Proximity-Based Unification and Matching for Fully Fuzzy Signatures

Cleo Pau
Research Institute for Symbolic Computation
Johannes Kepler University Linz
ipau@risc.jku.at

Temur Kutsia
Research Institute for Symbolic Computation
Johannes Kepler University Linz
kutsia@risc.jku.at

***Abstract*—We consider the problem of solving approximate equations between logic terms. The approximation is expressed by proximity relations. They are reflexive and symmetric (but not necessarily transitive) fuzzy binary relations. The equations are solved by variable substitutions that bring the sides of equations "close" to each other with respect to a predefined degree. We consider unification and matching equations in which mismatches in function symbol names, arity, and in the argument order are tolerated (i.e., the approximate equations are formulated over so called fully fuzzy signatures). This work generalizes on the one hand, class-based proximity unification to fully fuzzy signatures, and on the other hand, unification with similarity relations over a fully fuzzy signature by extending similarity to proximity.**

***Index Terms*—Unification, matching, proximity relations, approximate inference, fully fuzzy signatures.**

I. Introduction

Unification and matching are central computational mechanisms in automated reasoning, rewriting, and declarative programming. In the first-order syntactic case, these techniques fail when there is no match between two corresponding function symbols of the terms to be unified. While in many situations this is the desired outcome, there are cases when some tolerance regarding the mismatches would offer a better result. The type of the accepted differences can vary, and some mismatches were already explored in the fuzzy context, concerning reasoning with imprecise, vague information.

Mismatch between symbol names under similarity. Similarity is a fuzzy equivalence relation. Similarity-based unification is a relatively well-studied technique, developed for approximate reasoning and fuzzy logic programming. Some early works in this area include [1], [2]. Sessa's weak unification algorithm [3] extends first-order unification by allowing similarity between function symbols of the same arity. Later, weak unification has been extended and generalized in various ways: to multiple similarity relations [4], similarities in fully fuzzy signatures [5], extensions of similarity to proximity [6], [7].

Similarity-based unification to fully fuzzy signatures. In fully fuzzy signatures one allows mismatches not only in symbol names, but also in their arities. Function symbols of different arities are allowed to be similar. Unification in such languages was studied in [5], where the similarity relation between arguments of different-arity function symbols is assumed to be given by injective mappings.

Supported by Austrian Science Fund (FWF) under project 28789-N32.

Extending similarity to proximity. In [6] and later in [8], the authors relaxed similarity to proximity: reflexive, symmetric, but not necessarily transitive fuzzy binary relations. The proximity relation has been considered between function symbols of the same arity and a so-called block-based approach to unification has been proposed. Blocks are cliques in undirected graphs that are induced by proximity relations. Two symbols are considered proximal if they belong to the same clique in some fixed clique partition of such graphs. The first, restricted attempt of extending block-based proximity unification to fully fuzzy signatures was reported in [9].

Yet another approach to proximity-based unification was introduced in [7]. It is based on proximity classes, which correspond to node neighborhoods in graphs generated by proximity relations. Two symbols of the same arity are proximal if they belong to the same class. The algorithm is presented as a constraint solving procedure, suitable for constraint-based inferences such as, e.g., constraint logic programming.

Our contribution: generalizing proximity-based unification to fully fuzzy signatures. In this paper, we make a step further and consider proximity relations between function symbols of different arities. It leads to the development of proximity-based unification in fully fuzzy signatures which generalizes, on the one hand, the proximity-based unification from [7] and, on the other hand, the similarity-based unification in fully fuzzy signatures from [5]. We develop two algorithms, for unification and for matching, and prove their termination, soundness, and completeness. Our approach is class-based. Proximity between arguments of different-arity function symbols is given by relations, which are not required to be functional, although for unification we consider correspondence relations. It is a very flexible approach, which opens a way to extending proximity-based unification towards special equational theories.

II. Preliminaries

A. Proximity relations

We define the basic notions about proximity relations according to [6]. For 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 $\mathcal{R}_\lambda := \{(s_1, s_2) \mid \mathcal{R}(s_1, s_2) \geq \lambda\}$. We take the minimum

978-1-6654-4407-1/21/$31.00 © 2021 IEEE

as the T-norm $\wedge$. A *proximity relation* on a set S is a reflexive and symmetric fuzzy relation $\mathcal{R}$ on S.

B. Terms and substitutions

We consider first-order *terms* defined over a set of variables $\mathcal{V}$ and a set of function symbols $\mathcal{F}$: $t := x \mid f(t_1, \ldots, t_n)$, where $x \in \mathcal{V}$ is a variable and $f \in \mathcal{F}$ is an n-ary function symbol with $n \geq 0$. We denote arbitrary function symbols by f, g, h, p, q, constants (0-ary function symbols) by a, b, c, variables by x, y, z, u, v, and terms by s, t, r. The *head* of a term is defined as $head(x) := x$ and $head(f(t_1, \ldots, t_n)) := f$. The set of all variables appearing in t is denoted by $var(t)$. If $var(t) = \emptyset$, we call t a *ground* term. The notions of *position* and *term depth* are defined in the standard way, see, e.g. [10].

A *substitution* is a mapping from variables to terms, 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 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.

The *application* of a substitution σ to a term t, denoted by $t\sigma$, is defined as $x\sigma := \sigma(x)$ 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.

The notion of *more generality* for substitutions is defined with the help of syntactic equality: σ is more general than ϑ, written $\sigma \preceq \vartheta$, if there exists φ such that $\sigma\varphi = \vartheta$.[1] The strict part $of \preceq$ is denoted by $\prec$.

C. Argument relations

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.

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.

D. Proximity relations over terms

Each proximity relation $\mathcal{R}$ considered 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 $x \neq y$, $x, y \in \mathcal{V}$. It is assumed 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 then extend such an $\mathcal{R}$ to terms: (i) $\mathcal{R}(s, t) := 0$ if $\mathcal{R}(head(s), head(t)) = 0$; (ii) $\mathcal{R}(s, t) := 1$ if $s = t$, $s, t \in \mathcal{V}$; (iii) $\mathcal{R}(s, t) := \mathcal{R}(f, g) \wedge \mathcal{R}(s_{i_1}, t_{j_1}) \wedge \cdots \wedge \mathcal{R}(s_{i_k}, t_{j_k})$, if $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, $f \sim^{\rho}_{\mathcal{R},\lambda} g$, and $\rho = \{(i_1, j_1), \ldots, (i_k, j_k)\}$.

E. Unification problems, unifiers

We write $(\mathcal{R}, \lambda)$-*equations* between terms as $t \simeq^?_{\mathcal{R},\lambda} s$, with the question mark indicating that they are supposed to be solved, i.e., the terms t and s to be $(\mathcal{R}, \lambda)$-*unified*. A *solution* (*unifier*) of such an equation is a substitution σ such that $t\sigma \simeq_{\mathcal{R},\lambda} s\sigma$. We say that $\mathcal{R}(t\sigma, s\sigma) \geq \lambda$ is the *approximation degree* of $(\mathcal{R}, \lambda)$-solving $t \simeq^?_{\mathcal{R},\lambda} s$ by σ.

An $(\mathcal{R}, \lambda)$-*unification problem* (or, briefly, a *unification problem*) is a finite set of $(\mathcal{R}, \lambda)$-equations. A *solution* (*unifier*) of a unification problem P is a substitution that solves all the equations in P. We say that σ is a *most general unifier* (*mgu*) of P if no other unifier of P is strictly more general than σ. The approximation degree of the unification of P by σ, denoted by $deg(P\sigma)$, is $\wedge_{eq \in P} deg(eq\sigma)$, where $deg(eq\sigma)$ is the approximation degree of solving $eq \in P$ by σ.

F. Matching problems, matchers

An equation with a ground side is called a *matching equation*. We write $t \precsim^?_{\mathcal{R},\lambda} s$ for an $(\mathcal{R}, \lambda)$-matching equation between t and s, where s is the ground side. A *solution* (*matcher*) of such an equation is a substitution σ that matches t to s with respect to $\mathcal{R}$ and λ, i.e., $t\sigma \simeq_{\mathcal{R},\lambda} s$. The number $\mathcal{R}(t\sigma, s) \geq \lambda$ is the corresponding *approximation degree*.

A peculiarity of matching for proximity relations with arity mismatch is that matchers do not have to be ground substitutions. For instance, if f is binary, and g and h are unary symbols with $f \sim^{\{(2,1)\}}_{\mathcal{R},0.7} g$ and $f \sim^{\{(2,1)\}}_{\mathcal{R},0.6} h$, then $\sigma = \{x \mapsto f(y, a)\}$ is a matcher of $f(x, x) \precsim_{\mathcal{R},0.5} f(g(a), h(a))$ with the degree 0.6. In fact, σ is more general than any other solution of this problem. Analogously to unification, we will use the notion of most general solution for matching problems and write $(\mathcal{R}, \lambda)$-*mgm* for most general $(\mathcal{R}, \lambda)$-matchers.

III. Unification

In this section we assume that in proximity relations, all argument relations are of a special form: they are both left- and right-total. A 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 $(i, j) \in \rho$. Relations that are both left-and right-total are called *correspondence relations*. Hence, all argument relations we consider for unification are correspondence relations.

Lemma 1. *If all argument relations in $\mathcal{R}$ are correspondence relations, then for any λ-cut: (a) $t \simeq_{\mathcal{R},\lambda} s$ implies $var(t) = var(s)$; (b) no term is $(\mathcal{R}, \lambda)$-close to its proper subterm.*

Proof. We prove (a) by structural induction for terms. If both t and s are variables, then $t \simeq_{\mathcal{R},\lambda} s$ implies $t = s$. If they are nonvariable terms $t = f(t_1 \ldots, t_n)$ and $s = g(s_1, \ldots, s_n)$ with $f \sim^{\rho}_{\mathcal{R},\lambda} g$, then the correspondence property of ρ implies the following: for each t_i there is s_j such that $t_i \simeq_{\mathcal{R},\lambda} s_j$ and, hence, by the induction hypothesis $var(t_i) = var(s_j)$, and vice versa: for each s_j there is t_i such that $s_j \simeq_{\mathcal{R},\lambda} t_i$

[1] Note that we did not use proximity in the definition of more generality, in order to guarantee that $\preceq$ is a quasi-order, preserving good properties of unifiers. See Remark 1 in [7].

and, hence, by the induction hypothesis $var(s_j) = var(t_i)$. Therefore, $var(t) = \cup_{i=1}^{n} var(t_i) = \cup_{j=1}^{m} var(s_j) = var(s)$.

To prove (b), by the definition of correspondence relations, a non-constant term cannot be $(\mathcal{R}, \lambda)$-close to a constant. According to the definition of proximity, no nonvariable term is $(\mathcal{R}, \lambda)$-close to a variable. By structural induction over terms we get that no term is $(\mathcal{R}, \lambda)$-close to its proper subterm. $\square$

A set of $(\mathcal{R}, \lambda)$-equations $\{x \simeq^?_{\mathcal{R},\lambda} s\} \uplus P$ contains an *occurrence cycle* for x if $s \notin \mathcal{V}$ and there exist term-pairs (x_0, s_0), $(x_1, s_1), \ldots, (x_n, s_n)$ such that $x_0 = x$, $s_0 = s$, and for each $0 \le i \le n$ the set P contains an equation $x_i \simeq^?_{\mathcal{R},\lambda} s_i$ or $s_i \simeq^?_{\mathcal{R},\lambda} x_i$ with $x_{i+1} \in var(s_i)$, where $x_{n+1} = x_0$.

Lemma 2. *Let all argument relations in $\mathcal{R}$ be correspondence relations. If a set of $(\mathcal{R}, \lambda)$-equations P contains an occurrence cycle for some variable, then P has no solution.*

Proof. By Lemma 1, no term can be $(\mathcal{R}, \lambda)$-close to its proper subterm. Therefore, equations containing an occurrence cycle cannot have an solution. $\square$

Now we formulate a unification algorithm in a rule-based way. The rules work on triples $P; \sigma; \alpha$, called unification configurations, where P is a unification problem, σ is the substitution computed so far, and α is the approximation degree, also computed so far. The symbol $\perp$ is a special configuration, indicating failure. The rules transform configurations into configurations ($\mathcal{R}$ and λ are given, $\uplus$ is disjoint union):

Tri-U: **Trivial**

$\{x \simeq^?_{\mathcal{R},\lambda} x\} \uplus P; \sigma; \alpha \Longrightarrow P; \sigma; \alpha.$

Dec-U: **Decomposition**

$\{f(t_1, \ldots, t_n) \simeq^?_{\mathcal{R},\lambda} g(s_1, \ldots, s_m)\} \uplus P; \sigma; \alpha \Longrightarrow$
$\quad P \cup \{t_i \simeq^?_{\mathcal{R},\lambda} s_j \mid (i,j) \in \rho\}; \sigma; \alpha \wedge \beta,$

where $n, m \ge 0$, $f \sim^{\rho}_{\mathcal{R},\beta} g$, and $\beta \ge \lambda$.

Cla-U: **Clash**

$\{f(t_1, \ldots, t_n) \simeq^?_{\mathcal{R},\lambda} g(s_1, \ldots, s_m)\} \uplus P; \sigma; \alpha \Longrightarrow \perp,$

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

Ori-U: **Orient**

$\{t \simeq^?_{\mathcal{R},\lambda} x\} \uplus P; \sigma; \alpha \Longrightarrow P \cup \{x \simeq^?_{\mathcal{R},\lambda} t\}; \sigma; \alpha,$

if t is not a variable.

Occ-U: **Occurrence check**

$\{x \simeq^?_{\mathcal{R},\lambda} g(s_1, \ldots, s_n)\} \uplus P; \sigma; \alpha \Longrightarrow \perp,$

if $\{x \simeq^?_{\mathcal{R},\lambda} g(s_1, \ldots, s_n)\} \uplus P$ has an occurrence cycle for x.

Var-E-U: **Variable elimination**

$\{x \simeq^?_{\mathcal{R},\lambda} g(s_1, \ldots, s_n)\} \uplus P; \sigma; \alpha \Longrightarrow$
$\quad P\vartheta \cup \{v_i \simeq^?_{\mathcal{R},\lambda} s_j \mid (i,j) \in \rho\}; \sigma\vartheta; \alpha \wedge \beta,$

where $\{x \simeq^?_{\mathcal{R},\lambda} g(s_1, \ldots, s_n)\} \uplus P$ does not contain an occurrence cycle for x, $\vartheta = \{x \mapsto f(v_1, \ldots, v_m)\}$ with fresh variables $v_1, \ldots, v_m$, $f \sim^{\rho}_{\mathcal{R},\beta} g$ with $\beta \ge \lambda$, and $n, m \ge 0$.

Given a unification problem P, we create the initial system $P; Id; 1$ and start applying the unification rules in all possible ways, generating a complete tree of derivations in the standard way. The Var-E-U rule causes branching, since there can be multiple f's satisfying the condition there. No rule applies to $\perp$ or to a configuration of the form $\{x_1 \simeq^?_{\mathcal{R},\lambda} y_1, \ldots, x_n \simeq^?_{\mathcal{R},\lambda} y_n\}; \sigma; \alpha$, $n \ge 0$, called *variables-only* configuration. In the latter case we say that α is the computed approximation degree, $\sigma|_{var(P)}$ is the computed substitution, and $\{x_1 \simeq^?_{\mathcal{R},\lambda} y_1, \ldots, x_n \simeq^?_{\mathcal{R},\lambda} y_n\}$ is the computed constraint. We denote the obtained unification algorithm by $\mathfrak{U}$.

In the examples below it is assumed that $\mathcal{R}(sym_1, sym_2) = 0$ for any pair of distinct symbols sym_1 and sym_2 except those for which the proximity is explicitly given.

Example 1. Assume p is a unary function symbol, q, g, and h are binary, f is ternary, and a, b, c are constants such that $p \sim^{\{(1,1),(1,2)\}}_{\mathcal{R},0.7} q$, $f \sim^{\{(1,1),(2,2),(3,1)\}}_{\mathcal{R},0.6} g$, $f \sim^{\{(1,2),(2,1),(3,2)\}}_{\mathcal{R},0.5} h$, and $b \sim^{\emptyset}_{\mathcal{R},0.4} c$. Consider the unification problem $P = \{p(x) \simeq^?_{\mathcal{R},0.4} q(g(u, y), h(z, u))\}$. Then $\mathfrak{U}$ stops with the configuration $S; \sigma; \alpha$ where $S = \{v_1 \simeq^?_{\mathcal{R},0.4} u, v_2 \simeq^?_{\mathcal{R},0.4} y, v_2 \simeq^?_{\mathcal{R},0.4} z, v_3 \simeq^?_{\mathcal{R},0.4} u\}$, $\sigma = \{x \mapsto f(v_1, v_2, v_3)\}$, and $\alpha = 0.5$. For illustration, we take three unifiers of P: ϑ_1, ϑ_2, and ϑ_3 together with their approximation degrees β_1, β_2, and β_3, and show how they can be obtained from $S; \sigma$:

1) $\vartheta_1 = \{x \mapsto f(u, z, u), y \mapsto z\}$ and $\beta_1 = 0.5$.

   The instance of $S; \sigma$ under $\varphi = \{y \mapsto z, v_1 \mapsto u, v_2 \mapsto z, v_3 \mapsto u\}$: $S\varphi = \{u \simeq^?_{\mathcal{R},0.4} u, z \simeq^?_{\mathcal{R},0.4} z\}$ and $\sigma\varphi = \{x \mapsto f(u, z, u), y \mapsto z, v_1 \mapsto u, v_2 \mapsto z, v_3 \mapsto u\}$.

   $S\varphi$ is solved and $(\sigma\varphi)|_{var(P)} = \vartheta_1$. Besides, $\alpha \ge \beta_1$.

2) $\vartheta_2 = \{x \mapsto f(u, b, u), y \mapsto b, z \mapsto b\}$ and $\beta_2 = 0.5$.

   The instance of $S; \sigma$ under $\varphi = \{y \mapsto b, z \mapsto b, v_1 \mapsto u, v_2 \mapsto b, v_3 \mapsto u\}$: $S\varphi = \{u \simeq^?_{\mathcal{R},0.4} u, b \simeq^?_{\mathcal{R},0.4} b\}$ and $\sigma\varphi = \{x \mapsto f(u, b, u), y \mapsto b, z \mapsto b, v_1 \mapsto u, v_2 \mapsto b, v_3 \mapsto u\}$.

   $S\varphi$ is solved and $(\sigma\varphi)|_{var(P)} = \vartheta_2$. Besides, $\alpha \ge \beta_2$.

3) $\vartheta_3 = \{x \mapsto f(u, c, u), y \mapsto b, z \mapsto c\}$ and $\beta_3 = 0.4$.

   The instance of $S; \sigma$ under $\varphi = \{v_1 \mapsto u, v_2 \mapsto c, y \mapsto b, z \mapsto c, v_3 \mapsto u\}$: $S\varphi = \{u \simeq^?_{\mathcal{R},0.4} u, c \simeq^?_{\mathcal{R},0.4} b, c \simeq^?_{\mathcal{R},0.4} c\}$ and $\sigma\varphi = \{x \mapsto f(u, c, u), v_1 \mapsto u, v_2 \mapsto c, y \mapsto b, z \mapsto c, v_3 \mapsto u\}$.

   $S\varphi$ is solved, and $(\sigma\varphi)|_{var(P)} = \vartheta_3$. Besides, $\alpha \ge \beta_3$.

This example explains why $\mathfrak{U}$ stops at variables-only configuration. If it went further from $S; \sigma; \alpha$ as usual and eliminated y, v_1, v_2, v_3 by $\{y \mapsto z, v_1 \mapsto u, v_2 \mapsto z, v_3 \mapsto u\}$, we would end up with the configuration $\emptyset; \{x \mapsto f(u, z, u), y \mapsto z, v_1 \mapsto u, v_2 \mapsto z, v_3 \mapsto u\}$, computing the unifier ϑ_1 as above, but it would not be more general than the unifier ϑ_3. (Recall: more generality is defined by equality, not by proximity.)

Theorem 1. *The decision problem of $(\mathcal{R}, \lambda)$-unifiability with arity mismatch is NP-hard.*

Proof. By reduction from positive 1-in-3-SAT. Consider the argument correspondence relations[2] $\rho_1 = \{(1,1),(2,2),(3,3)\}$, $\rho_2 = \{(1,3),(2,1),(3,2)\}$, $\rho_3 = \{(1,2),(2,3),(3,1)\}$, and assume $h_i \simeq^{\rho_1}_{\mathcal{R},\lambda} f$ and $h_i \simeq^{\rho_i}_{\mathcal{R},\lambda} g$ for each $1 \le i \le 3$. Then each positive 3-SAT clause $x_1 \vee x_2 \vee x_3$ can be encoded as two proximity equations $y \simeq^?_{\mathcal{R},\lambda} f(x_1,x_2,x_3)$ and $y \simeq^?_{\mathcal{R},\lambda} g(1,0,0)$, where 1 and 0 are constants. Their unifiers force exactly one x to be mapped to 1, and the other two to 0 ($\{y \mapsto h_1(1,0,0), x_1 \mapsto 1, x_2 \mapsto 0, x_3 \mapsto 0\}$, $\{y \mapsto h_2(0,1,0), x_1 \mapsto 0, x_2 \mapsto 1, x_3 \mapsto 0\}$, and $\{y \mapsto h_3(0,0,1), x_1 \mapsto 0, x_2 \mapsto 0, x_3 \mapsto 1\}$). The reduction is polynomial and preserves solvability in both directions. ☐

Below we state the properties of the algorithm $\mathfrak{U}$.

Theorem 2. *$\mathfrak{U}$ terminates for any input.*

Proof. According to [11], for a syntactic unification problem P, the maximal depth of terms in an mgu of P does not exceed the size of P (i.e., the number of symbols in P, denoted by $size(P)$). Due to the definition of proximity between terms, no proximal mgu can be deeper than a syntactic mgu. Therefore, the same bound applies to $(\mathcal{R},\lambda)$-unification problems.

Given a variable v and a substitution σ, let $md_\sigma(v)$ be the natural number that denotes the *maximal depth* at which this variable occurs in the range of σ. If v does not appear in the range of σ, then $md_\sigma(v) = 0$.

To an $(\mathcal{R},\lambda)$-unification configuration $P_i;\sigma_i;\alpha_i$, we associate the multiset $M_i := \{\!\{ md_{\sigma_i}(v) \mid v \in var(P_i) \}\!\}$. Then, for the initial configuration $P_0;\sigma_0;\alpha_0$, where $\sigma_0 = Id$, we have $M_0 = \{\!\{0,\dots,0\}\!\}$ with $|M_0| = |var(P_0)|$. These multisets are ordered by the multiset extension $<_{mul}$ of the standard natural number ordering [12].

When Var-E-U transforms $P_i;\sigma_i;\alpha_i$ into $P_{i+1};\sigma_{i+1};\alpha_{i+1}$ with $\sigma_{i+1} = \sigma_i\{x \mapsto f(v_1,\dots,v_m)\}$, we get $var(P_{i+1}) = \big(var(P_i) \setminus \{x\}\big) \cup \{v_1,\dots,v_m\}$ and $md_{\sigma_{i+1}}(v_1) = \cdots = md_{\sigma_{i+1}}(v_m) = 1 + md_{\sigma_i}(x)$. Hence, we have $M_{i+1} = \big(M_i \setminus \{\!\{md_{\sigma_i}(x)\}\!\}\big) \cup \{\!\{md_{\sigma_{i+1}}(v_1),\dots,md_{\sigma_{i+1}}(v_m)\}\!\}$. Therefore, $M_i <_{mul} M_{i+1}$ after the application of Var-E-U.

On the other hand, occurrence cycle check in Var-E-U prevents an uncontrolled growth of the multisets. Thus, with each derivation we get the chain $M_0 = \cdots = M_{i_1} <_{mul} M_{i_1+1} = \cdots = M_{i_2} <_{mul} M_{i_2+1} = \cdots <_{mul} \{\!\{size(P)+1\}\!\}$, where $i_1, i_2 \dots$ are the steps when Var-E-U is used. Since the chain is bounded, Var-E-U cannot be applied infinitely often.

From the other rules, Tri-U and Dec-U do not affect the multisets and strictly decrease $size(P)$. Var-E-U may increase the size but, as we said above, it may be applied only finitely many times. Therefore, Tri-U and Dec-U cannot be applied infinitely often. Ori-U does not change the multisets and the size, but strictly decreases the number of equations of the form $t \simeq^?_{\mathcal{R},\lambda} x$, where t is not a variable. The number of such equations may grow after the application of Dec-U or Var-E-U, but it can happen only finitely many times. Therefore, Ori-U cannot be applied infinitely often either. The failure rules stop immediately. Hence, $\mathfrak{U}$ is terminating. ☐

[2] Actually, these relations are total bijective functions.

Lemma 3. *Given $\mathcal{R}$, λ, and an $(\mathcal{R},\lambda)$-unification problem P:*
1) *If $P;\sigma;\alpha \Longrightarrow \bot$ in $\mathfrak{U}$, then P has no solution.*
2) *If $P;\sigma;\alpha \Longrightarrow P';\sigma\vartheta;\alpha\wedge\beta$ in $\mathfrak{U}$ and φ is a solution of P' with the approximation degree γ, then $\vartheta\varphi$ is a solution of P with the approximation degree $\beta\wedge\gamma$.*

Proof. In 1), if the step is made by the Cla-U rule, then it is obvious that P has no solution. If the Occ-U rule is used, then the theorem follows from Lemma 2.

To prove 2), we shall consider each non-failing rule. The nontrivial cases are Dec-U and Var-E-U.

In Dec-U, the transformation is $\{f(t_1,\dots,t_n) \simeq^?_{\mathcal{R},\lambda} g(s_1,\dots,s_m)\} \uplus P;\sigma;\alpha \Longrightarrow P \cup \{t_i \simeq^?_{\mathcal{R},\lambda} s_j \mid (i,j) \in \rho\}$; $\sigma;\alpha\wedge\beta$, where $n,m \ge 0$, $f \sim^\rho_{\mathcal{R},\beta} g$, and $\beta \ge \lambda$. If φ is a solution of $P \cup \{t_i \simeq^?_{\mathcal{R},\lambda} s_j \mid (i,j) \in \rho\}$, we have $deg(P\varphi) \wedge \bigwedge_{(i,j)\in\rho} \mathcal{R}(t_i\varphi, s_j\varphi) = \gamma \ge \lambda$. But then $deg(P\varphi) \wedge \mathcal{R}(f(t_1,\dots,t_n)\varphi, g(s_1,\dots,s_m)\varphi) = \beta\wedge\gamma$. Hence, φ is a solution of $\{f(t_1,\dots,t_n) \simeq^?_{\mathcal{R},\lambda} g(s_1,\dots,s_m)\} \uplus P$ with the approximation degree $\beta\wedge\gamma \ge \lambda$.

In Var-E-U, the step is $\{x \simeq^?_{\mathcal{R},\lambda} g(s_1,\dots,s_n)\} \uplus P;\sigma;\alpha \Longrightarrow P\vartheta \cup \{v_i \simeq^?_{\mathcal{R},\lambda} s_j \mid (i,j) \in \rho\}$; $\sigma\vartheta;\alpha\wedge\beta$, where $\vartheta = \{x \mapsto f(v_1,\dots,v_m)\}$ with $v_1,\dots,v_m$ fresh variables, and $f \sim^\rho_{\mathcal{R},\beta} g$ with $\beta \ge \lambda$. If φ solves $P\vartheta \cup \{v_i \simeq^?_{\mathcal{R},\lambda} s_j \mid (i,j) \in \rho\}$, we have $deg(P\vartheta\varphi) \wedge \bigwedge_{(i,j)\in\rho} \mathcal{R}(v_i\varphi, s_j\varphi) = \gamma \ge \lambda$. But then $deg(P\vartheta\varphi) \wedge \mathcal{R}(x\vartheta\varphi, g(s_1,\dots,s_n)\vartheta\varphi) = deg(P\vartheta\varphi) \wedge \mathcal{R}(f(v_1,\dots,v_m)\varphi, g(s_1,\dots,s_n)\varphi) = \beta\wedge\gamma$, and thus, φ is a solution of $\{x \simeq^?_{\mathcal{R},\lambda} g(s_1,\dots,s_n)\} \uplus P$ with the approximation degree $\beta\wedge\gamma \ge \lambda$. ☐

Theorem 3 (Soundness of $\mathfrak{U}$)**.** *Let $P;Id;1 \Longrightarrow^* S;\sigma;\alpha$ be a derivation in $\mathfrak{U}$ where $S;\sigma;\alpha$ is a variables-only configuration. Let φ be a unifier of S with the approximation degree β. Then $\sigma\varphi$ is a unifier of P with the approximation degree $\alpha\wedge\beta$.*

Proof. Induction on the derivation length, using Lemma 3. ☐

Theorem 4 (Completeness of $\mathfrak{U}$)**.** *Let P be a $(\mathcal{R},\lambda)$-unification problem and ϑ be its unifier with the approximation degree β. Then there exists a derivation $P;Id;1 \Longrightarrow^* S;\sigma;\alpha$ in $\mathfrak{U}$, where $S;\sigma;\alpha$ is a variables-only configuration with $\alpha \ge \beta$ and there is a unifier φ of S such that $(\sigma\varphi)|_{var(P)} = \vartheta|_{var(P)}$.*

Proof. The existence of a derivation in $\mathfrak{U}$ that ends in a variables-only configuration follows from Lemma 3 and from the assumption that P is solvable.

We now construct recursively the desired derivation and the substitution φ using ϑ. For the initial configuration $P;Id;1$ we take $\sigma = Id$, $\alpha = 1$, $\varphi = \vartheta$. Then $\alpha \ge \beta$ and $(\sigma\varphi)|_{var(P)} = \vartheta|_{var(P)}$ hold. Next, we take $C_0 = \{t \simeq^?_{\mathcal{R},\lambda} s\} \uplus P_0;\sigma_0;\alpha_0$ and assume that φ_0 is a unifier of $\{t \simeq^?_{\mathcal{R},\lambda} s\} \uplus P_0$ such that $(\sigma_0\varphi_0)|_{var(P)} = \vartheta|_{var(P)}$ and $\alpha_0 \ge \beta$. We prove that there exists a configuration $C_1 = P_1;\sigma_1;\alpha_1$ and a unifier φ_1 of P_1 such that $C_0 \Longrightarrow C_1$, $(\sigma_1\varphi_1)|_{var(P)} = \vartheta|_{var(P)}$ and $\alpha_1 \ge \beta$.

From the four non-failing rules that can perform the step $C_0 \Longrightarrow C_1$ only Var-E-U with $t = x$ and $s = g(s_1,\dots,s_n)$ is non-trivial. Since φ_0 solves $\{x \simeq^?_{\mathcal{R},\lambda} g(s_1,\dots,s_n)\} \uplus P_0$, we have $x\varphi_0 = f(r_1,\dots,r_m)$ for an f with $f \sim^\rho_{\mathcal{R},\beta_1} g$ and $r_i \simeq_{\mathcal{R},\lambda} s_j$ for all $(i,j) \in \rho$. Note that $\beta_1 \ge \beta$. Then

$P_1 = \{v_i \simeq^?_{\mathcal{R},\lambda} s_j \mid (i,j) \in \rho\} \cup P_0\psi$, $\sigma_1 = \sigma_0\psi$ where $\psi = \{x \mapsto f(v_1,\dots,v_m)\}$, and $\alpha_1 = \alpha_0 \wedge \beta_1$. Take $\varphi_1 = \nu\varphi_0$, where $\nu = \{v_i \mapsto r_i \mid 1 \le i \le m\}$. Then φ_1 solves P_1, since (i) $y\varphi_1 = y\varphi_0$ for all $y \in var(P_0)\setminus\{x\}$; (ii) $v_k\varphi_1 = r_k$ for all $1 \le k \le m$; and (iii) $v_i\varphi_1 \simeq_{\mathcal{R},\lambda} s_j$ for those v_i's for which $(i,j) \in \rho$. Moreover, $(\sigma_1\varphi_1)|_{var(P)} = (\sigma_0\psi\nu\varphi_0)|_{var(P)} = (\sigma_0\{x \mapsto x\varphi_0\}\nu\varphi_0)|_{var(P)} = (\sigma_0\{x \mapsto x\varphi_0\}\varphi_0\nu)|_{var(P)} = (\sigma_0\varphi_0\nu)|_{var(P)} = (\sigma_0\varphi_0)|_{var(P)} = \vartheta|_{var(P)}$. Finally, $\alpha_1 \ge \beta$, because $\alpha_0 \ge \beta$, $\beta_1 \ge \beta$, and $\alpha_1 = \alpha_0 \wedge \beta_1$. ☐

IV. Matching

In this section, there are no restrictions on argument relations. To solve a matching problem $t \precsim^?_{\mathcal{R},\lambda} s$, we create the triple $\{t \precsim^?_{\mathcal{R},\lambda} s\}; \emptyset; 1$ and apply the rules below. They work on triples $M; S; \alpha$, where M is a set of matching equations to be solved, S is the set of solved equations of the form $x \approx s$, and α is the approximation degree computed so far.

In the rule Var-E-M, we need the $(\mathcal{R},\lambda)$-proximity class $\mathbf{pc}_{\mathcal{R},\lambda}(s)$ of a term s, defined as follows:

$$\mathbf{pc}_{\mathcal{R},\lambda}(x) = \{x\}.$$

$$\mathbf{pc}_{\mathcal{R},\lambda}(g(s_1,\dots,s_m)) := \big\{ f(t_1,\dots,t_n) \,\big|\, g \sim^{\rho}_{\mathcal{R},\beta} f, \; \beta \ge \lambda,\ f \text{ is } n\text{-ary, and for each } 1 \le j \le n, \; t_j \in \mathbf{pc}_{\mathcal{R},\lambda}(s_i),\ \text{if } (i,j) \in \rho,\ \text{or} \; t_j = v,\ \text{if for no } i,\ 1 \le i \le m,\ (i,j) \in \rho, \; \text{where } v \text{ is a fresh variable}\big\}.$$

In the rule Mer-M, we need the operation $\Cap$ of merging two terms, defined as (i) $x \Cap t = t \Cap x = t$ for any variable x and term t; (ii) $f(t_1,\dots,t_n) \Cap f(s_1,\dots,s_n) := f(t_1 \Cap s_1,\dots,t_n \Cap s_n)$, $n \ge 0$; (iii) $t \Cap s$ is undefined in any other case.

The matching rules are the following:

Dec-M: **Decomposition**

$$\{f(t_1,\dots,t_n) \precsim^?_{\mathcal{R},\lambda} g(s_1,\dots,s_m)\} \uplus M; S; \alpha \Longrightarrow M \cup \{t_i \precsim^?_{\mathcal{R},\lambda} s_j \mid (i,j) \in \rho\}; S; \alpha \wedge \beta,$$

if $n, m \ge 0$ and $f \sim^{\rho}_{\mathcal{R},\beta} g$ with $\beta \ge \lambda$.

Var-E-M: **Variable elimination**

$$\{x \precsim^?_{\mathcal{R},\lambda} s\} \uplus M; S; \alpha \Longrightarrow M; S \cup \{x \approx t\}; \alpha \wedge \beta,$$

where $t \in \mathbf{pc}_{\mathcal{R},\lambda}(s)$ and $\mathcal{R}(t,s) = \beta \ge \lambda$.

Mer-M: **Merging**

$$M; \{x \approx t, x \approx s\} \uplus S; \alpha \Longrightarrow M; S \cup \{x \approx t \Cap s\}; \alpha,$$

if $t \Cap s$ is defined.

Cla-M: **Clash**

$$\{f(t_1,\dots,t_n) \precsim^?_{\mathcal{R},\lambda} g(s_1,\dots,s_m)\} \uplus M; S; \alpha \Longrightarrow \bot,$$

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

Inc-M: **Inconsistency**

$$M; \{x \approx t,\ x \approx s\} \uplus S; \alpha \Longrightarrow \bot, \qquad \text{if } t \Cap s \text{ is undefined.}$$

The matching algorithm $\mathfrak{M}$ uses these rules to transform triples as long as possible, returning either $\bot$ (indicating failure), or $\emptyset; S; \alpha$ (indicating success). In the latter case, each variable occurs in S at most once. Therefore, from S one can obtain a substitution $\sigma_S := \{x \mapsto s \mid x \approx s \in S\}$. We call it the *computed substitution*.

We call a substitution σ an $(\mathcal{R},\lambda)$-solution of an $M; S$ pair, iff σ is an $(\mathcal{R},\lambda)$-matcher of M and for all $x \approx t \in S$, we have $x\sigma = t$. We also assume that $\bot$ has no solution.

Example 2. Assume p, g and h are unary symbols, q is binary, f is ternary, and a, b, and c are constants such that $p \sim^{\{(1,1),(1,2)\}}_{\mathcal{R},0.7} q$, $f \sim^{\{(1,1)\}}_{\mathcal{R},0.6} g$, $f \sim^{\{(3,1)\}}_{\mathcal{R},0.5} h$, and $b \sim^{\emptyset}_{\mathcal{R},0.4} c$. We consider the matching problem $p(x) \precsim_{\mathcal{R},0.4} q(g(a), h(c))$ and show only the successful derivations. They start with

$$\{p(x) \precsim^?_{\mathcal{R},\lambda} q(g(a), h(c))\}; \emptyset; 1 \Longrightarrow_{\text{Dec-M}}$$
$$\{x \precsim^?_{\mathcal{R},\lambda} g(a),\ x \precsim^?_{\mathcal{R},\lambda} h(c)\}; \emptyset; 0.7 \Longrightarrow_{\text{Var-E-M}}$$
$$\{x \precsim^?_{\mathcal{R},\lambda} h(c)\}; \{x \approx f(a, v_1, v_2)\}; 0.6$$

and then continue by Var-E-M in two different ways:

1: $\emptyset; \{x \approx f(a, v_1, v_2), x \approx f(v_3, v_4, c)\}; 0.5 \Longrightarrow_{\text{Mer-M}}$
$\emptyset; \{x \approx f(a, v_1, c)\}; 0.5.$

2: $\emptyset; \{x \approx f(a, v_1, v_2), x \approx f(v_3, v_4, b)\}; 0.4 \Longrightarrow_{\text{Mer-M}}$
$\emptyset; \{x \approx f(a, v_1, b)\}; 0.4.$

The computed substitutions $\{x \mapsto f(a, v_1, c)\}$ and $\{x \mapsto f(a, v_1, b)\}$ are matchers of the original problem with the approximation degrees 0.5 and 0.4, respectively.

Remark 1. In Theorem 1, we proved the NP-hardness of unification. It can be also shown for *well-moded unification problems*. They are special unification problems in which the equations can be ordered as $t_0 \simeq^?_{\mathcal{R},\lambda} s_0, \dots, t_n \simeq^?_{\mathcal{R},\lambda} s_n$, with s_0 ground and $var(s_i) \subseteq \cup_{j=0}^{i-1} var(t_j)$, $1 \le i \le n$. Hence, $t_0 \simeq^?_{\mathcal{R},\lambda} s_0$ is actually a matching problem $t_0 \precsim^?_{\mathcal{R},\lambda} s_0$. If we solve these equations from left to right, the s's get ground as we move. Thus, we will encounter only matching equations. The encoding in the proof of Theorem 1 can be expressed as a well-moded unification problem, written as matching equations $y \precsim^?_{\mathcal{R},\lambda} g(1,0,0), f(x_1,x_2,x_3) \precsim^?_{\mathcal{R},\lambda} y$. Hence, the decision problem for well-moded proximity unification is NP-hard.

Lemma 4. *Let $M_1; S_1; \alpha \Longrightarrow M_2; S_2; \alpha \wedge \beta$ be a step made by $\mathfrak{M}$. If ϑ is an $(\mathcal{R},\lambda)$-solution of $M_2; S_2$ with the approximation degree γ then it is an $(\mathcal{R},\lambda)$-solution of $M_1; S_1$ with the approximation degree $\beta \wedge \gamma$.*

Proof. By the definition of a matcher, the lemma holds for Dec-M and Cla-M. The definition of $\Cap$ implies it for Mer-M and Inc-M. For Var-E-M, by the definition of $\mathbf{pc}$, we have $x\vartheta \in \mathbf{pc}_{\mathcal{R},\lambda}(s)$ iff $\mathcal{R}(x\vartheta, s) = \beta \ge \lambda$, which implies that ϑ with $x\vartheta = t$ is an $(\mathcal{R},\lambda)$-matcher of $x \precsim^?_{\mathcal{R},\lambda} s$ with the degree β. Since ϑ is a matcher of $M_2; S_2$ with the degree γ, it is a matcher of $M_1; S_1$ with the degree $\beta \wedge \gamma$. ☐

Theorem 5. *Given an $(\mathcal{R},\lambda)$-matching problem $t \precsim^?_{\mathcal{R},\lambda} s$, the matching algorithm $\mathfrak{M}$ terminates and computes a substitution σ that is an $(\mathcal{R},\lambda)$-matcher of t to s.*

Proof. We prove termination and soundness separately.

Termination. The rules Dec-M and Var-E-M strictly reduce the size of M. Mer-M does the same for S, without changing M.

$\mathsf{Cla\text{-}M}$ and $\mathsf{Inc\text{-}M}$ stop immediately. Hence, $\mathfrak{M}$ strictly reduces the lexicographic combination $\langle size(M), size(S)\rangle$ of sizes of M and S, which implies termination.

Soundness. Let $\{t \precsim^?_{\mathcal{R},\lambda} s\}; \emptyset; 1 \Longrightarrow^+ \emptyset; S; \alpha$ be the derivation in $\mathfrak{M}$ that computes σ. Then σ is a solution of $\emptyset; S$. By induction on the length of the derivation, using Lemma 4, we can prove that σ is an $(\mathcal{R}, \lambda)$-matcher of t to s. $\square$

Theorem 6. *Given an $(\mathcal{R}, \lambda)$-matching problem M and its solution ϑ, the algorithm $\mathfrak{M}$ computes a substitution σ such that $x\vartheta \Cap x\sigma = x\vartheta$ for all x in M.*

Proof. This theorem essentially says that for an x occurring in the matching problem, if $r_1 = x\vartheta$ and $r_2 = x\sigma$, then r_1 and r_2 have exactly same structure (otherwise $r_1 \Cap r_2$ would not be defined) and they may differ from each other only at those positions where r_2 contains a variable.

We need to construct a derivation that computes σ. The only steps in the derivation that take into account ϑ are $\mathsf{Var\text{-}E\text{-}M}$ steps. When we transform $\{x \precsim^?_{\mathcal{R},\lambda} s\} \uplus M; S; \alpha$ to $M; S \cup \{x \approx t\}; \alpha \wedge \beta$, we will construct t according to $x\vartheta$: if p is a position in t where by the definition of $\mathbf{pc}_{\mathcal{R},\lambda}(s)$ we should have a function symbol, then this symbol is chosen as the one that appears in $x\vartheta$ at position p. Otherwise t has a variable in p and such positions do not play a role in the proximity of t with s. All equations for the same x in S are constructed in this way. $\mathsf{Mer\text{-}M}$ merges them by replacing some variables by terms in $x\vartheta$. Therefore, if a subterm r occurring at position p in $x\sigma$ differs from the subterm at the same position p in $x\vartheta$, then r is a variable. Hence, $x\vartheta \Cap x\sigma = x\vartheta$ for all x in M. $\square$

Corollary 1. *Given an $(\mathcal{R}, \lambda)$-matching problem M and its solution ϑ, the algorithm $\mathfrak{M}$ computes a substitution σ such that $x\sigma \precsim_{\mathcal{R},\lambda} x\vartheta$ for all x in M.*

V. Concluding remarks

We designed class-based unification and matching algorithms for proximity relations in fully fuzzy signatures, where mismatches are permitted not only in symbol names but also in their arities, and proved their termination, soundness, and completeness. The decision problem is NP-hard for unification and its special well-moded case, which can be solved by matching. Proximity between arguments of distinct function symbols is expressed by argument relations. For unification, we require them to be correspondence relations in order not to have arguments skipped. For matching, there is no restriction: we can use arbitrary argument relations. Note that the requirement of using correspondence relations is not really a restriction since any argument relation can be extended into a correspondence by adding dummy arguments. Due to the lack of space, we did not elaborate on these details in the paper.

Our results can be seen as an attempt to extend proximity-based unification to equational theories. Our argument correspondence relations can be represented as a version of regular, collapse-free, shallow theories, which have been studied quite intensively in first-order equational unification, e.g., [13], [14].

We did not explicitly mention whether different argument relations are allowed between the same pair of function symbols, or whether a function symbol can be related to itself with a relation other than the identity. However, our unification and matching rules do not depend on these kind of assumptions. Such relations would be treated in the same way as we have in the paper. However, those relations play a role if we want to extend proximity-based unification to equational theories, since some well-known axioms such as, e.g., commutativity, can be encoded in them. For instance, we can declare $f \sim^{\{(1,1),(2,2)\}}_{\mathcal{R},1} f$ and $f \sim^{\{(1,2),(2,1)\}}_{\mathcal{R},0.8} f$ to express a fuzzy version of commutativity for f. A detailed investigation of these and related topics is subject of future work.

References

[1] F. A. Fontana and F. Formato, "A similarity-based resolution rule," *Int. J. Intell. Syst.*, vol. 17, no. 9, pp. 853–872, 2002. [Online]. Available: https://doi.org/10.1002/int.10067

[2] F. Formato, G. Gerla, and M. I. Sessa, "Similarity-based unification," *Fundam. Inform.*, vol. 41, no. 4, pp. 393–414, 2000. [Online]. Available: https://doi.org/10.3233/FI-2000-41402

[3] M. I. Sessa, "Approximate reasoning by similarity-based SLD resolution," *Theor. Comput. Sci.*, vol. 275, no. 1-2, pp. 389–426, 2002. [Online]. Available: https://doi.org/10.1016/S0304-3975(01)00188-8

[4] B. Dundua, T. Kutsia, M. Marin, and C. Pau, "Constraint solving over multiple similarity relations," in *5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference)*, ser. LIPIcs, Z. M. Ariola, Ed., vol. 167. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 30:1–30:19. [Online]. Available: https://doi.org/10.4230/LIPIcs.FSCD.2020.30

[5] H. Aït-Kaci and G. Pasi, "Fuzzy lattice operations on first-order terms over signatures with similar constructors: A constraint-based approach," *Fuzzy Sets Syst.*, vol. 391, pp. 1–46, 2020. [Online]. Available: https://doi.org/10.1016/j.fss.2019.03.019

[6] P. Julián-Iranzo and C. Rubio-Manzano, "Proximity-based unification theory," *Fuzzy Sets and Systems*, vol. 262, pp. 21–43, 2015. [Online]. Available: https://doi.org/10.1016/j.fss.2014.07.006

[7] T. Kutsia and C. 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*, ser. Lecture Notes in Computer Science, M. Gabbrielli, Ed., vol. 12042. Springer, 2019, pp. 107–122. [Online]. Available: https://doi.org/10.1007/978-3-030-45260-5_7

[8] P. Julián-Iranzo and F. Sáenz-Pérez, "An efficient proximity-based unification algorithm," in *2018 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2018*. IEEE, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/FUZZ-IEEE.2018.8491593

[9] M. E. Cornejo, J. Medina-Moreno, and C. Rubio-Manzano, "Towards a full fuzzy unification in the Bousi Prolog system," in *2018 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2018, Rio de Janeiro, Brazil, July 8-13, 2018*. IEEE, 2018, pp. 1–7. [Online]. Available: https://doi.org/10.1109/FUZZ-IEEE.2018.8491514

[10] F. Baader and T. Nipkow, *Term rewriting and all that*. Cambridge University Press, 1998.

[11] J. Krajícek and P. Pudlák, "The number of proof lines and the size of proofs in first order logic," *Arch. Math. Log.*, vol. 27, no. 1, pp. 69–84, 1988. [Online]. Available: https://doi.org/10.1007/BF01625836

[12] N. Dershowitz and Z. Manna, "Proving termination with multiset orderings," *Commun. ACM*, vol. 22, no. 8, pp. 465–476, 1979. [Online]. Available: https://doi.org/10.1145/359138.359142

[13] J. H. Siekmann, "Unification theory," *J. Symb. Comput.*, vol. 7, no. 3/4, pp. 207–274, 1989. [Online]. Available: https://doi.org/10.1016/S0747-7171(89)80012-4

[14] H. Comon, M. Haberstrau, and J. Jouannaud, "Syntacticness, cycle-syntacticness, and shallow theories," *Inf. Comput.*, vol. 111, no. 1, pp. 154–191, 1994. [Online]. Available: https://doi.org/10.1006/inco.1994.1043