gamma_1^{\text{D}} = \alpha \wedge \gamma_1^{\text{M}} \wedge \gamma_1^{\text{D}}$, and $\beta_1 = \beta \wedge \gamma_2^{\text{M}}$ and therefore $\beta_2 = \beta_1 \wedge \gamma_2^{\text{D}} = \beta \wedge \gamma_2^{\text{M}} \wedge \gamma_2^{\text{D}}$.

In the second sequence we perform both rule R_1 and rule R_2 the exact same way as in the first one, as was already argued in [30] for showing the equality of Γ_2 and Γ'_2 . Looking at the four possible cases again, we get:

- if we used the order $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$, $\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}}$ in the first sequence, we then use $\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}}$, $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$ and therefore get $\alpha'_2 = \alpha'_1 = \alpha = \alpha_2$ and $\beta'_2 = \beta'_1 = \beta = \beta_2$,
- if we used the order $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$, $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}$ in the first sequence, we then use $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}$, $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$ and therefore get $\alpha'_2 = \alpha'_1 = \alpha \wedge \gamma_1 = \alpha_2$ and $\beta'_2 = \beta'_1 = \beta \wedge \gamma_2 = \beta_2$,
- if we used the order $\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}$, $\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}}$ in the first sequence, we then use $\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}}$, $\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}$ and therefore get $\alpha'_2 = \alpha'_1 \wedge \gamma_1 = \alpha \wedge \gamma_1 = \alpha_2$ and $\beta'_2 = \beta'_1 \wedge \gamma_2 = \beta \wedge \gamma_2 = \beta_2$,
- if we used the order $\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}$, $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}$ in the first sequence, we then use $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}$, $\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}$ and therefore get $\alpha'_2 = \alpha'_1 \wedge \beta_2 = \alpha \wedge \beta_1 \wedge \beta_2 = \alpha_2$ and $\beta'_2 = \beta'_1 \wedge \gamma_2^{\text{M}} = \beta \wedge \gamma_2^{\text{D}} \wedge \gamma_2^{\text{M}} = \beta_2$,

This means $\alpha_2 = \alpha'_2$ and $\beta_2 = \beta'_2$. □

Lemma 5.3

If $\langle \Gamma; S_1; \sigma; \alpha; \beta \rangle \Longrightarrow^* \langle \emptyset; S_2; \sigma\theta; \alpha \wedge \gamma_1; \beta \wedge \gamma_2 \rangle$ is a derivation in $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ using only the rules $(\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}})$ and $(\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}})$, then $\theta \in \mathcal{A}(\Gamma, \mathcal{R}_{\mathcal{R},\lambda})_{\gamma_1, \gamma_2}$.

Proof. This proof will be done by induction on the number of steps in the sequence of transformations. If the derivation has only one step, i.e. $\langle \{\bar{x} : \tilde{s} \stackrel{?}{\mathcal{R},\lambda} \tilde{t}\}; S; \sigma; \alpha; \beta \rangle \Longrightarrow \langle \emptyset; \{\bar{x} : \tilde{s} \stackrel{?}{\mathcal{R},\lambda} \tilde{t}\} \cup S; \sigma\theta; \alpha \wedge \gamma_1; \beta \wedge \gamma_2 \rangle$, then either $\theta = \varepsilon$, $\gamma_1 = 1$ and $\gamma_2 = 1$ if the $(\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}})$ rule is used, or $\theta = \{\bar{x} \mapsto \bar{y}_0\}$ with $\bar{y}_0 \in \mathcal{V}_S$ being fresh and $(\gamma_1, \gamma_2) = \mathcal{R}(\ulcorner f_1[i_1, j_1], \dots, f_n[i_n, j_n] \urcorner)$ with $\ulcorner f_1[i_1, j_1], \dots, f_n[i_n, j_n] \urcorner \in \mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t})$ if $(\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}})$ is used. We know that the generalization in this case is a sequence variable and since we can take $(\nu_1, \nu_2) = \Sigma_{\text{gen}}(S)$ such that $\mathcal{R}(\bar{x}\theta\nu_1, \tilde{s}) = \gamma_1$ and $\mathcal{R}(\bar{x}\theta\nu_2, \tilde{t}) = \gamma_2$, $\bar{x}\theta$ it is also an $\mathcal{R}_{\mathcal{R},\lambda,\gamma_1,\gamma_2}$ -generalization.

We assume as the induction hypothesis that this lemma holds for sequences of the length m and want to prove that it is also correct for derivations of length $m+1$. A system taking $m+1$ steps to solve shall be of the form $\langle \Gamma_{m+1}; S_{m+1}; \sigma; \alpha; \beta \rangle$.

Let $\Gamma_{m+1} = \{\bar{x} : \tilde{s} \stackrel{?}{\mathcal{R},\lambda} \tilde{t}\} \cup \Gamma'_{m+1}$ and assume that it is to be transformed by $(\mathcal{R}\text{D-A}_{\text{sim}})$. This means we get $\Gamma_m = \{\bar{z}_k : \tilde{s}_k \stackrel{?}{\mathcal{R},\lambda} \tilde{t}_k \mid 1 \leq k \leq n\} \cup \Gamma'_{m+1}$, $S_m = \{\bar{y}_0 : \tilde{s}_0^{i_1} \stackrel{?}{\mathcal{R},\lambda} \tilde{t}_0^{j_1}\} \cup \{\bar{y}_k : \tilde{s}_{i_k}^{i_{k+1}} \stackrel{?}{\mathcal{R},\lambda} \tilde{t}_{j_k}^{j_{k+1}} \mid 1 \leq k \leq n-1\} \cup \{\bar{y}_n : \tilde{s}_{i_n}^{|\tilde{s}|+1} \stackrel{?}{\mathcal{R},\lambda} \tilde{t}_{j_n}^{|\tilde{t}|+1}\} \cup S_{m+1}$, $\theta_m = \{\bar{x} \mapsto \ulcorner \bar{y}_0, f_1(\bar{z}_1), \bar{y}_1, \dots, \bar{y}_{n-1}, f_n(\bar{z}_n), \bar{y}_n \urcorner\}$ and $(\gamma_{1,m}, \gamma_{2,m}) = \mathcal{R}(\ulcorner f_1[i_1, j_1], \dots, f_n[i_n, j_n] \urcorner)$ as defined in rule $(\mathcal{R}\text{D-A}_{\text{sim}})$. Since the number of the remaining steps to solve the system $\langle \{\bar{z}_k : \tilde{s}_k \stackrel{?}{\mathcal{R},\lambda} \tilde{t}_k \mid 1 \leq k \leq n\} \cup \Gamma'_{m+1}; S_{m+1} \cup S_m; \sigma\theta_m; \alpha \wedge \gamma_{1,m}; \beta \wedge \gamma_{2,m} \rangle \Longrightarrow^* \langle \emptyset; S_{m+1} \cup \dots \cup S_0; \sigma\theta_m \dots \theta_0; \alpha \wedge \gamma_{1,m} \wedge \dots \wedge \gamma_{1,0}; \beta \wedge \gamma_{2,m} \wedge \dots \wedge \gamma_{2,0} \rangle$ is only m we know from the induction hypothesis that for $S = S_{m-1} \cup \dots \cup S_0$, $\theta = \theta_{m-1} \dots \theta_0$, $\gamma_1 = \gamma_{1,m-1} \wedge \dots \wedge \gamma_{1,0}$ and $\gamma_2 = \gamma_{2,m-1} \wedge \dots \wedge \gamma_{2,0}$, then $\theta \in \mathcal{A}(\Gamma_m, \mathcal{R}_{\mathcal{R},\lambda})_{\gamma_1, \gamma_2}$ which means that $\bar{v}\theta$ is an $\mathcal{R}_{\mathcal{R},\lambda,\gamma_1,\gamma_2}$ -generalization for both \tilde{r} and \tilde{q} for all $\{\bar{v} : \tilde{r} \stackrel{?}{\mathcal{R},\lambda} \tilde{q}\} \in \Gamma_m$ and therefore we also know that $\mathcal{R}(\bar{v}\theta\mu_1, \tilde{r}) = \gamma_1$ and $\mathcal{R}(\bar{v}\theta\mu_2, \tilde{q}) = \gamma_2$ holds for some substitutions (μ_1, μ_2) . Hence, this also holds for \bar{z}_k for $1 \leq k \leq n$ and therefore we can choose $(\nu_1, \nu_2) = (\mu_1, \mu_2) \cup \Sigma_{\text{gen}}(S_m)$ such that $\mathcal{R}(\bar{x}\theta_m\nu_1, \tilde{s}) = \gamma_1 \wedge \gamma_{1,m}$ and $\mathcal{R}(\bar{x}\theta_m\nu_2, \tilde{t}) = \gamma_1 \wedge \gamma_{2,m}$ and therefore $\bar{x}\theta_m\theta$ is an $\mathcal{R}_{\mathcal{R},\lambda,\gamma_1 \wedge \gamma_{1,m}, \gamma_2 \wedge \gamma_{2,m}}$ -generalization for \tilde{s} and \tilde{t} , i.e. $\theta_m\theta \in \mathcal{A}(\Gamma_{m+1}, \mathcal{R}_{\mathcal{R},\lambda})_{\gamma_1 \wedge \gamma_{1,m}, \gamma_2 \wedge \gamma_{2,m}}$.

Next let Γ_{m+1} be transformed by $(\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}})$, which indicates that $\mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t}) = \emptyset$. We get $\Gamma_m = \Gamma'_{m+1}$, $S_m = \{\bar{x} : \tilde{s} \stackrel{?}{\mathcal{R},\lambda} \tilde{t}\} \cup S_{m+1}$, $\theta_m = \varepsilon$ and $\gamma_{1,m} = \gamma_{2,m} = 1$. Since the number of the remaining steps to solve $\langle \Gamma_m; S_{m+1} \cup S_m; \sigma\theta_m; \alpha \wedge \gamma_{1,m}; \beta \wedge \gamma_{2,m} \rangle \Longrightarrow \langle \emptyset; S_{m+1} \cup \dots \cup S_1; \sigma\theta_m \dots \theta_0; \alpha \wedge \gamma_{1,m} \wedge \dots \wedge \gamma_{1,0}; \beta \wedge \gamma_{2,m} \wedge \dots \wedge \gamma_{2,0} \rangle$ is only m we know from our induction hypothesis that for $S = S_{m-1} \cup \dots \cup S_0$, $\theta = \theta_{m-1} \dots \theta_0$, $\gamma_1 = \gamma_{1,m} \wedge \dots \wedge \gamma_{1,0}$ and $\gamma_2 = \gamma_{2,m} \wedge \dots \wedge \gamma_{2,0}$, $\theta \in \mathcal{A}(\Gamma_m, \mathcal{R}_{\mathcal{R},\lambda})_{\gamma_1, \gamma_2}$ meaning that $\bar{v}\theta$ is an $\mathcal{R}_{\mathcal{R},\lambda,\gamma_1,\gamma_2}$ -generalization for both \tilde{r} and \tilde{q} for all $\{\bar{v} : \tilde{r} \stackrel{?}{\mathcal{R},\lambda} \tilde{q}\} \in \Gamma_m$. Hence we also know that $\mathcal{R}(\bar{v}\theta\mu_1, \tilde{r}) = \gamma_1$ and $\mathcal{R}(\bar{v}\theta\mu_2, \tilde{q}) = \gamma_2$ holds for some substitutions (μ_1, μ_2) . Since $\mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t}) = \emptyset$, $\theta_m\theta = \theta$ and we can choose $(\nu_1, \nu_2) = (\mu_1, \mu_2) \cup \Sigma(S_m)$ such that $\mathcal{R}(\bar{x}\theta\nu_1, \tilde{s}) = \gamma_1 \wedge \gamma_{1,m}$ and $\mathcal{R}(\bar{x}\theta\nu_2, \tilde{t}) = \gamma_1 \wedge \gamma_{2,m}$, we get that $\bar{x}\theta$ is an $\mathcal{R}_{\mathcal{R},\lambda,\gamma_1,\gamma_2}$ -generalization for both \tilde{s} and \tilde{t} , meaning $\theta \in \mathcal{A}(\Gamma_{m+1}, \mathcal{R}_{\mathcal{R},\lambda})_{\gamma_1, \gamma_2}$. \square

Lemma 5.4

If $\langle \{\bar{x} : \tilde{s} \stackrel{?}{\mathcal{R},\lambda} \tilde{t}\}; \emptyset; \varepsilon; 1; 1 \rangle \Longrightarrow^* \langle \emptyset; S_1; \sigma; \alpha; \beta \rangle \Longrightarrow_R \langle \emptyset; S; \sigma\theta; \alpha \wedge \gamma_1; \beta \wedge \gamma_2 \rangle$ is a derivation

in $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ such that $\bar{x}\sigma$ is an $\mathcal{R}_{\mathcal{R},\lambda,\alpha,\beta}$ -generalization of \tilde{s} and \tilde{t} and $R \in \{\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}, \mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}\}$, then $\sigma\theta \in \mathcal{A}(\Gamma, \mathcal{R}_{\mathcal{R},\lambda})_{\alpha \wedge \gamma_1, \beta \wedge \gamma_2}$ with $\Gamma = \{\bar{x} : \tilde{s} \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}\}$.

Proof. Case $R = \mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}$:

Let $S_1 = \{\bar{v}_1 : \tilde{q}_1 \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{r}_1, \bar{v}_2 : \tilde{q}_2 \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{r}_2\} \cup S'_1$ with $\mathcal{R}(\tilde{q}_1, \tilde{q}_2) = \gamma_1 \geq \lambda$ and $\mathcal{R}(\tilde{r}_1, \tilde{r}_2) = \gamma_2 \geq \lambda$. The last step of the derivation then results in $S = \{\bar{v}_1 : \tilde{q}_1 \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{r}_1\} \cup S'_1$ and $\theta = \{\bar{v}_2 \mapsto \bar{v}_1\}$. Each occurrence of the pairs \tilde{q}_1 and \tilde{r}_1 , and \tilde{q}_2 and \tilde{r}_2 in $\bar{x}\sigma$ will be represented by the same variable at each of their corresponding positions from \tilde{s} and \tilde{t} . Replacing all \bar{v}_2 's in $\bar{x}\sigma$ with \bar{v}_1 's does not change the fact that this sequence is a generalization of \tilde{s} and \tilde{t} , i.e. $\bar{x}\sigma\theta$ is also a generalization of \tilde{s} and \tilde{t} . Choosing $(\nu_1, \nu_2) = \Sigma_{\text{gen}}(S)$ it holds that $\mathcal{R}(\bar{x}\sigma\theta\nu_1, \tilde{s}) = \alpha \wedge \gamma_1$ and $\mathcal{R}(\bar{x}\sigma\theta\nu_2, \tilde{t}) = \beta \wedge \gamma_2$.

The proof of the remaining properties of $\bar{x}\sigma\theta$ being an $\mathcal{R}_{\mathcal{R},\lambda,\alpha \wedge \gamma_1, \beta \wedge \gamma_2}$ -generalization of \tilde{s} and \tilde{t} does not change from [30].

The proof for $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$ is straightforward. □

Theorem 5.5 (Soundness of $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$)

If $\langle \Gamma; \emptyset; \varepsilon; 1; 1 \rangle \Longrightarrow^* \langle \emptyset; S; \sigma; \alpha; \beta \rangle$ is a derivation in $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ with $\Gamma = \{\bar{x} : \tilde{s} \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}\}$, then $\sigma \in \mathcal{A}(\Gamma, \mathcal{R}_{\mathcal{R},\lambda})_{\alpha, \beta}$.

Proof. The idea of the proof remains identical to [30]. We know it is allowed to order the derivation in such a way that the store rules ($\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}$) and ($\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$) are only applied after the set of AUT's is emptied as proven in Lemma 5.2, leaving the computed substitutions and degrees to be equal. Let $\langle \emptyset; S'; \sigma'; \alpha'; \beta' \rangle$ be the point where the set of AUT's is empty but we have not yet applied the store rules, then we get by Lemma 5.3 that the $\bar{x}\sigma'$ is an $\mathcal{R}_{\mathcal{R},\lambda,\alpha',\beta'}$ -generalization of \tilde{s} and \tilde{t} . The changes to the substitution σ' and the degrees α' and β' by ($\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}}$) and ($\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$) resulting in σ , α and β preserve the properties of being an $\mathcal{R}_{\mathcal{R},\lambda,\alpha,\beta}$ -generalization to \tilde{s} and \tilde{t} . □

Next we want to prove completeness of $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$. Both theorem and proof are taken from [30] and modified to accommodate similarity relations and fit our notation:

Theorem 5.6 (Completeness of $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$)

Let \tilde{r} be an $\mathcal{R}_{\mathcal{R},\lambda,\alpha',\beta'}$ -generalization of \tilde{s} and \tilde{t} . Then $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ constructs a derivation $\langle \{\bar{x} : \tilde{s} \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}\}; \emptyset; \varepsilon; 1; 1 \rangle \Longrightarrow^+ \langle \emptyset; S; \sigma; \alpha; \beta \rangle$ such that σ is an $\mathcal{R}_{\mathcal{R},\lambda,\alpha,\beta}$ -anti-unifier for $\bar{x} : \tilde{s} \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}$ with $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$, $\alpha \geq \alpha' \geq \lambda$ and $\beta \geq \beta' \geq \lambda$.

Proof. The completeness of $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ will be shown via induction over the size s of \tilde{r} .

For $s = 0$ it would mean that $\tilde{r} = \ulcorner \urcorner$ and therefore $\tilde{s} = \tilde{t} = \ulcorner \urcorner$. Here $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ would first apply $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}$ followed by $\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}}$ resulting in $\sigma = \{\bar{x} \mapsto \ulcorner \urcorner\}$ and $\alpha = 1 \geq \beta \geq \lambda$.

For $s = 1$ there are several possible cases:

- if $\tilde{r} = c$, then $\ulcorner c[1, 1] \urcorner \in \mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t})$. Here the procedure can apply $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}$ followed by $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}$ giving us $\sigma = \{\bar{x} \mapsto c\}$ so that $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma = c$ and $(\alpha, \beta) = \mathcal{R}(\ulcorner c[1, 1] \urcorner)$ which is equal to (α', β') as we know that \tilde{r} is an $\mathcal{R}_{\mathcal{R},\lambda,\alpha',\beta'}$ -generalization of \tilde{s} and \tilde{t} and therefore $\mathcal{R}(c, \tilde{s}) = \alpha'$ and $\mathcal{R}(c, \tilde{t}) = \beta'$.
- if $\tilde{r} = \bar{x}$, then $\mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t}) = \emptyset$. Rule $\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{sim}}$ gets used and gives us a final configuration with $\sigma = \varepsilon$, $\alpha = 1$ and $\beta = 1$. This means $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x} = \bar{x}\sigma$, $\alpha \geq \alpha'$ and $\beta \geq \beta'$.
- if $\tilde{r} = \bar{y}_0$ with $\bar{y}_0 \neq \bar{x}$, then $\ulcorner \urcorner \in \mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t})$. Applying rule $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}$ gives $\sigma = \{\bar{x} \mapsto \bar{y}'_0\}$ with \bar{y}'_0 being a new sequence variable fulfilling the condition $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{y}'_0 = \bar{x}\sigma$; and $\alpha = 1 \geq \alpha'$ and $\beta = 1 \geq \beta'$.

Given that \tilde{r} is an $\mathcal{R}_{\mathcal{R},\lambda,\alpha',\beta'}$ -generalization of \tilde{s} and \tilde{t} we know that there exists an alignment $\ulcorner f_1[i_1, j_1], \dots, f_n[i_n, j_n] \urcorner \in \mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t})$ such that

- no successive sequence variables occur in \tilde{r} ,
- after removing all sequence variables from \tilde{r} it has the form of $\ulcorner f_1(\tilde{r}_1), \dots, f_k(\tilde{r}_k) \urcorner$,
- there exist two sequences \tilde{s}'_k and \tilde{t}'_k such that \tilde{r}_k is a generalization of \tilde{s}'_k and \tilde{t}'_k , $\tilde{s}|_{i_k} = f_k(\tilde{s}'_k)$ and $\tilde{t}|_{j_k} = f_k(\tilde{t}'_k)$, for $1 \leq k \leq n$,
- there exist two substitution μ_1 and μ_2 such that $\mathcal{R}(\tilde{r}\mu_1, \tilde{s}) = \alpha'$ and $\mathcal{R}(\tilde{r}\mu_2, \tilde{t}) = \beta'$.

For our induction hypothesis we assume that for all sequences \tilde{r}^H with $\text{size}(\tilde{r}^H) < s$ and for all sequences \tilde{s}^H and \tilde{t}^H , it holds that if \tilde{r}^H is an $\mathcal{R}_{\mathcal{R},\lambda,\alpha',\beta'}$ -generalization of \tilde{s}^H and \tilde{t}^H , then $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ constructs σ^H , an $\mathcal{R}_{\mathcal{R},\lambda,\alpha^H,\beta^H}$ -anti-unifier for $\bar{x}^H : \tilde{s}^H \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}^H$, such that $\tilde{r}^H \lesssim_{\mathcal{R},\lambda} \bar{x}^H\sigma^H$, $\alpha^H \geq \alpha' \geq \lambda$ and $\beta^H \geq \beta' \geq \lambda$. Next we prove this for sequences of size s :

Let $\tilde{r} = \ulcorner \bar{v}_0, f_1(\tilde{r}_1), \bar{v}_1, \dots, \bar{v}_{n-1}, f(\tilde{r}_n), \bar{v}_n \urcorner$. For $1 \leq k \leq n$ we have that $\text{size}(\tilde{r}_k) < \text{size}(\tilde{r})$ and therefore we know due to the induction hypothesis that there exists a derivation $\langle \{\bar{z}_k : \tilde{s}'_k \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}'_k\}; \emptyset; \varepsilon; 1; 1 \rangle \Longrightarrow^* \langle \emptyset; S_k; \sigma_k; \alpha_k; \beta_k \rangle$ such that $\tilde{r}_k \lesssim_{\mathcal{R},\lambda} \bar{z}_k\sigma_k$, $\alpha_k \geq \alpha' \geq \lambda$ and $\beta_k \geq \beta' \geq \lambda$. Putting a decomposition step from the initial configuration before this derivation gives us $\langle \{\tilde{s} \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}\}; \emptyset; \varepsilon; 1; 1 \rangle \Longrightarrow_{\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{sim}}} \langle \{\bar{z}_k : \tilde{s}'_k \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}'_k\}; S^D; \sigma^D; \alpha^D; \beta^D \rangle \Longrightarrow^* \langle \emptyset; S^D \cup S_1 \cup \dots \cup S_n; \sigma^D\sigma_1 \dots \sigma_n; \alpha^D \wedge \alpha_1 \wedge \dots \wedge \alpha_n; \beta^D \wedge \beta_1 \wedge \dots \wedge \beta_n \rangle$ with $\alpha^D \geq \alpha'$ and $\beta^D \geq \beta'$. Using our proof of soundness in Theorem 5.5, it follows that $\bar{x}\sigma^D\sigma_1 \dots \sigma_n$ is an $\mathcal{R}_{\mathcal{R},\lambda,\alpha^*,\beta^*}$ -generalization of \tilde{s} and \tilde{t} with $\alpha^* = \alpha^D \wedge \alpha_1 \wedge \dots \wedge \alpha_n \geq \alpha'$ and $\beta^* = \beta^D \wedge \beta_1 \wedge \dots \wedge \beta_n \geq \beta'$. It is known that $\bar{z}_k\sigma_k = \bar{z}_k\sigma_1 \dots \sigma_n$, since all \bar{z}_k 's for $1 \leq k \leq n$ are unique.

Next, we use the store cleaning rule in order to get our optimal $\mathcal{R}_{\mathcal{R},\lambda}$ -anti-unifier. This means our final derivation looks as follows: $\langle \{\tilde{s} \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}\}; \emptyset; \varepsilon; 1; 1 \rangle \Longrightarrow_{\mathcal{R}_{\mathcal{R},\lambda}\text{-A}_{\text{sim}}} \langle \{\bar{z}_k : \tilde{s}'_k \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}'_k\}; S^D; \sigma^D; \alpha^D; \beta^D \rangle \Longrightarrow^* \langle \emptyset; S^D \cup S_1 \cup \dots \cup S_n; \sigma^D \sigma_1 \dots \sigma_n; \alpha^D \wedge \alpha_1 \wedge \dots \wedge \alpha_n; \beta^D \wedge \beta_1 \wedge \dots \wedge \beta_n \rangle \Longrightarrow^*_{\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{sim}} | \mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{sim}}} \langle \emptyset; S; \sigma; \alpha^D \wedge \alpha_1 \wedge \dots \wedge \alpha_n \wedge \alpha^S; \beta^D \wedge \beta_1 \wedge \dots \wedge \beta_n \wedge \beta^S \rangle$. We again know from the soundness theorem that $\bar{x}\sigma$ is an $\mathcal{R}_{\mathcal{R},\lambda,\alpha,\beta}$ -generalization for $\alpha = \alpha^D \wedge \alpha_1 \wedge \dots \wedge \alpha_n \wedge \alpha^S \geq \alpha'$ and $\beta = \beta^D \wedge \beta_1 \wedge \dots \wedge \beta_n \wedge \beta^S \geq \beta'$ since $\alpha^S \geq \alpha'$ and $\beta^S \geq \beta'$.

In the case that \tilde{r} only contains unique variables, it holds that $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma^D \sigma_1 \dots \sigma_n \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$.

However, if \tilde{r} contains duplicate variables it is a bit more complex:

- In the case that a variable \bar{y} occurs twice in \tilde{r} , such that $\bar{y} = \bar{v}_m = \bar{v}_k$ for some $0 \leq m, k \leq n$ with $m \neq k$, then we know that $\mathcal{R}(\tilde{s}_1|_{i_m}^{i_{m+1}}, \tilde{s}_2|_{i_k}^{i_{k+1}}) = \alpha^S \geq \lambda$ and $\mathcal{R}(\tilde{t}_1|_{j_m}^{j_{m+1}}, \tilde{t}_2|_{j_k}^{j_{k+1}}) = \beta^S \geq \lambda$. Two new AUT's, $\bar{y}_m : \tilde{s}_1|_{i_m}^{i_{m+1}} \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}_1|_{j_m}^{j_{m+1}}$ and $\bar{y}_k : \tilde{s}_2|_{i_k}^{i_{k+1}} \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{t}_2|_{j_k}^{j_{k+1}}$, are added to S via the $\mathcal{R}_{\mathcal{R},\lambda}\text{-A}_{\text{sim}}$ rule in the first step of our derivation. These new variables \bar{y}_m and \bar{y}_k will later be united by the store cleaning rules which results in $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$.
- A similar reasoning can be used in the case that $\bar{y} \in \tilde{r}_m$ and $\bar{y} \in \tilde{r}_k$ for some $0 \leq m, k \leq n$ with $m \neq k$, or in the case that we have $\bar{y} \in \tilde{r}_m$ for some $0 \leq m \leq n$ and $\bar{y} = \bar{v}_k$ some $0 \leq k \leq n$ with $m \neq k$.
- If, however, the \bar{y} 's are in the same \tilde{r}_k with $0 \leq k \leq n$ we know from the induction hypothesis that there exists a substitution ψ_k such that $\tilde{r}_k \psi_k = \bar{z}_k \sigma_k = \bar{z}_k \sigma_1 \dots \sigma_n$. Therefore, $\bar{y} \psi_k$ also occurs twice in the corresponding positions in $\bar{z}_k \sigma_1 \dots \sigma_n$. Using $\bar{z}_k \sigma_1 \dots \sigma_n \lesssim_{\mathcal{R},\lambda} \bar{z}_k \sigma$ we get $\tilde{r}_k \lesssim_{\mathcal{R},\lambda} \bar{z}_k \sigma_1 \dots \sigma_n = \bar{z}_k \sigma^D \sigma_1 \dots \sigma_n$ and it follows that $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma^D \sigma_1 \dots \sigma_n \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$.

Thus, we have proven that $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ constructs a derivation such that σ is an $\mathcal{R}_{\mathcal{R},\lambda,\alpha,\beta}$ -anti-unifier with $\alpha \geq \alpha' \geq \lambda$ and $\beta \geq \beta' \geq \lambda$. \square

Hence, $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ computes a finite complete set of $(\mathcal{R}_{\mathcal{R},\lambda})$ -generalizations for the given sequences. On the other hand, this set might not be minimal (but can be minimized), which is not surprising, since the syntactic rigid anti-unification algorithm from [30] has the same property and it can be obtained from our algorithm as a special case.

5.2 Fuzzy Unranked Anti-Unification with Proximity Relations

In this section, we further modify the notions already introduced in order to create a procedure which is able to solve unranked anti-unification problems under proximity relations. For this, we will again use extended terms and their accompanying concepts.

Definition 5.10 ((\mathcal{R}, λ) -X-Alignment)

Let $\tilde{\mathbf{s}} = \lceil \mathbf{s}_1, \dots, \mathbf{s}_n \rceil$ and $\tilde{\mathbf{t}} = \lceil \mathbf{t}_1, \dots, \mathbf{t}_m \rceil$ be two finite sequences of X-terms, \mathcal{R} a proximity relation and λ a cut value. A sequence of the form $\lceil \mathbf{a}_1[i_1, j_1], \dots, \mathbf{a}_k[i_k, j_k] \rceil$ is called an (\mathcal{R}, λ) -X-alignment between \mathbf{s} and \mathbf{t} if

- $[i_l, j_l] \in \{1, \dots, n\} \times \{1, \dots, m\}$ for $1 \leq l \leq k$ and
- $\mathbf{a}_l = \text{head}(\tilde{\mathbf{s}}|_{i_l}) \cap \text{head}(\tilde{\mathbf{t}}|_{j_l})$ for $1 \leq l \leq k$.

Definition 5.11 ((\mathcal{R}, λ) -X-Rigidity Function)

Let \mathcal{R} be a proximity relation and λ a cut value. A binary function $\mathcal{R}_{\mathcal{R}, \lambda}$ that, given two sequences of X-terms $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$, returns a set of (\mathcal{R}, λ) -X-alignments between $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$, is called an (\mathcal{R}, λ) -X-rigidity function.

Note that the notions of $\mathcal{R}_{\mathcal{R}, \lambda}$ -generalizations, (\mathcal{R}, λ) -alignments and (\mathcal{R}, λ) -rigidity functions work the same way for proximity relations as for similarity relations, see Definition 5.7.

Definition 5.12 ($\mathcal{R}_{\mathcal{R}, \lambda}$ -X-Generalization)

Let $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$ be two variable-disjoint sequences of terms, \mathcal{R} a rigidity function, \mathcal{R} a proximity relation and λ a cut value. An X-term $\tilde{\mathbf{r}}$ is an $\mathcal{R}_{\mathcal{R}, \lambda}$ -X-generalization of $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$, if every $\tilde{r} \in \text{terms}(\tilde{\mathbf{r}})$ is an $\mathcal{R}_{\mathcal{R}, \lambda}$ -generalization of $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$.

Definition 5.13

An X-substitution σ is an $\mathcal{R}_{\mathcal{R}, \lambda}$ -anti-unifier of the $\mathcal{R}_{\mathcal{R}, \lambda}$ -anti-unification problem Γ , if for each $\bar{v} : \tilde{\mathbf{s}} \stackrel{?}{\simeq}_{\mathcal{R}, \lambda} \tilde{\mathbf{t}} \in \Gamma$, we have $\bar{v}\sigma$ as an $\mathcal{R}_{\mathcal{R}, \lambda}$ -X-generalization of both $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$.

The set of all $\mathcal{R}_{\mathcal{R}, \lambda}$ -X-anti-unifiers of Γ is denoted by $\mathcal{A}_X(\Gamma, \mathcal{R}_{\mathcal{R}, \lambda})$.

Next, we adapt the rules of syntactic unranked anti-unification [30] again; this time to work with a proximity relation \mathcal{R} , a cut value λ and a rigidity function \mathcal{R} . Moreover, instead of term sequences, we consider their proximity classes. As the result, we obtain the procedure $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R}, \lambda})$:

 $\mathcal{R}_{\mathcal{R}, \lambda}$ D-A_{prox}: $\mathcal{R}_{\mathcal{R}, \lambda}$ -Rigid Decomposition

$$\begin{aligned}
 & \langle \{\bar{x} : \tilde{\mathbf{s}} \stackrel{?}{\simeq}_{\mathcal{R}, \lambda} \tilde{\mathbf{t}}\} \cup \Gamma; S; \sigma \rangle \implies \\
 & \quad \langle \{\bar{z}_k : \tilde{\mathbf{s}}_k \stackrel{?}{\simeq}_{\mathcal{R}, \lambda} \tilde{\mathbf{t}}_k \mid 1 \leq k \leq n\} \cup \Gamma; \\
 & \quad \{\bar{y}_0 : \tilde{\mathbf{s}}|_0^{i_1} \stackrel{?}{\simeq}_{\mathcal{R}, \lambda} \tilde{\mathbf{t}}|_0^{j_1}\} \cup \{\bar{y}_k : \tilde{\mathbf{s}}|_{i_k}^{i_{k+1}} \stackrel{?}{\simeq}_{\mathcal{R}, \lambda} \tilde{\mathbf{t}}|_{j_k}^{j_{k+1}} \mid 1 \leq k \leq n-1\} \cup \\
 & \quad \{\bar{y}_n : \tilde{\mathbf{s}}|_{i_n}^{|\tilde{\mathbf{s}}|+1} \stackrel{?}{\simeq}_{\mathcal{R}, \lambda} \tilde{\mathbf{t}}|_{j_n}^{|\tilde{\mathbf{t}}|+1}\} \cup S; \\
 & \quad \sigma\{\bar{x} \mapsto \lceil \bar{y}_0, \mathbf{f}_1(\bar{z}_1), \bar{y}_1, \dots, \bar{y}_{n-1}, \mathbf{f}_n(\bar{z}_n), \bar{y}_n \rceil \}, \\
 & \text{if } \mathcal{R}_{\mathcal{R}, \lambda}(\tilde{\mathbf{s}}, \tilde{\mathbf{t}}) \text{ contains a sequence } \lceil \mathbf{f}_1[i_1, j_1], \dots, \mathbf{f}_n[i_n, j_n] \rceil \text{ such that for all} \\
 & \quad 1 \leq k \leq n, \tilde{\mathbf{s}}|_{i_k} \simeq_{\mathcal{R}, \lambda} \mathbf{f}_k(\tilde{\mathbf{s}}_k), \tilde{\mathbf{t}}|_{j_k} \simeq_{\mathcal{R}, \lambda} \mathbf{f}_k(\tilde{\mathbf{t}}_k), \text{ and } \bar{y}_0, \bar{y}_k \text{'s and } \bar{z}_k \text{'s are fresh.}
 \end{aligned}$$

$\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{prox}}$: $\mathcal{R}_{\mathcal{R},\lambda}\text{-Rigid Solve}$

$$\langle \{\bar{x} : \tilde{s} \hat{\simeq}_{\mathcal{R},\lambda}^? \tilde{t}\} \cup \Gamma; S; \sigma \rangle \Longrightarrow \langle \Gamma; \{\bar{x} : \tilde{s} \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}\} \cup S; \sigma \rangle,$$

if $\mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t}) = \emptyset$.

$\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{prox}}$: $\mathcal{R}_{\mathcal{R},\lambda}\text{-Merge}$

$$\langle \Gamma; \{\bar{x}_1 : \tilde{s}_1 \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}_1, \bar{x}_2 : \tilde{s}_2 \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}_2\} \cup S; \sigma \rangle \Longrightarrow \langle \Gamma; \{\bar{x}_1 : \tilde{s} \hat{\simeq}_{\mathcal{R},\lambda} \tilde{t}\} \cup S; \sigma \{ \bar{x}_2 \mapsto \bar{x}_1 \} \rangle,$$

if $\bar{x}_1 \neq \bar{x}_2$; $\tilde{s} = \tilde{s}_1 \sqcap \tilde{s}_2 \neq \emptyset$ and $\tilde{t} = \tilde{t}_1 \sqcap \tilde{t}_2 \neq \emptyset$.

$\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{prox}}$: $\mathcal{R}_{\mathcal{R},\lambda}\text{-Rigid Clean Store}$

$$\langle \Gamma; \{\bar{x} : \ulcorner \hat{\simeq}_{\mathcal{R},\lambda} \urcorner\} \cup S; \sigma \rangle \Longrightarrow \langle \Gamma; S; \sigma \{ \bar{x} \mapsto \ulcorner \urcorner \} \rangle.$$

Slightly different to $\mathfrak{G}_{\text{sim}}(\mathcal{R}_{\mathcal{R},\lambda})$ this procedure works on triples $\langle \Gamma; S; \sigma \rangle$ with Γ being a set of anti-unification triples for sequences of X-terms, S being the store of already solved AUT's and σ being the X-substitution transforming into an $\mathcal{R}_{\mathcal{R},\lambda}$ -X-anti-unifier. For $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ we start with two term sequences \tilde{s} and \tilde{t} to be generalized and construct our initial configuration as $\langle \{\bar{x} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \hat{\simeq}_{\mathcal{R},\lambda}^? \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})\}; \emptyset; \varepsilon \rangle$. After that, we start applying the rules in all possible ways until $\Gamma = \emptyset$ and there are no duplicate generalization variables or empty sequence pairs in S . The next theorem shows that the procedure terminates:

Theorem 5.7 (Termination of $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$)

The procedure $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ terminates for every input.

Proof. In order to show that the procedure terminates, we create a complexity measurement $\langle m_1, m_2 \rangle$, with m_1 being number of symbols in Γ and m_2 being the number of symbols in S , for a given rigidity function $\mathcal{R}_{\mathcal{R},\lambda}$, a set of $\mathcal{R}_{\mathcal{R},\lambda}$ -AUT's Γ , the store S and the $\mathcal{R}_{\mathcal{R},\lambda}$ -X-anti-unifier σ . The tuple is ordered by lexicographic ordering and is well founded. Each step of the procedure strictly reduces the complexity measure:

	m_1	m_2
$\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{prox}}$	$>$	
$\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{prox}}$	$>$	
$\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{prox}}$	\geq	$>$
$\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{prox}}$	\geq	$>$

Therefore each transformation chain terminates. Even though the decomposition rule $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{prox}}$ can cause branching, only a finite amount of branches can split as the only a finite amount of alignments are returned by the rigidity function. \square

The statement of Lemma 5.2, that the order of application between a store-cleaning rule and a non-store-cleaning rule does not change the outcome, is applicable to $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ as well:

Lemma 5.8

Let $\langle \Gamma; S; \sigma \rangle \Longrightarrow_{R_1} \langle \Gamma_1; S_1; \sigma\theta_1 \rangle \Longrightarrow_{R_2} \langle \Gamma_2; S_2; \sigma\theta_1\theta_2 \rangle$ be a sequence of transformations where $R_1 \in \{\mathcal{R}_{\mathcal{R},\lambda}M\text{-}A_{\text{prox}}, \mathcal{R}_{\mathcal{R},\lambda}CS\text{-}A_{\text{prox}}\}$ and $R_2 \in \{\mathcal{R}_{\mathcal{R},\lambda}D\text{-}A_{\text{prox}}, \mathcal{R}_{\mathcal{R},\lambda}S\text{-}A_{\text{prox}}\}$. Then there exists a transformation sequence $\langle \Gamma; S; \sigma \rangle \Longrightarrow_{R_2} \langle \Gamma'_1; S'_1; \sigma\theta_2 \rangle \Longrightarrow_{R_1} \langle \Gamma'_2; S'_2; \sigma\theta_2\theta_1 \rangle$ such that $\Gamma_2 = \Gamma'_2$, $S_2 = S'_2$ and $\sigma\theta_1\theta_2 = \sigma\theta_2\theta_1$.

Proof. Similar to [30]. □

Lemma 5.9

If $\langle \Gamma; S_1; \sigma \rangle \Longrightarrow^* \langle \emptyset; S_2; \sigma\theta \rangle$ is a derivation in $\mathfrak{S}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ using only the rules $(\mathcal{R}_{\mathcal{R},\lambda}D\text{-}A_{\text{prox}})$ and $(\mathcal{R}_{\mathcal{R},\lambda}S\text{-}A_{\text{prox}})$, then $\theta \in \mathcal{A}_X(\Gamma, \mathcal{R}_{\mathcal{R},\lambda})$.

Proof. This proof will be done by induction on the number of steps in the sequence of transformations. If the derivation has only one step, i.e. $\langle \bar{x} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}); S; \sigma \rangle \Longrightarrow \langle \emptyset; \bar{x} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}) \cup S; \sigma\theta \rangle$, then either $\theta = \varepsilon$ if the $(\mathcal{R}_{\mathcal{R},\lambda}S\text{-}A_{\text{prox}})$ rule is used, or $\theta = \{\bar{x} \mapsto \bar{y}_0\}$ with \bar{y}_0 being a new sequence variable if $(\mathcal{R}_{\mathcal{R},\lambda}D\text{-}A_{\text{prox}})$ is used. We know that the generalization in this case is a sequence variable.

We assume as the induction hypothesis that this lemma holds for sequences of the length m and want to prove that it is also correct for derivations of length $m+1$. A system taking $m+1$ steps to solve shall be of the form $\langle \Gamma_{m+1}; S_{m+1}; \sigma \rangle$.

Let $\Gamma_{m+1} = \{\bar{x} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})\} \cup \Gamma'_{m+1}$ and assume that it is to be transformed by $(\mathcal{R}D\text{-}A_{\text{prox}})$. This means we get $\Gamma_m = \{\bar{z}_k : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}_k) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}_k) \mid 1 \leq k \leq n\} \cup \Gamma'_{m+1}$, $S_m = \{\bar{y}_0 : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}_0^{i_1}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}_0^{j_1})\} \cup \{\bar{y}_k : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}_{i_k}^{i_k+1}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}_{j_k}^{j_k+1}) \mid 1 \leq k \leq n-1\} \cup \{\bar{y}_n : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}_{i_n}^{|\tilde{s}|+1}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}_{j_n}^{|\tilde{t}|+1})\} \cup S_{m+1}$ and $\theta_m = \{\bar{x} \mapsto \ulcorner \bar{y}_0, \mathbf{f}_1(\bar{z}_1), \bar{y}_1, \dots, \bar{y}_{n-1}, \mathbf{f}_n(\bar{z}_n), \bar{y}_n \urcorner\}$ as defined in rule $(\mathcal{R}D\text{-}A_{\text{prox}})$. Since the number of the remaining steps to solve the system $\langle \{\bar{z}_k : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}_k) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}_k) \mid 1 \leq k \leq n\} \cup \Gamma'_{m+1}; S_{m+1} \cup S_m; \sigma\theta_m \rangle \Longrightarrow^* \langle \emptyset; S_{m+1} \cup \dots \cup S_0; \sigma\theta_m \dots \theta_0 \rangle$ is only m we know from the induction hypothesis that for $S = S_{m-1} \cup \dots \cup S_0$ and $\theta = \theta_{m-1} \dots \theta_0$, then $\theta \in \mathcal{A}_X(\Gamma_m, \mathcal{R}_{\mathcal{R},\lambda})$ which means that for $\theta \in \text{subs}(\theta)$, $\bar{v}\theta$ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization for both \tilde{r} and \tilde{q} for all $\{\bar{v} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{r}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{q})\} \in \Gamma_m$. Hence, this also holds for \bar{z}_k for $1 \leq k \leq n$ and therefore $\bar{x}\theta_m\theta$ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization for \tilde{s} and \tilde{t} for all $\theta_m \in \text{subs}(\theta_m)$ and $\theta \in \text{subs}(\theta)$, i.e. $\theta_m\theta \in \mathcal{A}_X(\Gamma_{m+1}, \mathcal{R}_{\mathcal{R},\lambda})$.

Next let Γ_{m+1} be transformed by $(\mathcal{R}_{\mathcal{R},\lambda}S\text{-}A_{\text{prox}})$, which indicates that $\mathcal{R}_{\mathcal{R},\lambda}(\mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}), \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})) = \emptyset$. We get $\Gamma_m = \Gamma'_{m+1}$, $S_m = \{\bar{x} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})\} \cup S_{m+1}$ and $\theta_m = \varepsilon$. Since the number of the remaining steps to solve $\langle \Gamma_m; S_{m+1} \cup S_m; \sigma\theta_m \rangle \Longrightarrow \langle \emptyset; S_{m+1} \cup \dots \cup S_1; \sigma\theta_m \dots \theta_0 \rangle$ is only m we know from our induction hypothesis that for $S = S_{m-1} \cup \dots \cup S_0$ and $\theta = \theta_{m-1} \dots \theta_0$, that $\theta \in \mathcal{A}_X(\Gamma_m, \mathcal{R}_{\mathcal{R},\lambda})$ meaning that for $\theta \in \text{subs}(\theta)$, $\bar{v}\theta$ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization for both \tilde{r} and \tilde{q} for all $\{\bar{v} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{r}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{q})\} \in \Gamma_m$. Since

$\mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t}) = \emptyset$, it holds that $\theta_m \theta = \theta$ and we get that $\bar{x}\theta$ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization for both \tilde{s} and \tilde{t} , meaning $\theta \in \mathcal{A}_X(\Gamma_{m+1}, \mathcal{R}_{\mathcal{R},\lambda})$. \square

Lemma 5.10

If $\langle \{\bar{x} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})\}; \emptyset; \varepsilon \rangle \Longrightarrow^* \langle \emptyset; S_1; \sigma \rangle \Longrightarrow_R \langle \emptyset; S; \sigma\theta \rangle$ is a derivation in $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ such that $\bar{x}\sigma$ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -X-generalization of \tilde{s} and \tilde{t} , and $R \in \{\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{prox}}, \mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{prox}}\}$, then $\sigma\theta \in \mathcal{A}_X(\Gamma, \mathcal{R}_{\mathcal{R},\lambda})$ with $\Gamma = \{\bar{x} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})\}$.

Proof. Case $R = \mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{prox}}$:

Let $S_1 = \{\bar{v}_1 : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{q}_1) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{r}_1), \bar{v}_2 : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{q}_2) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{r}_2)\} \cup S'_1$. With $\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{prox}}$ the last step of the derivation produces $S = \{\bar{v}_1 : \tilde{\mathbf{q}} \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \tilde{\mathbf{r}}\} \cup S'_1$ with $\tilde{\mathbf{s}} = \tilde{\mathbf{s}}_1 \sqcap \tilde{\mathbf{s}}_2 \neq \emptyset$ and $\tilde{\mathbf{t}} = \tilde{\mathbf{t}}_1 \sqcap \tilde{\mathbf{t}}_2 \neq \emptyset$ and $\theta = \{\bar{v}_2 \mapsto \bar{v}_1\}$. With the same reasoning as in the proof of Lemma 5.4 we can argue that $\bar{x}\sigma\theta$ is also an X-generalization of \tilde{s} and \tilde{t} as the replacing of \bar{v}_2 's in $\bar{x}\sigma$ retains the property.

The proof of the remaining properties of $\bar{x}\sigma\theta$ being an $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization for $\sigma \in \text{subs}(\sigma)$ and $\theta \in \text{subs}(\theta)$ again does not change from [30]. The proof for $R = \mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{prox}}$ is straight forward. \square

Theorem 5.11 (Soundness of $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$)

If $\langle \Gamma; \emptyset; \varepsilon \rangle \Longrightarrow^* \langle \emptyset; S; \sigma \rangle$ is a derivation in $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ with $\Gamma = \{\bar{x} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})\}$, then $\sigma \in \mathcal{A}_X(\Gamma; \mathcal{R}_{\mathcal{R},\lambda})$.

Proof. Again the proof idea comes from [30]. By Lemma 5.8 we know that we can first concentrate on emptying the set of AUT's before applying the store rules $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{prox}}$ and $\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{prox}}$. After reaching the point of $\langle \emptyset; S'; \sigma' \rangle$, before the store rules are applied, we know from Lemma 5.9 that $\bar{x}\sigma'$ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -X-generalization of \tilde{s} and \tilde{t} . By applying $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{prox}}$ and $\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{prox}}$ as necessary we get $\langle \emptyset; S; \sigma \rangle$ and from Lemma 5.10 we know that the changes to the substitution and the degrees results in $\bar{x}\sigma$ being an $\mathcal{R}_{\mathcal{R},\lambda}$ -X-generalization of \tilde{s} and \tilde{t} . \square

Theorem 5.12 (Completeness of $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$)

Let \tilde{r} be an $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization of \tilde{s} and \tilde{t} . Then $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ constructs a derivation $\langle \bar{x} : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})\}; \emptyset; \varepsilon \rangle \Longrightarrow^+ \langle \emptyset; S; \sigma \rangle$ such that σ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -X-anti-unifier for \tilde{s} and \tilde{t} with $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$ for all $\sigma \in \text{subs}(\sigma)$.

Proof. The proof of completeness will be done via induction over the size s of \tilde{r} .

In the case that $s = 0$, we would have $\tilde{r} = \ulcorner \urcorner$ as well as $\tilde{s} = \tilde{t} = \ulcorner \urcorner$ and $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ would return $\sigma = \{\bar{x} \mapsto \ulcorner \urcorner\}$ after applying $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{prox}}$ followed by $\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{prox}}$.

For $s = 1$ we have three different cases:

- if $\tilde{r} = c$, then $\lceil \{c\} \rceil \in \mathcal{R}_{\mathcal{R},\lambda}(\mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}), \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}))$. After applying $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{prox}}$ and $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{prox}}$ the procedure returns $\sigma = \{\bar{x} \mapsto \{c\}\}$ which fulfills $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma = c$ for all $\sigma \in \text{subs}(\sigma)$.
- if $\tilde{r} = \bar{x}$, then $\mathcal{R}_{\mathcal{R},\lambda}(\mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}), \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})) = \emptyset$. The procedure applies the rule $\mathcal{R}_{\mathcal{R},\lambda}\text{S-A}_{\text{prox}}$ giving us $\sigma = \varepsilon$ and therefore $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma = \bar{x}$ for $\sigma \in \text{subs}(\sigma)$.
- if $\tilde{r} = \bar{y}_0$ with $\bar{y}_0 \neq \bar{x}$, then $\lceil \cdot \rceil \in \mathcal{R}_{\mathcal{R},\lambda}(\mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}), \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}))$. We get $\sigma = \{\bar{x} \mapsto \{\bar{y}'_0\}\}$, with \bar{y}'_0 being a fresh sequence variable, after using rule $\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{prox}}$. This means $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{y}'_0 = \bar{x}\sigma$ for $\sigma \in \text{subs}(\sigma)$.

From \tilde{r} being an $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization of \tilde{s} and \tilde{t} , we know that there exists an alignment $\lceil f_1[i_1, j_1], \dots, f_n[i_n, j_n] \rceil \in \mathcal{R}_{\mathcal{R},\lambda}(\tilde{s}, \tilde{t})$ such that

- no successive sequence variables occur in \tilde{r} ,
- after removing all sequence variables from \tilde{r} it would have the form of $\lceil f_1(\tilde{r}_1), \dots, f_n(\tilde{r}_n) \rceil$,
- there exist two sequences \tilde{s}'_k and \tilde{t}'_k such that \tilde{r}_k is a generalization of \tilde{s}'_k and \tilde{t}'_k , $\tilde{s}|_{i_k} = f_k(\tilde{s}'_k)$ and $\tilde{t}|_{j_k} = f_k(\tilde{t}'_k)$, for $1 \leq k \leq n$.

As induction hypothesis, we assume that for all \tilde{r}^H with $\text{size}(\tilde{r}^H) < s$ and for all sequences \tilde{s}^H and \tilde{t}^H , it holds that if \tilde{r}^H is an $\mathcal{R}_{\mathcal{R},\lambda}$ -generalization of \tilde{s}^H and \tilde{t}^H , then $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ constructs σ^H , an $\mathcal{R}_{\mathcal{R},\lambda}$ -X-anti-unifier for $\bar{x}^H : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}^H) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}^H)$, such that $\tilde{r}^H \lesssim_{\mathcal{R},\lambda} \bar{x}^H \sigma^H$ for $\sigma^H \in \text{subs}(\sigma^H)$. Next we make the proof for sequences of size s :

Let $\tilde{r} = \lceil \bar{v}_0, f_1(\tilde{r}_1), \bar{v}_1, \dots, \bar{v}_{n-1}, f(\tilde{r}_n), \bar{v}_n \rceil$. We know that for $1 \leq k \leq n$ it holds that $\text{size}(\tilde{r}_k) < \text{size}(\tilde{r})$ and therefore by the induction hypothesis there exists a derivation $\langle \{\bar{z}_k : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}'_k) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}'_k)\}; \emptyset; \varepsilon \rangle \Longrightarrow^* \langle \emptyset; S_k; \sigma_k \rangle$ with $\tilde{r}_k \lesssim_{\mathcal{R},\lambda} \bar{z}_k \sigma_k$ for all $\sigma_k \in \text{subs}(\sigma_k)$. Adding a decomposition step at the beginning of the derivation gives us $\langle \{\mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})\}; \emptyset; \varepsilon \rangle \Longrightarrow_{\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{prox}}} \langle \{\bar{z}_k : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}'_k) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}'_k)\}; S^D; \sigma^D \rangle \Longrightarrow^* \langle \emptyset; S^D \cup S_1 \cup \dots \cup S_n; \sigma^D \sigma_1 \dots \sigma_n \rangle$. From Theorem 5.11 that $\bar{x}\sigma^D \sigma_1 \dots \sigma_n$ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -X-generalization of \tilde{s} and \tilde{t} . Since for $1 \leq k \leq n$ all \bar{z}_k 's are unique, it holds that $\bar{z}_k \sigma_k = \bar{z}_k \sigma_1 \dots \sigma_n$.

After applying the store rules $\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{prox}}$ and $\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{prox}}$ our derivation takes on the following form: $\langle \{\mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t})\}; \emptyset; \varepsilon \rangle \Longrightarrow_{\mathcal{R}_{\mathcal{R},\lambda}\text{D-A}_{\text{prox}}} \langle \{\bar{z}_k : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}'_k) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}'_k)\}; S^D; \sigma^D \rangle \Longrightarrow^* \langle \emptyset; S^D \cup S_1 \cup \dots \cup S_n; \sigma^D \sigma_1 \dots \sigma_n \rangle \Longrightarrow^*_{\mathcal{R}_{\mathcal{R},\lambda}\text{M-A}_{\text{prox}}|\mathcal{R}_{\mathcal{R},\lambda}\text{CS-A}_{\text{prox}}} \langle \emptyset; S; \sigma \rangle$. By soundness we get that $\bar{x}\sigma$ is an $\mathcal{R}_{\mathcal{R},\lambda}$ -X-generalization of \tilde{s} and \tilde{t} .

We want to show that $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$. We know it holds that $\sigma^D \sigma_1 \dots \sigma_n \lesssim_{\mathcal{R},\lambda} \sigma$, hence, for all $\sigma \in \sigma$ there exists a substitution $\sigma^D \sigma_1 \dots \sigma_n \in \text{subs}(\sigma^D \sigma_1 \dots \sigma_n)$ such that $\bar{x}\sigma^D \sigma_1 \dots \sigma_n \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$. Should \tilde{r} contain only unique variables, then we know that $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma^D \sigma_1 \dots \sigma_n$ for all $\sigma^D \sigma_1 \dots \sigma_n \in \text{subs}(\sigma^D \sigma_1 \dots \sigma_n)$. Therefore it holds that $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma^D \sigma_1 \dots \sigma_n \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$.

Duplicate variables require a more thorough look:

- Let \bar{y} be the sequence variable occurring twice in \tilde{r} with $\bar{y} = \bar{v}_m = \bar{v}_k$ for $1 \leq m, k \leq n$ with $m \neq k$. We know that $\mathcal{R}(\mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}|_{i_m}^{i_{m+1}}), \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}|_{i_k}^{i_{k+1}})) \geq \lambda$ and $\mathcal{R}(\mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}|_{j_m}^{j_{m+1}}), \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}|_{j_k}^{j_{k+1}})) \geq \lambda$ and through $\mathcal{R}_{\mathcal{R},\lambda}$ -D-A_{prox} in the first step two new AUT's, $\bar{y}_m : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}|_{i_m}^{i_{m+1}}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}|_{j_m}^{j_{m+1}})$ and $\bar{y}_k : \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{s}|_{i_k}^{i_{k+1}}) \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \mathbf{pc}_{\mathcal{R},\lambda}(\tilde{t}|_{j_k}^{j_{k+1}})$, are added to S . Later on, these two new sequence variables will be merged or cleaned by the store cleaning rules.
- In the case that $\bar{y} \in \tilde{r}_m$ and $\bar{y} \in \tilde{r}_k$ or if $\bar{y} \in \tilde{r}_m$ and $\bar{y} = \bar{y}_k$ for $1 \leq m, k, \leq n$ with $m \neq k$ then a similar reasoning can be used.
- Finally if there are two instances of \bar{y} in \tilde{r}_m for $0 \leq m \leq n$, we can use our induction hypothesis to know that for all $\sigma_m \in \text{subs}(\sigma_m)$ there exists a substitution ψ , such that $\tilde{r}_m\psi = \bar{z}_m\sigma_m$. We want to show that $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$ for all $\sigma \in \sigma$ and we know that $\bar{z}_m\sigma_1 \cdots \sigma_n \lesssim_{\mathcal{R},\lambda} \bar{z}_m\sigma$ and $\bar{z}_m\sigma_1 \cdots \sigma_n = \bar{z}_m\sigma^D\sigma_1 \cdots \sigma_n$. Therefore we can say, that for all $\sigma \in \sigma$ there exist substitutions $\sigma_1 \cdots \sigma_n \in \text{subs}(\sigma_1 \cdots \sigma_n)$ and $\sigma^D \in \text{subs}(\sigma^D)$ such that $\bar{x}\sigma^D\sigma_1 \cdots \sigma_n \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$. We also know that $\bar{z}_m\sigma_m = \bar{z}_m\sigma_1 \cdots \sigma_n$, meaning there exists a $\sigma_m \in \text{subs}(\sigma_m)$ such that $\bar{y}_m\psi$ occurs twice in the corresponding positions of $\bar{z}_m\sigma_1 \cdots \sigma_n$. Hence, $\tilde{r}_m \lesssim_{\mathcal{R},\lambda} \bar{z}_m\sigma_1 \cdots \sigma_n$ and we get that $\tilde{r} \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma^D\sigma_1 \cdots \sigma_n \lesssim_{\mathcal{R},\lambda} \bar{x}\sigma$.

We have shown that the derivation constructed by $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ produces an X-substitution σ which is an $\mathcal{R}_{\mathcal{R},\lambda}$ -X-anti-unifier of \tilde{s} and \tilde{t} . \square

If we wanted to know the approximation degree for the computed generalizations, we could simply use the graded proximity classes. Running the procedure $\mathfrak{G}_{\text{prox}}(\mathcal{R}_{\mathcal{R},\lambda})$ as just described, we can then simply extract the degree from the graded terms.

Example 5.3. Let \mathcal{R} be a similarity relation with $\mathcal{R}(f, g) = 0.9$, $\mathcal{R}(g, h) = 0.8$, $\mathcal{R}(a, b) = 0.9$, $\mathcal{R}(b, c) = 0.7$ and $\mathcal{R}(a, c) = 0.9$, $\lambda = 0.8$ a cut value and \mathcal{R} an alignment function returning the longest common subsequences. We will only show small example, as it can be quite exhausting to read with all the (graded) proximity classes. Wanting to generalize $s = f(a, b)$ and $t = g(c)$ we set

$$\Gamma = \{\bar{x} : \{\langle f, 1 \rangle, \langle g, 0.9 \rangle\}(\{\langle a, 1 \rangle, \langle b, 0.9 \rangle, \langle c, 0.9 \rangle\}, \{\langle b, 1 \rangle, \langle a, 0.9 \rangle, \langle c, 0.7 \rangle\}) \\ \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \{\langle g, 1 \rangle, \langle f, 0.9 \rangle, \langle h, 0.8 \rangle\}(\{\langle c, 1 \rangle, \langle a, 0.9 \rangle, \langle b, 0.7 \rangle\})\}.$$

The following shows one branch of the procedure:

$$\begin{aligned} & \langle \Gamma; \emptyset; \varepsilon; 1; 1 \rangle \\ \xrightarrow{\text{D-A}_{\text{sim}}} & \langle \{\bar{z} : \ulcorner \{\langle a, 1 \rangle, \langle b, 0.9 \rangle, \langle c, 0.9 \rangle\}, \{\langle b, 1 \rangle, \langle a, 0.9 \rangle, \langle c, 0.7 \rangle\} \urcorner \} \\ & \stackrel{?}{\simeq}_{\mathcal{R},\lambda} \ulcorner \{\langle c, 1 \rangle, \langle a, 0.9 \rangle, \langle b, 0.7 \rangle\} \urcorner \rangle; \end{aligned}$$

$$\begin{array}{l}
 \emptyset; \{\bar{x} \mapsto \{\langle f, 0.9 \rangle, \langle g, 0.9 \rangle\}(\bar{z})\}, \\
 \text{for } \{\langle f, 0.9 \rangle, \langle g, 0.9 \rangle\}[1, 1] \in \mathcal{R}_{\mathcal{R}, \lambda}(\mathbf{pc}_{\mathcal{R}, \lambda}(s), \mathbf{pc}_{\mathcal{R}, \lambda}(t)) \\
 \xrightarrow{\text{D-A}_{\text{sim}}} \langle \{\bar{z}_1 : \ulcorner \ulcorner \hat{\simeq}_{\mathcal{R}, \lambda}^? \urcorner \urcorner\}; \{\bar{z}_2 : \{\langle b, 1 \rangle, \langle a, 0.9 \rangle, \langle c, 0.7 \rangle\} \hat{\simeq}_{\mathcal{R}, \lambda} \urcorner \urcorner\}; \\
 \{\bar{x} \mapsto \{\langle f, 0.9 \rangle, \langle g, 0.9 \rangle\}(\{\langle a, 0.9 \rangle, \langle c, 0.9 \rangle, \langle b, 0.7 \rangle\}(\bar{z}_1, \bar{z}_2))\}, \\
 \text{for } \{\langle a, 0.9 \rangle, \langle c, 0.9 \rangle, \langle b, 0.7 \rangle\}[1, 1] \\
 \in \mathcal{R}_{\mathcal{R}, \lambda}(\ulcorner \{\langle a, 1 \rangle, \langle b, 0.9 \rangle, \langle c, 0.9 \rangle\}, \{\langle b, 1 \rangle, \langle a, 0.9 \rangle, \langle c, 0.7 \rangle\} \urcorner, \\
 \{\langle c, 1 \rangle, \langle a, 0.9 \rangle, \langle b, 0.7 \rangle\}) \\
 \xrightarrow{\text{S-A}_{\text{sim}}} \langle \emptyset; \{\bar{z}_1 : \ulcorner \ulcorner \hat{\simeq}_{\mathcal{R}, \lambda}^? \urcorner \urcorner, \bar{z}_2 : \{\langle b, 1 \rangle, \langle a, 0.9 \rangle, \langle c, 0.7 \rangle\} \hat{\simeq}_{\mathcal{R}, \lambda} \urcorner \urcorner\}; \\
 \{\bar{x} \mapsto \{\langle f, 0.9 \rangle, \langle g, 0.9 \rangle\}(\{\langle a, 0.9 \rangle, \langle c, 0.9 \rangle, \langle b, 0.7 \rangle\}(\bar{z}_1, \bar{z}_2))\} \\
 \xrightarrow{\text{CS-A}_{\text{sim}}} \langle \emptyset; \{\bar{z}_2 : \{\langle b, 1 \rangle, \langle a, 0.9 \rangle, \langle c, 0.7 \rangle\} \hat{\simeq}_{\mathcal{R}, \lambda} \urcorner \urcorner\}; \\
 \{\bar{x} \mapsto \{\langle f, 0.9 \rangle, \langle g, 0.9 \rangle\}(\{\langle a, 0.9 \rangle, \langle c, 0.9 \rangle, \langle b, 0.7 \rangle\}, \bar{z}_2)\}
 \end{array}$$

We get the following generalizations of s and t :

- $f(a, \bar{z}_2)$ with proximity degree of 0.9,
- $g(a, \bar{z}_2)$ with proximity degree of 0.9,
- $f(c, \bar{z}_2)$ with proximity degree of 0.9,
- $g(c, \bar{z}_2)$ with proximity degree of 0.9,
- $f(b, \bar{z}_2)$ with proximity degree of 0.7,
- $g(b, \bar{z}_2)$ with proximity degree of 0.7.

6 Conclusion

How can we modify already existing procedures for solving unranked unification, matching and anti-unification problems for them to allow solutions under a quantitative degree under similarity and proximity relations?

In this thesis, we have answered this question by developing extended algorithms for all three procedures. We showed how existing methods can be adapted so that equality between symbols is replaced by flexible relations that express different degrees of similarity and closeness.

We modified the algorithm for syntactic unranked unification by removing rules dealing with sequence function symbols, as well as changing the decomposition rules such that now there is only one rule dealing with one pair at a time. By introducing the head removal rule we then could have the rules previously working on terms, now work on sequences of terms. Additionally we also decided to do a weaker approach by not using a decision algorithm and instead introduced terminating rules. This decision, however, led to the procedure not guaranteeing termination.

For our procedure for solving unranked matching problems with similarity and proximity relations we focused on those rules from our reference for syntactic unranked matching which deal with free function symbols. We kept the approach of linearizing non-linear problems to apply the linear matching algorithm before reconstructing a solution for the original non-linear problem. Carrying over our approach from unification, we introduced a head removal rule in order to be able to work on sequences of terms. Even though we proved our procedure without tracking the degree when solving with proximity relations, we also showed how a matching degree of a matcher can be computed.

As for anti-unification, we decided to modify the rigid generalization algorithm in order to avoid solutions which are not of interest. After changing all the necessary notions to work with similarity and later with proximity relation and proximity classes, it was quite straightforward to change the rules of syntactic rigid anti-unification. Again we only mentioned how to compute a degree for solving with proximity relations with graded proximity classes.

For future work, it would be interesting to design and integrate a decision algorithm into unranked unification for similarity and proximity relations. In our procedure for unranked

unification with proximity relations we did not show an option on how to compute a unification degree. This addition would also be an option for improvements.

For the entirety of this thesis we have only used the minimum triangular norm for its idempotent property. Without this idempotence we would have had to be more careful with how we compute degrees and would have had to change up some of the rules used. Generalizing the introduced procedures to work with other t-norms offers another interesting direction.

Bibliography

- [1] Hassan Aït-Kaci and Gabriella Pasi. “Fuzzy lattice operations on first-order terms over signatures with similar constructors: A constraint-based approach”. In: *Fuzzy Sets Syst.* 391 (2020), pp. 1–46. DOI: 10.1016/j.fss.2019.03.019.
- [2] Franz Baader and Wayne Snyder. “Unification Theory”. In: *Handbook of Automated Reasoning (in 2 volumes)*. Ed. by John Alan Robinson and Andrei Voronkov. Elsevier and MIT Press, 2001, pp. 445–532. DOI: 10.1016/b978-044450813-3/50010-2.
- [3] Johannes Bader et al. “Getafix: learning to fix bugs automatically”. In: *Proc. ACM Program. Lang.* 3.OOPSLA (2019), 159:1–159:27. DOI: 10.1145/3360585.
- [4] Alexander Baumgartner. “Anti-Unification Algorithms: Design, Analysis, and Implementation”. PhD thesis. RISC, Johannes Kepler University Linz, 2015. URL: <https://epub.jku.at/obvulihs/content/titleinfo/818923>.
- [5] Alexander Baumgartner and Temur Kutsia. *Quantitative generalization of variadic structures with binders*. Tech. rep. Research Institute for Symbolic Computation, Johannes Kepler University Linz, Austria, 2025.
- [6] Alexander Baumgartner and Temur Kutsia. “Unranked second-order anti-unification”. In: *Inf. Comput.* 255 (2017), pp. 262–286. DOI: 10.1016/J.IC.2017.01.005.
- [7] Bruno Buchberger et al. “Theorema 2.0: Computer-Assisted Natural-Style Mathematics”. In: *J. Formaliz. Reason.* 9.1 (2016), pp. 149–185. DOI: 10.6092/ISSN.1972-5787/4568.
- [8] Jorge Coelho and Mário Florido. “CLP(Flex): Constraint Logic Programming Applied to XML Processing”. In: *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences, Agia Napa, Cyprus, October 25-29, 2004, Proceedings, Part II*. Ed. by Robert Meersman and Zahir Tari. Vol. 3291. Lecture Notes in Computer Science. Springer, 2004, pp. 1098–1112. DOI: 10.1007/978-3-540-30469-2_17.
- [9] Nachum Dershowitz and Zohar Manna. “Proving Termination with Multiset Orderings”. In: *Commun. ACM* 22.8 (1979), pp. 465–476. DOI: 10.1145/359138.359142.
- [10] Besik Dundua, Temur Kutsia, and Mircea Marin. “Variadic equational matching in associative and commutative theories”. In: *J. Symb. Comput.* 106 (2021), pp. 78–109. DOI: 10.1016/j.jsc.2021.01.001.

- [11] Besik Dundua et al. “Extending the ρ Log Calculus with Proximity Relations”. In: *Applications of Mathematics and Informatics in Natural Sciences and Engineering*. Ed. by George Jaiani and David Natroshvili. Cham: Springer International Publishing, 2020, pp. 83–100. DOI: 10.1007/978-3-030-56356-1_6.
- [12] Jörg Endrullis, Jan Willem Klop, and Roy Overbeek. “Star Games and Hydras”. In: *Log. Methods Comput. Sci.* 17.2 (2021). URL: <https://lmcs.episciences.org/7518>.
- [13] Francesca Arcelli Fontana and Ferrante Formato. “A similarity-based resolution rule”. In: *Int. J. Intell. Syst.* 17.9 (2002), pp. 853–872. DOI: 10.1002/INT.10067.
- [14] Francesca Arcelli Fontana and Ferrante Formato. “Likelog: A Logic Programming Language for Flexible Data Retrieval”. In: *Proceedings of the 1999 ACM Symposium on Applied Computing, SAC’99, San Antonio, Texas, USA, February 28 - March 2, 1999*. Ed. by Barrett R. Bryant et al. ACM, 1999, pp. 260–267. DOI: 10.1145/298151.298348.
- [15] Ferrante Formato, Giangiacomo Gerla, and Maria I. Sessa. “Similarity-based Unification”. In: *Fundam. Informaticae* 41.4 (2000), pp. 393–414. DOI: 10.3233/FI-2000-41402.
- [16] Michael R. Genesereth. “Knowledge Interchange Format”. In: *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR’91). Cambridge, MA, USA, April 22-25, 1991*. Ed. by James F. Allen, Richard Fikes, and Erik Sandewall. Morgan Kaufmann, 1991, pp. 599–600.
- [17] Matthew L. Ginsberg. “The MVL Theorem Proving System”. In: *SIGART Bull.* 2.3 (1991), pp. 57–60. DOI: 10.1145/122296.122304.
- [18] Makoto Hamana. “Term rewriting with sequences”. In: *Proceedings of the First International Theorema Workshop*. RISC Technical Report series 97-20. Hagenberg, Austria, 1997.
- [19] Feryal Fulya Horozal, Florian Rabe, and Michael Kohlhase. *Flexary Operators for Formalized Mathematics*. Ed. by Stephen M. Watt et al. 2014. DOI: 10.1007/978-3-319-08434-3_23.
- [20] Haruo Hosoya and Benjamin C. Pierce. “Regular expression pattern matching for XML”. In: *J. Funct. Program.* 13.6 (2003), pp. 961–1004. DOI: 10.1017/S0956796802004410.
- [21] Pascual Julián Iranzo, Ginés Moreno, and José Antonio Riaza. “The Fuzzy Logic Programming language FASILL: Design and implementation”. In: *Int. J. Approx. Reason.* 125 (2020), pp. 139–168. DOI: 10.1016/J.IJAR.2020.06.002.
- [22] Pascual Julián Iranzo and Clemente Rubio-Manzano. “Proximity-based unification theory”. In: *Fuzzy Sets Syst.* 262 (2015), pp. 21–43. DOI: 10.1016/J.FSS.2014.07.006.
- [23] Pascual Julián Iranzo and Fernando Sáenz-Pérez. “A Fuzzy Datalog Deductive Database System”. In: *IEEE Trans. Fuzzy Syst.* 26.5 (2018), pp. 2634–2648. DOI: 10.1109/TFUZZ.2018.2806923.

- [24] Pascual Julián Iranzo and Fernando Sáenz-Pérez. “Bousi~Prolog: Design and implementation of a proximity-based fuzzy logic programming language”. In: *Expert Syst. Appl.* 213.Part A (2023), p. 118858. DOI: 10.1016/J.ESWA.2022.118858.
- [25] Pascual Julián Iranzo and Fernando Sáenz-Pérez. “Implementing WordNet Measures of Lexical Semantic Similarity in a Fuzzy Logic Programming System”. In: *Theory Pract. Log. Program.* 21.2 (2021), pp. 264–282. DOI: 10.1017/S1471068421000028.
- [26] Pascual Julián Iranzo and Fernando Sáenz-Pérez. “Proximity-Based Unification: An Efficient Implementation Method”. In: *IEEE Trans. Fuzzy Syst.* 29.5 (2021), pp. 1238–1251. DOI: 10.1109/TFUZZ.2020.2973129.
- [27] Artur Jeż. “Word equations in non-deterministic linear space”. In: *J. Comput. Syst. Sci.* 123 (2022), pp. 122–142. DOI: 10.1016/J.JCSS.2021.08.001.
- [28] Temur Kutsia. “Solving Equations Involving Sequence Variables and Sequence Functions”. In: *Artificial Intelligence and Symbolic Computation, 7th International Conference, AISC 2004, Linz, Austria, September 22-24, 2004, Proceedings*. Ed. by Bruno Buchberger and John A. Campbell. Vol. 3249. Lecture Notes in Computer Science. Springer, 2004, pp. 157–170. DOI: 10.1007/978-3-540-30210-0_14.
- [29] Temur Kutsia. “Solving equations with sequence variables and sequence functions”. In: *J. Symb. Comput.* 42.3 (2007), pp. 352–388. DOI: 10.1016/j.jsc.2006.12.002.
- [30] Temur Kutsia, Jordi Levy, and Mateu Villaret. “Anti-unification for Unranked Terms and Hedges”. In: *J. Autom. Reason.* 52.2 (2014), pp. 155–190. DOI: 10.1007/s10817-013-9285-6.
- [31] Temur Kutsia and Mircea Marin. “Can Context Sequence Matching be Used for Querying XML?” In: *Proceedings of the 19th International Workshop on Unification (UNIF’05)*. Ed. by Laurent Vigneron. Nara, Japan, 2005, pp. 77–92.
- [32] Temur Kutsia and Mircea Marin. “Solving, Reasoning, and Programming in Common Logic”. In: *14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2012, Timisoara, Romania, September 26-29, 2012*. Ed. by Andrei Voronkov et al. IEEE Computer Society, 2012, pp. 119–126. DOI: 10.1109/SYNASC.2012.27.
- [33] Temur Kutsia and Cleo Pau. “A Framework for Approximate Generalization in Quantitative Theories”. In: *Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings*. Ed. by Jasmin Blanchette, Laura Kovács, and Dirk Pattinson. Vol. 13385. Lecture Notes in Computer Science. Springer, 2022, pp. 578–596. DOI: 10.1007/978-3-031-10769-6_34.

- [34] Temur Kutsia and Cleo Pau. “Matching and Generalization Modulo Proximity and Tolerance Relations”. In: *Language, Logic, and Computation - 13th International Tbilisi Symposium, TbiLLC 2019, Batumi, Georgia, September 16-20, 2019, Revised Selected Papers*. Ed. by Aybüke Özgün and Yulia Zinova. Vol. 13206. Lecture Notes in Computer Science. Springer, 2019, pp. 323–342. DOI: 10.1007/978-3-030-98479-3_16.
- [35] Temur Kutsia and Cleo Pau. “Solving Proximity Constraints”. In: *Logic-Based Program Synthesis and Transformation - 29th International Symposium, LOPSTR 2019, Porto, Portugal, October 8-10, 2019, Revised Selected Papers*. Ed. by Maurizio Gabrielli. Vol. 12042. Lecture Notes in Computer Science. Springer, 2019, pp. 107–122. DOI: 10.1007/978-3-030-45260-5_7.
- [36] Vincenzo Loia, Sabrina Senatore, and Maria I. Sessa. “Similarity-based SLD resolution and its role for web knowledge discovery”. In: *Fuzzy Sets Syst.* 144.1 (2004), pp. 151–171. DOI: 10.1016/J.FSS.2003.10.018.
- [37] Jaron Maene and Luc De Raedt. “Soft-Unification in Deep Probabilistic Logic”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023, pp. 60804–60820. URL: <https://dl.acm.org/doi/10.5555/3666122.3668778>.
- [38] Gennady S. Makanin. “The problem of solvability of equations in a free semigroup”. In: *Mathematics of the USSR – Sbornik* 32.2 (1977), pp. 129–198. DOI: 10.1070/SM1977v032n02ABEH002376.
- [39] Mircea Marin and Temur Kutsia. “Foundations of the rule-based system ρLog ”. In: *J. Appl. Non Class. Logics* 16.1-2 (2006), pp. 151–168. DOI: 10.3166/JANCL.16.151-168.
- [40] Jesús Medina, Manuel Ojeda-Aciego, and Peter Vojtás. “Similarity-based unification: a multi-adjoint approach”. In: *Fuzzy Sets Syst.* 146.1 (2004), pp. 43–62. DOI: 10.1016/J.FSS.2003.11.005.
- [41] Sonu Mehta et al. “Rex: Preventing Bugs and Misconfiguration in Large Services Using Correlated Change Analysis”. In: *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*. Ed. by Ranjita Bhagwan and George Porter. USENIX Association, 2020, pp. 435–448. URL: <https://www.usenix.org/conference/nsdi20/presentation/mehta>.
- [42] Gian Carlo Milanese and Gabriella Pasi. “Similarity-Based Reasoning With Order-Sorted Feature Logic”. In: *IEEE Trans. Fuzzy Syst.* 32.5 (2024), pp. 2797–2810. DOI: 10.1109/TFUZZ.2024.3362897.
- [43] Cleo Pau. “Symbolic Techniques for Approximate Reasoning”. PhD thesis. RISC, Johannes Kepler University Linz, 2022. URL: <https://epub.jku.at/obvulihs/content/titleinfo/7818355>.

- [44] Cleo Pau and Temur Kutsia. “Proximity-Based Unification and Matching for Fully Fuzzy Signatures”. In: *30th IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2021, Luxembourg, July 11-14, 2021*. IEEE, 2021, pp. 1–6. DOI: 10.1109/FUZZ45933.2021.9494438.
- [45] Wojciech Plandowski. “On *PSPACE* generation of a solution set of a word equation and its applications”. In: *Theor. Comput. Sci.* 792 (2019), pp. 20–61. DOI: 10.1016/J.TCS.2018.10.023.
- [46] Julian Richardson and Norbert E. Fuchs. “Development of Correct Transformation Schemata for Prolog Programs”. In: *Logic Programming Synthesis and Transformation, 7th International Workshop, LOPSTR’97, Leuven, Belgium, July 10-12, 1997, Proceedings*. Ed. by Norbert E. Fuchs. Vol. 1463. Lecture Notes in Computer Science. Springer, 1997, pp. 263–281. DOI: 10.1007/3-540-49674-2\14.
- [47] Reudismam Rolim de Sousa et al. “Learning Quick Fixes from Code Repositories”. In: *35th Brazilian Symposium on Software Engineering, SBES 2021, Joinville, Santa Catarina, Brazil, 27 September 2021 - 1 October 2021*. Ed. by Cristiano D. Vasconcellos et al. ACM, 2021, pp. 74–83. DOI: 10.1145/3474624.3474650.
- [48] Ralf Schweimeier and Michael Schroeder. “Fuzzy Unification and Argumentation for Well-Founded Semantics”. In: *SOFSEM 2004: Theory and Practice of Computer Science, 30th Conference on Current Trends in Theory and Practice of Computer Science, Merin, Czech Republic, January 24-30, 2004*. Ed. by Peter van Emde Boas et al. Vol. 2932. Lecture Notes in Computer Science. Springer, 2004, pp. 102–121. DOI: 10.1007/978-3-540-24618-3\9.
- [49] Maria I. Sessa. “Approximate reasoning by similarity-based SLD resolution”. In: *Theor. Comput. Sci.* 275.1-2 (2002), pp. 389–426. DOI: 10.1016/S0304-3975(01)00188-8.
- [50] Peter Vojtás. “Declarative and procedural semantics of fuzzy similarity based unification”. In: *Kybernetika* 36.6 (2000), pp. 707–720. URL: <http://www.kybernetika.cz/content/2000/6/707>.
- [51] Peter Vojtás. “Fuzzy logic programming”. In: *Fuzzy Sets Syst.* 124.3 (2001), pp. 361–370. DOI: 10.1016/S0165-0114(01)00106-3.
- [52] Leon Weber et al. “NLProlog: Reasoning with Weak Unification for Question Answering in Natural Language”. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez. Association for Computational Linguistics, 2019, pp. 6151–6161. DOI: 10.18653/V1/P19-1618.
- [53] Emily Rowan Winter et al. “Towards developer-centered automatic program repair: findings from Bloomberg”. In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*,

- ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*. Ed. by Abhik Roychoudhury, Cristian Cadar, and Miryung Kim. ACM, 2022, pp. 1578–1588. DOI: 10.1145/3540250.3558953.
- [54] Stephen Wolfram. *The Mathematica book, 5th Edition*. Wolfram-Media, 2003. ISBN: 978-1-57955-022-6.
- [55] Akihiro Yamamoto et al. “Modelling Semi-structured Documents with Hedges for Deduction and Induction”. In: *Inductive Logic Programming, 11th International Conference, ILP 2001, Strasbourg, France, September 9-11, 2001, Proceedings*. Ed. by Céline Rouveirol and Michèle Sebag. Vol. 2157. Lecture Notes in Computer Science. Springer, 2001, pp. 240–247. DOI: 10.1007/3-540-44797-0_20.