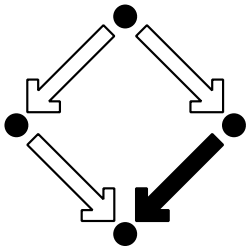
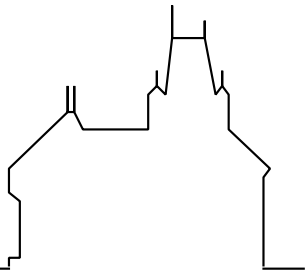


RISC-Linz

Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria, Europe



Generalization Algorithms with Proximity Relations in Full Fuzzy Signatures

Temur Kutsia, Cleo Pau

April 2021

RISC-Linz Report Series No. 21-09

Generalization Algorithms with Proximity Relations in Full Fuzzy Signatures

Temur Kutsia
kutsia@risc.jku.at

Cleo Pau
ipau@risc.jku.at

RISC, Johannes Kepler University Linz

Abstract

Anti-unification aims at computing generalizations for given terms, retaining their common structure and abstracting differences by variables. We study anti-unification for full fuzzy signatures, where the notion of common structure is relaxed into a "proximal" one with respect to a given proximity relation. Mismatches between both symbol names and their arities are permitted. We develop algorithms for different cases of the problem and study their properties.

1 Introduction

Generalization problems play an important role in various areas of mathematics, computer science, and artificial intelligence. They are closely related to detecting similarities between different objects and to learning general structures from concrete instances. Anti-unification [9, 10] is a logic-based method for computing generalizations, with a wide range of applications, see, e.g., software analysis [4, 8] or natural language processing [2, 5] for some recent examples.

Extending anti-unification techniques from a crisp to a fuzzy setting is interesting from a pragmatic point of view because of its application potential. For instance, software code clone detection, indexing (noisy) structures, or developing chatbots would benefit from them. It is also a theoretically interesting and challenging task, since the existing crisp techniques do not directly extend to fuzzy generalizations. Investigations to this direction have been started only recently. In [1], the authors proposed an anti-unification algorithm for fuzzy similarity (reflexive, symmetric, min-transitive) relations, where mismatches are allowed not only in symbol names, but also in their arities (full fuzzy signatures). The algorithm from [7] is designed for fuzzy proximity (i.e., reflexive and symmetric) relations with mismatches in symbol names.

In this paper, we study the fuzzy anti-unification problem in a more general setting. The considered relations are proximity relations. Mismatches are allowed both in symbol names and their arities. In the latter case, we consider four different variants of relating arguments between different proximal symbols: unrestricted relations / functions, and correspondence (i.e. left- and right-total) relations / functions. We design the corresponding algorithms, study their properties, and show how to obtain the existing fuzzy anti-unification problems as special cases of our problems.

2 Preliminaries

Proximity relations

We define the basic notions about proximity relations according to [6]. Given a set S, a mapping from $S \times S$ to the real interval $[0, 1]$ is called a binary *fuzzy relation* on S. By fixing a number λ, $0 \leq \lambda \leq 1$, we can define the crisp counterpart of $\mathcal{R}$, named the λ-*cut* of $\mathcal{R}$ on S, as $\mathcal{R}_\lambda := \{(s_1, s_2) \mid \mathcal{R}(s_1, s_1) \geq \lambda\}$. We take the minimum as the T-norm $\wedge$. A *proximity relation* on a set S is a reflexive and symmetric fuzzy relation $\mathcal{R}$ on S.

Terms and substitutions

We consider a first-order alphabet consisting of a set of fixed arity function symbols $\mathcal{F}$ and a set of variables $\mathcal{V}$, which includes a special symbol $\_$ (the anonymous variable). The set of non-anonymous variables $\mathcal{V} \setminus \{\_\}$ is denoted by $\mathcal{V}^-$. When the set of variables is not explicitly specified, we mean $\mathcal{V}$.

The set of terms $\mathcal{T}(\mathcal{F}, \mathcal{V})$ over $\mathcal{F}$ and $\mathcal{V}$ is defined in the standard way: $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ iff t is defined by the grammar $t := x \mid f(t_1, \ldots, t_n)$, where $x \in \mathcal{V}$ and $f \in \mathcal{F}$ is an n-ary function symbol with $n \geq 0$. Terms over $\mathcal{T}(\mathcal{F}, \mathcal{V}^-)$ are defined similarly except that all variables are taken from $\mathcal{V}^-$.

We denote arbitrary function symbols by f, g, h, constants by a, b, c, variables by x, y, z, v, and terms by s, t, r. The *head* of a term is defined as $\mathit{head}(x) := x$ and $\mathit{head}(f(t_1, \ldots, t_n)) := f$. For a term t, we denote with $\mathcal{V}(t)$ (resp. by $\mathcal{V}^-(t)$) the set of all variables (resp. all non-anonymous variables) appearing in t.

The deanonymization operation *deanon* replaces each occurrence of the anonymous variable in a term by a fresh variable. For instance, we have $\mathit{deanon}(f(\_, x, g(\_))) = f(y', x, g(y''))$, where y' and y'' are fresh. Hence, $\mathit{deanon}(t) \in \mathcal{T}(\mathcal{F}, \mathcal{V}^-)$ is unique up to variable renaming for all $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$.

The notions of *term depth* and a *position* in a term are defined in the standard way, see, e.g. [3]. By $t|_p$ we denote the subterm of t at position p. For instance, $f(a, g(\_, x))|_\epsilon = f(a, g(\_, x))$, $f(a, g(\_, x))|_1 = a$, $f(a, g(\_, x))|_2 = g(\_, x)$, $f(a, g(\_, x))|_{2.1} = \_$, and $f(a, g(\_, x))|_{2.2} = x$. We

denote by $t[s]_p$ a term that is obtained from t by replacing the subterm at position p by the term s.

A *substitution* is a mapping from $\mathcal{V}^-$ to $\mathcal{T}(\mathcal{F}, \mathcal{V}^-)$ (i.e., without anonymous variables), which is the identity almost everywhere. We use the Greek letters $\sigma, \vartheta, \varphi$ to denote substitutions, except for the identity substitution which is written as Id. We represent substitutions with the usual set notation: σ is represented as $\{x \mapsto \sigma(x) \mid x \neq \sigma(x)\}$. The restriction of a substitution σ on a set of (non-anonymous) variables V, denoted by $\sigma|_V$, is the substitution defined as $\sigma|_V(x) = \sigma(x)$ when $x \in V$ and $\sigma|_V(x) = x$ otherwise.

Application of a substitution σ to a term t, denoted by $t\sigma$, is defined as $\_\sigma := \_$, $x\sigma := \sigma(x)$ and $f(t_1, \ldots, t_n)\sigma := f(t_1\sigma, \ldots, t_n\sigma)$. Substitution *composition* is defined as a composition of mappings, and we write $\sigma\vartheta$ for the composition of σ with ϑ. The operation is associative but not commutative.

Argument relations and mappings

Given two sets $N = \{1, \ldots, n\}$ and $M = \{1, \ldots, m\}$, a binary *argument relation* over $N \times M$ is a (possibly empty) subset of $N \times M$. We denote argument relations by ρ.

An *argument mapping* is an argument relation that is an injective function. That implies that an argument mapping π from $N = \{1, \ldots, n\}$ to $M = \{1, \ldots, m\}$ is a function $\pi : I_n \mapsto I_m$, where $I_n \subseteq N$, $I_m \subseteq M$ and $|I_n| = |I_m|$. Note that it can be also the empty mapping: $\pi : \emptyset \mapsto \emptyset$. Usually, for $\pi : I_n \mapsto I_m$ we write $\pi = \{i \mapsto \pi(i) \mid i \in I_n\}$. The inverse of an argument mapping is again an argument mapping.

Given a proximity relation $\mathcal{R}$ over $\mathcal{F}$, we assume that for any pair of function symbols f and g with $\mathcal{R}(f, g) = \alpha > 0$, where f is n-ary and g is m-ary, there is an argument relation ρ over $\{1, \ldots, n\} \times \{1, \ldots, m\}$. We use the notation $f \sim^{\rho}_{\mathcal{R},\alpha} g$. Note that ρ is the empty relation if f or g is a constant, and the identity relation if $f = g$. Moreover, $f \sim^{\rho}_{\mathcal{R},\alpha} g$ iff $g \sim^{\rho^{-1}}_{\mathcal{R},\alpha} f$, where ρ^{-1} is the inverse of ρ.

An argument relation $\rho \subseteq N \times M$ is (i) *left-total* if for all $i \in N$ there exists $j \in M$ such that $(i, j) \in \rho$; (ii) *right-total* if for all $j \in M$ there exists $i \in N$ such that $(i, j) \in \rho$. *Correspondence relations* are those that are both left- and right-total.

Proximity relations over terms

Each proximity relation $\mathcal{R}$ in this paper is defined on $\mathcal{F} \cup \mathcal{V}$ such that $\mathcal{R}(f, x) = 0$ for all $f \in \mathcal{F}$ and $x \in \mathcal{V}$, and $\mathcal{R}(x, y) = 0$ for all $x \neq y$, $x, y \in \mathcal{V}$. We assume that $\mathcal{R}$ is *strict*: for all $w_1, w_2 \in \mathcal{F} \cup \mathcal{V}$, if $\mathcal{R}(w_1, w_2) = 1$, then $w_1 = w_2$. Yet another assumption is that for each $f \in \mathcal{F}$, its $(\mathcal{R}, \lambda)$-proximity class $\{g \mid \mathcal{R}(f, g) \geq \lambda\}$ is finite for any $\mathcal{R}$ and λ.

We extend such an $\mathcal{R}$ to terms from $\mathcal{T}(\mathcal{F},\mathcal{V})$ as follows:

(a) $\mathcal{R}(t,s) := 0$ if $\mathcal{R}(head(s), head(t)) = 0$;

(b) $\mathcal{R}(t,s) := 1$ if $t = s$ and $t, s \in \mathcal{V}$;

(c) $\mathcal{R}(t,s) := \mathcal{R}(f,g) \wedge \mathcal{R}(t_{i_1}, s_{j_1}) \wedge \cdots \wedge \mathcal{R}(t_{i_k}, s_{j_k})$, if $t = f(t_1,\ldots,t_n)$, $s = g(s_1,\ldots,s_m)$, $f \sim^{\rho}_{\mathcal{R},\lambda} g$, and $\rho = \{(i_1,j_1),\ldots,(i_k,j_k)\}$.

If $\mathcal{R}(t,s) \geq \lambda$, we write $t \simeq_{\mathcal{R},\lambda} s$. When $\lambda = 1$, the relation $\simeq_{\mathcal{R},\lambda}$ does not depend on $\mathcal{R}$ due to strictness of the latter and is just the syntactic equality $=$.

Anti-unification problems, generalizations

Given $\mathcal{R}$ and λ, a term r is an $(\mathcal{R},\lambda)$-*generalization* of (alternatively, $(\mathcal{R},\lambda)$-*more general than*) a term t, written as $r \precsim_{\mathcal{R},\lambda} t$, if there exists a substitution σ such that $deanon(r)\sigma \simeq_{\mathcal{R},\lambda} deanon(t)$. The strict part of $\precsim_{\mathcal{R},\lambda}$ is denoted by $\prec_{\mathcal{R},\lambda}$, i.e., $r \prec_{\mathcal{R},\lambda} t$ if $r \precsim_{\mathcal{R},\lambda} t$ and not $t \precsim_{\mathcal{R},\lambda} r$.

Example 1. *Given a proximity relation $\mathcal{R}$, a cut value $\lambda = 0.3$, constants $a \sim^{\emptyset}_{\mathcal{R},0.4} b$ and $b \sim^{\emptyset}_{\mathcal{R},0.5} c$, binary function symbols f and h, and a unary function symbol g such that $h \sim^{\{(1,1),(1,2)\}}_{\mathcal{R},0.5} f$ and $h \sim^{\{(1,1)\}}_{\mathcal{R},0.6} g$, we have*

- $h(x,\_) \precsim_{\mathcal{R},\lambda} h(a,x)$, *because* $h(x,x')\{x \mapsto a, x' \mapsto x\} = h(a,x) \simeq_{\mathcal{R},\lambda} h(a,x)$.
- $h(x,\_) \precsim_{\mathcal{R},\lambda} h(\_,x)$, *because* $h(x,x')\{x \mapsto y', x' \mapsto x\} \simeq_{\mathcal{R},\lambda} h(y',x)$.
- $h(x,x) \not\precsim_{\mathcal{R},\lambda} h(\_,x)$, *because* $h(x,x) \not\precsim_{\mathcal{R},\lambda} h(y',x)$.
- $h(x,\_) \precsim_{\mathcal{R},\lambda} f(a,c)$, *because* $h(x,x')\{x \mapsto b\} = h(b,x') \simeq_{\mathcal{R},\lambda} f(a,c)$, *since* $\mathcal{R}(h(b,x'), f(a,c)) = 0.4$.
- $h(x,\_) \precsim_{\mathcal{R},\lambda} g(c)$, *because* $h(x,x')\{x \mapsto c\} = h(c,x') \simeq_{\mathcal{R},\lambda} g(c)$, *since* $\mathcal{R}(h(c,x'), g(c)) = 0.6$.

The notion of *syntactic generalization* of a term is a special case of $(\mathcal{R},\lambda)$-generalization for $\lambda = 1$. We write $r \precsim t$ to indicate that r is a syntactic generalization of t. Its strict part is denoted by $\prec$.

Since $\mathcal{R}$ is strict, $r \precsim t$ is equivalent to $deanon(r)\sigma = deanon(t)$ for some σ (note the syntactic equality here).

Theorem 1. *If $r \precsim t$ and $t \precsim_{\mathcal{R},\lambda} s$, then $r \precsim_{\mathcal{R},\lambda} s$.*

Proof. $r \precsim t$ implies $deanon(r)\sigma = deanon(t)$ for some σ, while from $t \precsim_{\mathcal{R},\lambda} s$ we have $deanon(t)\vartheta \simeq_{\mathcal{R},\lambda} deanon(s)$ for some ϑ. Then $deanon(r)\sigma\vartheta \simeq_{\mathcal{R},\lambda} deanon(s)$, which implies $r \precsim_{\mathcal{R},\lambda} s$. □

Note that $r \precsim_{\mathcal{R},\lambda} t$ and $t \precsim_{\mathcal{R},\lambda} s$, in general, do not imply $r \precsim_{\mathcal{R},\lambda} s$ due to non-transitivity of $\simeq_{\mathcal{R},\lambda}$. A simple counterexample: $a \precsim_{\mathcal{R},0.3} b$, $b \precsim_{\mathcal{R},0.3} c$, but not $a \precsim_{\mathcal{R},0.3} c$.

Definition 1 (Minimal complete set of $(\mathcal{R},\lambda)$-generalizations)**.** *Given $\mathcal{R}$, λ, t_1, and t_2, a set of terms T is a* complete set of $(\mathcal{R},\lambda)$-generalizations *of t_1 and t_2 if*

(a) every $r \in T$ is an $(\mathcal{R},\lambda)$-generalization of t_1 and t_2,

(b) if r' is an $(\mathcal{R},\lambda)$-generalization of t_1 and t_2, then there exists $r \in T$ such that $r' \precsim r$ (note that we use syntactic generalization here).

In addition, T is minimal, if it satisfies the following property:

(c) if $r, r' \in T$, $r \neq r'$, then neither $r \prec_{\mathcal{R},\lambda} r'$ nor $r' \prec_{\mathcal{R},\lambda} r$.

A minimal complete set of $(\mathcal{R},\lambda)$-generalizations *($(\mathcal{R},\lambda)$-mcsg) of two terms is unique modulo variable renaming. The elements of the $(\mathcal{R},\lambda)$-mcsg of t_1 and t_2 are called least general $(\mathcal{R},\lambda)$-generalizations ($(\mathcal{R},\lambda)$-lggs) of t_1 and t_2.*

This definition directly extends to generalizations of finitely many terms.

The problem of computing an $(\mathcal{R},\lambda)$-generalization of terms t and s is called the $(\mathcal{R},\lambda)$-anti-unification problem of t and s. It is dual to unification, where one aims at computing a most general $(\mathcal{R},\lambda)$-unifier (i.e., a substitution) of the given terms. In anti-unification, the goal is to compute their least general $(\mathcal{R},\lambda)$-generalization (i.e., a term).

Ideally, the precise formulation of anti-unification problem would be the following: Given $\mathcal{R}$, λ, t_1 and t_2, find an $(\mathcal{R},\lambda)$-lgg r of t_1 and t_2, some substitutions σ_1 and σ_2, and the approximation degrees α_1 and α_2 such that $\mathcal{R}(r\sigma_1, t) = \alpha_1$ and $\mathcal{R}(r\sigma_2, s) = \alpha_2$. A minimal and complete algorithm to solve this problem would compute exactly the elements of $(\mathcal{R},\lambda)$-mcsg of t and s together with their approximation degrees. However, as we see below, it is problematic to solve the problem in this form. Therefore, we will consider a slightly modified variant, taking into account anonymous variables in generalizations and relaxing bounds on approximation degrees.

3 Generalization problems

We assume that terms to be generalized are ground. It is not a restriction because we can treat variables in them as constants that are close only to themselves.

Recall that the proximity class of any alphabet symbol is finite. Also, the symbols are related to each other by finitely many argument relations. One may think that it leads to finite proximity classes of terms, but this

is not the case. Consider, e.g., $\mathcal{R}$ and λ, where $h \simeq_{\mathcal{R},\lambda}^{\{(1,1)\}} f$ with binary h and unary f. Then the $(\mathcal{R},\lambda)$-proximity class of $f(a)$ is infinite: $\{f(a)\} \cup \{h(a,t) \mid t \in \mathcal{T}(\mathcal{F},\mathcal{V})\}$. Also, the $(\mathcal{R},\lambda)$-mcsg for $f(a)$ and $f(b)$ is infinite: $\{f(x)\} \cup \{h(x,t) \mid t \in \mathcal{T}(\mathcal{F},\emptyset)\}$.

Definition 2. *Given the terms $t_1,\ldots,t_n$, $n \geq 1$, a position p in a term r is called* irrelevant for $(\mathcal{R},\lambda)$-generalizing *(resp.* for $(\mathcal{R},\lambda)$-proximity to*)* $t_1,\ldots,t_n$ *if $r[s]_p \precsim_{\mathcal{R},\lambda} t_i$ (resp. $r[s]_p \simeq_{\mathcal{R},\lambda} t_i$) for all $1 \leq i \leq n$ and for any term s.*

We say that r is a relevant $(\mathcal{R},\lambda)$-generalization *(resp.* relevant $(\mathcal{R},\lambda)$-proximal term*) of $t_1,\ldots,t_n$ if $r|_p = \_$ for any position p in r that is irrelevant for generalizing (resp. for proximity to) $t_1,\ldots,t_n$. The* $(\mathcal{R},\lambda)$-relevant proximity class *of t is*

$$\mathbf{rpc}_{\mathcal{R},\lambda}(t) := \{s \mid s \text{ is a relevant } (\mathcal{R},\lambda)\text{-proximal term of } t\}.$$

In the example above, position 2 in $h(x,t)$ is irrelevant for generalizing $f(a)$ and $f(b)$, and $h(x,\_)$ is one of their relevant generalizations. Note that $f(x)$ is also a relevant generalization of $f(a)$ and $f(b)$, since it contains no irrelevant positions. More general generalizations like, e.g., x, are relevant as well. Similarly, position 2 in $h(a,t)$ is irrelevant for proximity to $f(a)$ and $\mathbf{rpc}_{\mathcal{R},\lambda}(f(a)) = \{f(a), h(a,\_)\}$. Generally, $\mathbf{rpc}_{\mathcal{R},\lambda}(t)$ is finite for any t due to the finiteness of proximity classes of symbols and argument relations mentioned above.

Definition 3 (Minimal complete set of relevant $(\mathcal{R},\lambda)$-generalizations)**.** *Given $\mathcal{R}$, λ, t_1, and t_2, a set of terms T is a* complete set of relevant $(\mathcal{R},\lambda)$-generalizations *of t_1 and t_2 if*

(a) every element of T is a relevant $(\mathcal{R},\lambda)$-generalization of t_1 and t_2, and

(b) if r is a relevant $(\mathcal{R},\lambda)$-generalization of t_1 and t_2, then there exists $r' \in T$ such that $r \precsim r'$.

The minimality property is defined as in Definition 1.

This definition directly extends to relevant generalizations of finitely many terms. We use $(\mathcal{R},\lambda)$-mcsrg as an abbreviation for minimal complete set of relevant $(\mathcal{R},\lambda)$-generalization. Like relevant proximity classes, mcsrg's are also finite.

Lemma 1. *For given $\mathcal{R}$ and λ, if all argument relations are correspondence relations, then $(\mathcal{R},\lambda)$-mcsg's and $(\mathcal{R},\lambda)$-proximity classes for all terms are finite.*

Proof. Under correspondence relations no term contains an irrelevant position for generalization or for proximity. □

Hence, for correspondence relations the notions of mcsg and mcsrg coincide, as well as the notions of proximity class and relevant proximity class.

Definition 4 (Consistent set of terms)**.** *A set of terms T is* $(\mathcal{R},\lambda)$-consistent *if* $\bigcap_{t\in T}\mathbf{rpc}_{\mathcal{R},\lambda}(t)\neq\emptyset$.

For a term r, we define its *linearized version* $lin(r)$ as a term obtained from r by replacing each occurrence of a non-anonymous variable in r by a fresh one. Linearized versions of terms are unique modulo variable renaming. For instance, $lin(f(x,\_,g(y,x,a),b)) = f(x',\_,g(y',x'',a),b)$, where x', x'', and y' are fresh variables.

Definition 5 (Linear generalization degree)**.** *For two terms r and t such that $r \precsim_{\mathcal{R},\lambda} t$, the* linear $(\mathcal{R},\lambda)$-generalization degree $lgd_{\mathcal{R},\lambda}(r,t)$ *is the smallest number $\alpha\in(0,1]$ such that $\lambda \le \mathcal{R}(lin(r)\sigma,t)\le\alpha$ for any substitution σ.*

Example 2. *Let $\mathcal{R}(a,b)=0.6$, $\mathcal{R}(b,c)=0.7$, and $\lambda=0.5$. Then we have $lgd_{\mathcal{R},\lambda}(f(x,b),f(a,c)) = 0.7$ and $lgd_{\mathcal{R},\lambda}(f(x,x),f(a,c)) = lgd_{\mathcal{R},\lambda}(f(x,y),f(a,c))=1$.*

It is not difficult to see that if σ is a substitution such that $s\sigma\simeq_{\mathcal{R},\lambda} t$, then $\mathcal{R}(s\sigma,t)\le lgd_{\mathcal{R},\lambda}(s,t)$. In Example 2, for $\sigma=\{x\mapsto b\}$ we have $\mathcal{R}(f(x,x)\sigma,f(a,c)) = \mathcal{R}(f(b,b),f(a,c)) = 0.6 < lgd_{\mathcal{R},\lambda}(f(x,x),f(a,c)) = 1$.

Given $r\preceq_{\mathcal{R},\lambda} t$, we can compute $lgd_{\mathcal{R},\lambda}(r,t)$ as follows:

- If r is a variable, then $lgd_{\mathcal{R},\lambda}(r,t)=1$.
- Otherwise, if $head(r)\sim^{\rho}_{\mathcal{R},\beta} head(t)$, then

$$lgd_{\mathcal{R},\lambda}(r,t)=\beta\wedge\bigwedge_{(i,j)\in\rho} lgd_{\mathcal{R},\lambda}(r|_i,t|_j).$$

Now we can reformulate anti-unification problem that will be solved in the remaining part of the paper.

Given: a proximity relation $\mathcal{R}$, a cut value λ, and the ground terms $t_1,\ldots,t_n$, $n\ge 2$.

Find: a set S of tuples $(r,\sigma_1,\ldots,\sigma_n,\alpha_1,\ldots,\alpha_n)$ such that

- $\{r \mid (r,\ldots)\in\mathsf{S}\}$ is a minimal complete set of relevant $(\mathcal{R},\lambda)$-generalizations of $t_1,\ldots,t_n$,
- $r\sigma_i\simeq_{\mathcal{R},\lambda} t_i$ and $\alpha_i = lgd_{\mathcal{R},\lambda}(r,t_i)$, $1\le i\le n$, for each $(r,\sigma_1,\ldots,\sigma_n,\alpha_1,\ldots,\alpha_n)\in\mathsf{S}$.

(Note that when $n=1$, this is a problem of computing a relevant proximity class of a term.) Below we solve the anti-unification problem for four versions of argument relations:

1. the most general (unrestricted) case; see algorithm $\mathfrak{A}_1$ below, the computed set of generalizations is an mcsrg;
2. correspondence relations: using the same algorithm $\mathfrak{A}_1$, the computed set of generalizations is an mcsg;
3. argument mappings: using a dedicated algorithm $\mathfrak{A}_2$, the computed set of generalizations is an mcsrg;
4. correspondence mappings (bijections): using the algorithm $\mathfrak{A}_2$, the computed set of generalizations is an mcsg.

For simplicity, we formulate the algorithms for the case $n = 2$. They can be extended for arbitrary n straightforwardly.

The main data structure in these algorithms is an anti-unification triple (AUT) $x : T_1 \triangleq T_2$, where T_1 and T_2 are finite *consistent* sets of ground terms. A configuration is a tuple $A; S; r; \alpha_1; \alpha_2$, where A is a set of AUTs to be solved, S is a set of solved AUTs (the store), r is the generalization computed so far, and the α's are the current approximations of linear generalization degrees of r for the input terms.

3.1 Anti-Unification for Unrestricted Argument Relations

Algorithm $\mathfrak{A}_1$ uses the rules below to transform configurations into configurations. Given $\mathcal{R}$, λ, and the ground terms t_1 and t_2, we create the initial configuration $\{x : \{t_1\} \triangleq \{t_2\}\}; \emptyset; x; 1; 1$ and apply the rules as long as possible. Note that the rules preserve consistency of AUTs. The process generates a finite complete tree of derivations, whose terminal nodes have configurations with the first component empty. We will show how from these terminal configurations one collects the result as required in the anti-unification problem statement.

Tri: **Trivial**

$\{x : \emptyset \triangleq \emptyset\} \uplus A;\ S;\ r;\ \alpha_1; \alpha_2 \Longrightarrow A;\ S;\ r\{x \mapsto \_\};\ \alpha_1; \alpha_2.$

Dec: **Decomposition**

$$\{x : T_1 \triangleq T_2\} \uplus A; S; r; \alpha_1; \alpha_2 \Longrightarrow \\ \{y_i : Q_{i1} \triangleq Q_{i2} \mid 1 \le i \le n\} \cup A; S; \\ r\{x \mapsto h(y_1, \ldots, y_n)\}; \alpha_1 \wedge \beta_1; \alpha_2 \wedge \beta_2,$$

where $T_1 \cup T_2 \neq \emptyset$; h is n-ary with $n \ge 0$; $y_1, \ldots, y_n$ are fresh; and for $j = 1, 2$, if $T_j = \{t^j_1, \ldots, t^j_{m_j}\}$, then

- $h \sim^{\rho^j_k}_{\mathcal{R},\gamma^j_k} head(t^j_k)$ with $\gamma^j_k \ge \lambda$ for all $1 \le k \le m_j$ and $\beta_j = \gamma^j_1 \wedge \cdots \wedge \gamma^j_{m_j}$ (note that $\beta_j = 1$ if $m_j = 0$),
- for all $1 \le i \le n$, $Q_{ij} = \cup_{k=1}^{m_j} \{t^j_k|_q \mid (i, q) \in \rho^j_k\}$ and is $(\mathcal{R}, \lambda)$-consistent.

Sol: **Solving**

$\{x : T_1 \triangleq T_2\} \uplus A;\ S;\ r;\ \alpha_1; \alpha_2 \Longrightarrow A;\ \{x : T_1 \triangleq T_2\} \cup S;\ r;\ \alpha_1; \alpha_2,$

if Tri and Dec rules are not applicable. (It means that at least one $T_i \neq \emptyset$ and either there is no h as it is required in the Dec rule, or at least one Q_{ij} from Dec is not consistent.)

Mer: **Merge**

$\emptyset; \{x_i : T_{i1} \triangleq T_{i2} \mid 1 \leq i \leq k\} \uplus S; r; \alpha_1; \alpha_2 \Longrightarrow$
$\emptyset; \{y : Q_1 \triangleq Q_2\} \cup S; r\sigma; \alpha_1; \alpha_2,$

where for $j = 1, 2$,

- $Q_j = \bigcap_{t \in T_j} \mathbf{rpc}_{\mathcal{R},\lambda}(t) \neq \emptyset$, where $T_j = \cup_{i=1}^{k} T_{ij}$,
- y is fresh and $\sigma = \{x_i \mapsto y \mid 1 \leq i \leq k\}$,
- for each $x' : T'_1 \triangleq T'_2 \in S$ there exist $1 \leq i \leq k$ such that $\mathcal{R}(t', t) < \lambda$ for some $t' \in T'_j$ and $t \in T_{ij}$.

To a store $S = \{y_1 : Q_{11} \triangleq Q_{12}, \ldots, y_n : Q_{n1} \triangleq Q_{n2}\}$ we associate two sets of substitutions $\Sigma_L(S)$ and $\Sigma_R(S)$, defined as follows: $\sigma \in \Sigma_L(S)$ (resp. $\sigma \in \Sigma_R(S)$) iff $dom(\sigma) = \{y_1, \ldots, y_n\}$ and $y_i\sigma \in Q_{i1}$ (resp. $y_i\sigma \in Q_{i2}$) for each $1 \leq i \leq n$.

From each terminal configuration $C = \emptyset; S; r; \alpha_1; \alpha_2$, we construct the set $\mathsf{S}(C) := \{(r, \sigma_1, \sigma_2, \alpha_1, \alpha_2) \mid \sigma_1 \in \Sigma_L(S), \sigma_2 \in \Sigma_R(S)\}$. Given an anti-unification problem $\mathcal{R}$, λ, t_1 and t_2, the *answer computed by* Algorithm $\mathfrak{A}_1$ is the set $\mathsf{S} := \cup_{i=1}^{m} \mathsf{S}(C_i)$, where $C_1, \ldots, C_m$ are all of the final configurations reached by $\mathfrak{A}_1$ for $\mathcal{R}$, λ, t_1, and t_2.

Example 3. *Assume a, b, c and d are constants with $b \sim^{\emptyset}_{\mathcal{R},0.5} c$, $c \sim^{\emptyset}_{\mathcal{R},0.6} d$, and f, g and h are respectively binary, ternary and quaternary function symbols with $h \sim^{\{(1,1),(3,2),(4,2)\}}_{\mathcal{R},0.7} f$ and $h \sim^{\{(1,1),(3,3)\}}_{\mathcal{R},0.8} g$. For the proximity relation $\mathcal{R}$ given in this way and $\lambda = 0.5$, Algorithm $\mathfrak{A}_1$ performs the following steps to anti-unify $f(a, b)$ and $g(a, c, d)$:*

$\{x : \{f(a, b)\} \triangleq \{g(a, c, d)\}\}; \emptyset; x; 1; 1 \Longrightarrow_{Dec}$
$\{x_1 : \{a\} \triangleq \{a\},\ x_2 : \emptyset \triangleq \emptyset,\ x_3 : \{b\} \triangleq \{d\},$
$\quad x_4 : \{b\} \triangleq \emptyset\}; \emptyset; h(x_1, x_2, x_3, x_4); 0.7; 0.8 \Longrightarrow_{Dec}$
$\{x_2 : \emptyset \triangleq \emptyset,\ x_3 : \{b\} \triangleq \{d\},\ x_4 : \{b\} \triangleq \emptyset\};$
$\quad \emptyset; h(a, x_2, x_3, x_4); 0.7; 0.8 \Longrightarrow_{Tri}$
$\{x_3 : \{b\} \triangleq \{d\},\ x_4 : \{b\} \triangleq \emptyset\};$
$\quad \emptyset; h(a, \_, x_3, x_4); 0.7; 0.8 \Longrightarrow_{Dec}$
$\{x_4 : \{b\} \triangleq \emptyset\}; \emptyset; h(a, \_, c, x_4); 0.5; 0.6.$

Here Dec *applies in two different ways, with the substitutions* $\{x_4 \mapsto b\}$ *and* $\{x_4 \mapsto c\}$*, leading to two final configurations:*

$$\emptyset; \emptyset; h(a, \_, c, b); 0.5; 0.6, \qquad \emptyset; \emptyset; h(a, \_, c, c); 0.5; 0.6.$$

If we had $h \sim_{\mathcal{R},0.7}^{\{(1,1),(1,2),(4,2)\}} f$*, then the algorithm would perform only the* Sol *step, because in the attempt to apply* Dec *to the initial configuration, the set* $Q_{11} = \{a, b\}$ *is inconsistent:* $\mathbf{rpc}_{\mathcal{R},\lambda}(a) = \{a\}$*,* $\mathbf{rpc}_{\mathcal{R},\lambda}(b) = \{b, c\}$*, and, hence,* $\mathbf{rpc}_{\mathcal{R},\lambda}(a) \cap \mathbf{rpc}_{\mathcal{R},\lambda}(b) = \emptyset$*.*

Example 4. *Assume* $a_1, a_2, b_1, b_2, c_1, c_2$ *are constants and* f, g, h *are ternary function symbols. Let* $\lambda = 0.4$ *and the proximity relation* $\mathcal{R}$ *be defined by* $a_1 \sim_{\mathcal{R},0.5}^{\emptyset} b_1$*,* $b_1 \sim_{\mathcal{R},0.5}^{\emptyset} c_1$*,* $a_2 \sim_{\mathcal{R},0.6}^{\emptyset} b_2$*,* $b_2 \sim_{\mathcal{R},0.6}^{\emptyset} c_2$*,* $h \sim_{\mathcal{R},0.7}^{\{(1,1),(1,2),(2,2),(3,3)\}} f$ *and* $h \sim_{\mathcal{R},0.8}^{\{(1,1),(2,2),(3,2),(3,3)\}} g$*. Then Algorithm* $\mathfrak{A}_1$ *performs the following steps to anti-unify* $f(a_1, b_1, c_1)$ *and* $g(a_2, b_2, c_2)$*:*

$$\begin{aligned}
&\{x : \{f(a_1, b_1, c_1)\} \triangleq \{g(a_2, b_2, c_2)\}\}; \emptyset; x; 1; 1 \Longrightarrow_{\mathsf{Dec}} \\
&\{x_1 : \{a_1, b_1\} \triangleq \{a_2\},\ x_2 : \{b_1\} \triangleq \{b_2\}, \\
&\quad x_3 : \{c_1\} \triangleq \{b_2, c_2\}\}; \emptyset; h(x_1, x_2, x_3); 0.7; 0.8 \Longrightarrow_{\mathsf{Sol}}^{3} \\
&\emptyset; \{x_1 : \{a_1, b_1\} \triangleq \{a_2\},\ x_2 : \{b_1\} \triangleq \{b_2\}, \\
&\quad x_3 : \{c_1\} \triangleq \{b_2, c_2\}\}; h(x_1, x_2, x_3); 0.7; 0.8.
\end{aligned}$$

From here we can continue either by merging x_1 *and* x_2*, or by merging* x_2 *and* x_3*. At the end, we get two final configurations:*

$$\begin{aligned}
&\emptyset; \{x_1 : \{a_1, b_1\} \triangleq \{a_2, b_2\}, x_3 : \{c_1\} \triangleq \{b_2, c_2\}\}; \\
&\quad h(x_1, x_1, x_3); 0.7; 0.8. \\
&\emptyset; \{x_1 : \{a_1, b_1\} \triangleq \{a_2\}, x_3 : \{b_1, c_1\} \triangleq \{b_2, c_2\}\}; \\
&\quad h(x_1, x_3, x_3); 0.7; 0.8.
\end{aligned}$$

Taking two substitutions from the first of them, e.g., $\sigma_1 = \{x_1 \mapsto a_1, x_3 \mapsto c_1\}$ *and* $\sigma_2 = \{x_1 \mapsto b_2, x_3 \mapsto b_2\}$*, we get:*

$$\begin{aligned}
&\mathcal{R}(h(x_1, x_1, x_3)\sigma_1,\ f(a_1, b_1, c_1)) = \\
&\mathcal{R}(h(a_1, a_1, c_1),\ f(a_1, b_1, c_1)) = 0.5 \leq 0.7, \text{ and} \\
&\mathcal{R}(h(x_1, x_1, x_3)\sigma_2,\ g(a_2, b_2, c_2)) = \\
&\mathcal{R}(h(b_2, b_2, b_2),\ g(a_2, b_2, c_2)) = 0.6 \leq 0.8.
\end{aligned}$$

Theorem 2. *Given* $\mathcal{R}$*,* λ*, and the ground terms* t_1 *and* t_2*, Algorithm* $\mathfrak{A}_1$ *terminates for the initial configuration* $\{x : \{t_1\} \triangleq \{t_2\}\}; \emptyset; x; 1; 1$ *and computes an answer set* S *such that*

1. *the set* $\{r \mid (r, \sigma_1, \sigma_2, \alpha_1, \alpha_2) \in \mathsf{S}\}$ *is an* $(\mathcal{R}, \lambda)$*-mcsrg of* t_1 *and* t_2*,*

2. *for each* $(r, \sigma_1, \sigma_2, \alpha_1, \alpha_2) \in \mathsf{S}$ *we have* $\mathcal{R}(r\sigma_i, t_i) \leq \alpha_i = lgd_{\mathcal{R},\lambda}(r, t_i)$, $i = 1, 2$.

Proof. Termination: Define the depth of an AUT $x : \{t_1, \ldots, t_m\} \triangleq \{s_1, \ldots, s_n\}$ as the depth of the term $f(g(t_1, \ldots, t_m), h(s_1, \ldots, s_n))$. The rules Tri, Dec, and Sol strictly reduce the multiset of depths of AUTs in the first component of the configurations. Mer strictly reduces the number of distinct variables in generalizations. Hence, these rules cannot be applied infinitely often and $\mathfrak{A}_1$ terminates.

In order to prove 1), we need to verify three properties:

- Soundness: If $(r, \sigma_1, \sigma_2, \alpha_1, \alpha_2) \in \mathsf{S}$, then r is a relevant $(\mathcal{R}, \lambda)$-generalization of t_1 and t_2.
- Completeness: If r' is a relevant $(\mathcal{R}, \lambda)$-generalization of t_1 and t_2, then there exists $(r, \sigma_1, \sigma_2, \alpha_1, \alpha_2) \in \mathsf{S}$ such that $r' \precsim r$.
- Minimality: If r and r' belong to two tuples from S such that $r \neq r'$, then neither $r \prec_{\mathcal{R},\lambda} r'$ nor $r' \prec_{\mathcal{R},\lambda} r$.

Soundness: We show that each rule transforms an $(\mathcal{R}, \lambda)$-generalization into an $(\mathcal{R}, \lambda)$-generalization. Since we start from a most general $(\mathcal{R}, \lambda)$-generalization of t_1 and t_2 (a fresh variable x), at the end of the algorithm we will get an $(\mathcal{R}, \lambda)$-generalization of t_1 and t_2. We also show that in this process all irrelevant positions are abstracted by anonymous variables, to guarantee that each computed generalization is relevant.

Dec: The computed h is $(\mathcal{R}, \lambda)$-close to the head of each term in $T_1 \cup T_2$. Q_{ij}'s correspond to argument relations between h and those heads, and each Q_{ij} is $(\mathcal{R}, \lambda)$-consistent, i.e., there exists a term that is $(\mathcal{R}, \lambda)$-close to each term in Q_{ij}. It implies that $x\sigma = h(y_1, \ldots, y_n)$ $(\mathcal{R}, \lambda)$-generalizes all the terms from $T_1 \cup T_2$. Note that at this stage, $h(y_1, \ldots, y_n)$ might not yet be a relevant $(\mathcal{R}, \lambda)$-generalization of T_1 and T_2: if there exists an irrelevant position $1 \leq i \leq n$ for the $(\mathcal{R}, \lambda)$-generalization of T_1 and T_2, then in the new configuration we will have an AUT $y_i : \emptyset \triangleq \emptyset$.

Tri: When Dec generates $y : \emptyset \triangleq \emptyset$, the Tri rule replaces y by _ in the computed generalization, making it relevant.

Sol does not change generalizations.

Mer merges AUTs whose terms have *nonempty* intersection of **rpc**'s. Hence, we can reuse the same variable in the corresponding positions in generalizations, i.e., Mer transforms a generalization computed so far into a less general one.

Completeness: We prove a slightly more general statement. Given two finite consistent sets of ground terms T_1 and T_2, if r' is a relevant $(\mathcal{R}, \lambda)$-generalization for all $t_1 \in T_1$ and $t_2 \in T_2$, then starting from $\{x : T_1 \triangleq T_2\}; \emptyset; x; 1; 1$, Algorithm $\mathfrak{A}_1$ computes a $(r, \sigma_1, \sigma_2, \alpha_1, \alpha_2)$ such that $r' \precsim r$.

We may assume w.l.o.g. that r' is a relevant $(\mathcal{R},\lambda)$-lgg. Due to the transitivity of $\precsim$, completeness for such an r' will imply it for all terms more general than r'.

We proceed by structural induction on r'. If r' is a (named or anonymous) variable, the statement holds. Assume $r' = h(r'_1,\ldots,r'_n)$, $T_1 = \{u_1,\ldots,u_m\}$, and $T_2 = \{w_1,\ldots,w_l\}$. Then h is such that $h \sim^{\rho_i}_{\mathcal{R},\beta_i} head(u_i)$ for all $1 \le i \le m$ and $h \sim^{\mu_j}_{\mathcal{R},\gamma_j} head(w_j)$ for all $1 \le j \le l$. Moreover, each r'_k is a relevant $(\mathcal{R},\lambda)$-generalization of $Q_{k1} = \cup_{i=1}^{m}\{u_i|_q \mid (k,q) \in \rho_i\}$ and $Q_{k2} = \cup_{j=1}^{l}\{w_j|_q \mid (k,q) \in \mu_j\}$ and, hence, Q_{k1} and Q_{k2} are $(\mathcal{R},\lambda)$-consistent. Therefore, we can perform a step by Dec, choosing $h(y_1,\ldots,y_k)$ as the generalization term and $y_i : Q_{i1} \triangleq Q_{i2}$ as the new AUTs. By the induction hypothesis, for each $1 \le i \le n$ we can compute a relevant $(\mathcal{R},\lambda)$-generalization r_i for Q_{i_1} and Q_{i2} such that $r'_i \precsim r_i$.

If r' is linear, then the combination of the current Dec step with the derivations that lead to those r_i's computes a tuple $(r,\ldots) \in \mathsf{S}$, where $r = h(r_1,\ldots,r_n)$ and, hence, $r' \precsim r$.

If r' is non-linear, assume w.l.o.g. that all occurrences of a shared variable z appear as the direct arguments of h: $z = r'_{k_1} = \cdots = r'_{k_p}$ for $1 \le k_1 < \cdots < k_p \le n$. Since r' is an lgg, Q_{k_i1} and Q_{k_i2} cannot be generalized by a non-variable term, thus, Tri and Dec are not applicable. Therefore, the AUTs $y_i : Q_{k_i1} \triangleq Q_{k_i2}$ would be transformed by Sol. Since all pairs Q_{k_i1} and Q_{k_i2}, $1 \le i \le p$, are generalized by the same variable, we have $\cap_{t\in Q_j}\mathbf{rpc}_{\mathcal{R},\lambda}(t) \neq \emptyset$, where $Q_j = \cup_{i=1}^{p} Q_{k_ij}$, $j = 1,2$. Additionally, $r'_{k_1},\ldots,r'_{k_p}$ are all occurrences of z in r'. Hence, the condition of Mer is satisfied and we can extend our derivation with this rule, obtaining $r = h(r_1,\ldots,r_n)$ with $z = r_{k_1} = \cdots = r_{k_p}$, implying $r' \precsim r$.

Minimality: Alternative generalizations are obtained by branching in Dec or Mer. If the current generalization r is transformed by Dec into two generalizations r_1 and r_2 on two branches, then $r_1 = h_1(y_1,\ldots,y_m)$ and $r_2 = h_2(z_1,\ldots,z_n)$ for some h's, and fresh y's and z's. It may happen that $r_1 \precsim_{\mathcal{R},\lambda} r_2$ or vice versa (if h_1 and h_2 are $(\mathcal{R},\lambda)$-close to each other), but neither $r_1 \prec_{\mathcal{R},\lambda} r_2$ nor $r_2 \prec_{\mathcal{R},\lambda} r_1$ holds. Hence, the set of generalizations computed before applying Mer is minimal. Mer groups AUTs together maximally, and different groupings are not comparable. Therefore, variables in generalizations are merged so that distinct generalizations are not $\prec_{\mathcal{R},\lambda}$-comparable. Hence, 1) is proven.

As for 2), for $i = 1,2$, from the construction in Dec follows $\mathcal{R}(r\sigma_i,t_i) \le \alpha_i$. Mer does not change α_i, thus, $\alpha_i = lgd_{\mathcal{R},\lambda}(r,t_i)$ also holds: α_i remains the approximation degree of a linear generalization obtained before Mer. □

3.2 Anti-Unification with Correspondence Argument Relations

Correspondence relations make sure that for a pair of proximal symbols, no argument is irrelevant for proximity. Left- and right-totality of those relations guarantee that each argument of a term is close to at least one argument of its proximal term. Consequently, in the Dec rule of $\mathfrak{A}_1$, the sets Q_{ij} never get empty. Therefore, the Tri rule becomes obsolete and no anonymous variable appears in generalizations. As a result, the $(\mathcal{R}, \lambda)$-mcsrg and the $(\mathcal{R}, \lambda)$-mcsg coincide, and the algorithm computes a solution from which we get an $(\mathcal{R}, \lambda)$-mcsg for the given anti-unification problem.

3.3 Anti-Unification with Argument Mappings

When the argument relations are mappings, we are able to design a more constructive method for computing generalizations and their degree bounds. The configurations stay the same as in Algorithm $\mathfrak{A}_1$, but the AUTs in A will contain only empty or singleton sets of terms. In the store, we may still get (after Mer) AUTs with term sets containing more than one element. Only the Dec rule differs in $\mathfrak{A}_2$:

Dec: **Decomposition**

$$\{x : T_1 \triangleq T_2\} \uplus A; S; r; \alpha_1; \alpha_2 \Longrightarrow \\ \{y_i : Q_{i1} \triangleq Q_{i2} \mid 1 \leq i \leq n\} \cup A; S; \\ r\{x \mapsto h(y_1, \ldots, y_n)\}; \alpha_1 \wedge \beta_1; \alpha_2 \wedge \beta_2,$$

where $T_1 \cup T_2 \neq \emptyset$; h is n-ary with $n \geq 0$; $y_1, \ldots, y_n$ are fresh; for $j = 1, 2$ and for all $1 \leq i \leq n$,

- if $T_j = \{t_j\}$, then $h \sim^{\pi_j}_{\mathcal{R},\beta_j} head(t_j)$, $Q_{ij} = \{t_j|_{\pi_j(i)}\}$,
- if $T_j = \emptyset$, then $\beta_j = 1$ and $Q_{ij} = \emptyset$.

This Dec rule is equivalent to the special case of Dec in $\mathfrak{A}_1$ for $m_j \leq 1$. The new Q_{ij}'s contain at most one element (due to mappings) and, thus, are always $(\mathcal{R}, \lambda)$-consistent. Various choices of h in Dec and alternatives in grouping AUTs in Mer cause branching in the same way as in $\mathfrak{A}_1$. It is easy to see that the counterpart of Theorem 2 holds for $\mathfrak{A}_2$ as well.

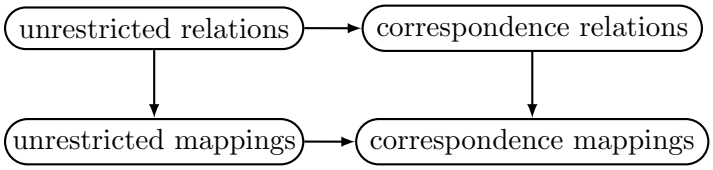
A special case of this fragment of anti-unification is anti-unification for similarity relations in full fuzzy signatures from [1]. Similarity relations are min-transitive proximity relations. The position mappings in [1] can be modeled by our argument mappings, requiring them to be total for symbols of the smaller arity.

3.4 Anti-Unification with Correspondence Argument Mappings

Correspondence argument mappings are bijections between arguments of function symbols of the same arity. For such mappings, if $h \simeq^{\pi}_{\mathcal{R},\lambda} f$ and h is n-ary, then f is also n-ary and π is a permutation of $(1, \ldots, n)$. Hence, $\mathfrak{A}_2$ combines in this case the properties of $\mathfrak{A}_1$ for correspondence relations (Section 3.2) and of $\mathfrak{A}_2$ for argument mappings (Section 3.3): all generalizations are relevant, computed answer gives an mcsg of the input terms, and the algorithm works with term sets of the cardinality at most 1.

4 Discussion and conclusion

The diagram below illustrates the connections between different anti-unification problems based on argument relations:



The arrows indicate the direction from more general problems to more specific ones. For the unrestricted cases (left column) we compute mcsrg's. For correspondence relations and mappings (right column), mcsg's are computed. (In fact, for them, the notions of mcsrg and mcsg coincide). The algorithms for relations (upper row) are more involved than those for mappings (lower row): Those for relations deal with AUTs containing arbitrary sets of terms, while for mappings, those sets have cardinality at most one, thus simplifying the conditions in the rules. Moreover, the two cases in the lower row generalize the existing anti-unification problems:

- the unrestricted mappings case generalizes the problem from [1] by extending similarity to proximity and relaxing the smaller-side-totality restriction for mappings;
- the correspondence mappings case generalizes the problem from [7] by allowing permutations between arguments of proximal function symbols.

All our algorithms can be easily turned into anti-unification algorithms for crisp tolerance relations by taking lambda-cuts and ignoring the computation of the approximation degrees. Besides, they are modular and can be used to compute only linear generalizations by just skipping the merging rule.

Acknowledgments. This work has been supported by the Austrian Science Fund (FWF) under project 28789-N32.

References

[1] Hassan Aït-Kaci and Gabriella Pasi. Fuzzy lattice operations on first-order terms over signatures with similar constructors: A constraint-based approach. *Fuzzy Sets Syst.*, 391:1–46, 2020.

[2] Nino Amiridze and Temur Kutsia. Anti-unification and natural language processing. EasyChair Preprint no. 203, EasyChair, 2018.

[3] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[4] Johannes Bader, Andrew Scott, Michael Pradel, and Satish Chandra. Getafix: learning to fix bugs automatically. *Proc. ACM Program. Lang.*, 3(OOPSLA):159:1–159:27, 2019.

[5] Boris Galitsky. *Developing Enterprise Chatbots - Learning Linguistic Structures*. Springer, 2019.

[6] Pascual Julián-Iranzo and Clemente Rubio-Manzano. Proximity-based unification theory. *Fuzzy Sets and Systems*, 262:21–43, 2015.

[7] Temur Kutsia and Cleo Pau. Matching and generalization modulo proximity and tolerance relations. In Aybüke Özgün, Alex Simpson, and Yulia Zinova, editors, *13th International Tbilisi Symposium on Logic, Language, and Computation*, Lecture Notes in Computer Science. Springer. To appear.

[8] Sonu Mehta, Ranjita Bhagwan, Rahul Kumar, Chetan Bansal, Chandra Shekhar Maddila, B. Ashok, Sumit Asthana, Christian Bird, and Aditya Kumar. Rex: Preventing bugs and misconfiguration in large services using correlated change analysis. In Ranjita Bhagwan and George Porter, editors, *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*, pages 435–448. USENIX Association, 2020.

[9] Gordon D. Plotkin. A note on inductive generalization. *Machine Intel.*, 5(1):153–163, 1970.

[10] John C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. *Machine Intel.*, 5(1):135–151, 1970.