Fuzzy Similarity and Proximity Constraint Solving in Commutative Theories

Besik Dundua[a,b], Demetre Labadze[a], Tornike Tsereteli[a]

[a]*School of Computer Science, Kutaisi International University, Kutaisi, Georgia*
[b]*Institute of Applied Mathematics, Tbilisi State University, Georgia*

Abstract

Constraint solving modulo equational theories plays a central role in symbolic computation, logic programming, and knowledge representation. In this paper, we extend classical constraint solving techniques by incorporating fuzzy similarity and proximity relations into constraint solving modulo commutativity. Our contributions include two constraint-solving algorithms: one for similarity-based constraint solving, allowing mismatches between function symbol names, and another for proximity-based constraint solving, which also supports mismatches in arity. Both algorithms are supported by formal termination, soundness, and completeness proofs and have been implemented in a user-friendly, web-based prototype. We further discuss how these methods enhance expressiveness and enable approximate reasoning in domains where rigid syntactic equality is insufficient.

Keywords: Fuzzy similarity and proximity relations, constraint solving, unification

1. Introduction

Constraints are fundamental to many areas of computer science because they provide a natural and powerful way to model problems. They define the conditions under which variables can interact, allowing complex requirements to be expressed declaratively. Two important types of constraints commonly used in logic-based and symbolic systems are equational constraints and membership constraints. Equational constraints specify that two terms must be equal, often within a given theory such as commutativity or associativity, while membership constraints express that a term must belong to a specific set, structure, or language.

In this paper, we focus on equational constraints in which symbols are connected through fuzzy relations. Specifically, we consider two types of fuzzy relations: fuzzy similarity relations and fuzzy proximity relations. Both generalize classical equality by introducing degrees of closeness or resemblance between terms, enabling more flexible and approximate forms of reasoning.

A fuzzy similarity relation is a special case of a fuzzy proximity relation that satisfies the transitivity property. While this imposes stronger restrictions, it also leads to more favorable computational properties, such as improved solvability and reduced search space in certain constraint-solving tasks. In contrast, fuzzy proximity relations relax the transitivity requirement, allowing them to capture more nuanced or context-sensitive forms of closeness. However, this added expressiveness often comes at the cost of increased computational complexity. For example, a constraint problem that is unsolvable under a given similarity relation $\mathcal{R}$ may become solvable when the same relation is interpreted as a non-transitive proximity relation. Our goal is to investigate constraint solving with both similarity and proximity relations within the framework of commutative theories

Solving equations modulo commutativity Baader (1989); Stickel (1981) is a fundamental problem with wide-ranging applications in computer science. It plays a central role in domains such as automated theorem proving, term rewriting, equational reasoning, cryptographic protocol analysis, natural language processing, planning, knowledge representation, program analysis, and software verification. Reasoning modulo commutativity is essential in these contexts because many real-world operations—such as addition and multiplication in arithmetic, or set union in logic—are inherently commutative. Effectively handling such operations requires constraint algorithms that respect the commutative properties of the involved functions. The practical relevance of commutative constraint solving is further emphasized by its diverse applications, as discussed in Baader and Nipkow (1998).

Constraints based on fuzzy similarity and fuzzy proximity relations enable the modeling of uncertainty and imprecision, providing a more flexible and expressive framework for representing real-world scenarios where exact matches are either unavailable or inappropriate. By incorporating fuzzy relations, our approach generalizes classical constraint-solving techniques to support approximate reasoning, which is particularly valuable in domains such as artificial intelligence, decision-making, and knowledge representation.

The study of equational constraints based on fuzzy similarity relations in standard form was first initiated by Sessa (2002), laying the foundational framework for incorporating approximate reasoning into constraint solving. This foundational work has since been significantly extended in multiple directions. Notably, Dundua et al. (2020) introduced a constraint-solving mechanism that operates under multiple distinct similarity relations, thereby enhancing the flexibility and applicability of the framework. A further contribution comes from Ehling and Kutsia (2024), who extended equational reasoning by introducing a similarity framework based on quantales. They investigated unification in shallow, subterm-collapse-free quantitative equational theories and developed rule-based algorithms for solving such problems over both generic and idempotent Lawvereian quantales. In a different direction, Aït-Kaci and Pasi (2020) proposed a model for managing similarity within fully fuzzy signatures. A related line of development is the FASILL programming language Julián-Iranzo et al. (2020), which integrates similarity-based weak unification with fuzzy connectives and aggregators to support flexible reasoning in logic programming.

Proximity constraint solving was introduced to handle approximate reasoning by relaxing strict syntactic equality, allowing terms to be considered equivalent based on proximity relations—reflexive and symmetric fuzzy relations that are not necessarily transitive. This approach has evolved into two main methodologies: the block-based and the class-based approaches. The block-based approach groups symbols into maximal cliques, treating symbols within the same block as interchangeable, and has been implemented in systems like Bousi Prolog Julián-Iranzo and Rubio-Manzano (2015); Cornejo et al. (2018); Julián-Iranzo and Sáenz-Pérez (2018). Conversely, the class-based approach, as explored by Kutsia and Pau, permits more flexible symbol associations by considering neighborhoods of function symbols, facilitating unification, matching, and anti-unification under proximity relations Kutsia and Pau (2019a,b); Pau and Kutsia (2021).

This paper extends our previous work accepted for publication at FUZZ-IEEE 2025 Dundua et al. (2025b). In Dundua et al. (2025b), we introduced a similarity-based constraint-solving algorithm modulo commutativity, which supports solving equations with mismatches between function symbol names. In the present paper, we generalize this approach to cover both similarity- and proximity-based constraint solving modulo commutativity. Moreover, we allow mismatches not only between symbol names but also between their arities, thereby significantly enhancing the expressiveness and applicability of constraint solving.

Specifically, our contributions are as follows:

- An algorithm for similarity-based constraint solving modulo commutativity, where mismatches are permitted only between symbol names, along with formal proofs of its properties;
- An algorithm for proximity-based constraint solving modulo commutativity, which supports mismatches both between symbol names and between arities, accompanied by formal proofs of its properties;
- An implementation of both algorithms in a user-friendly, web-based interfaceDundua et al. (2025a);
- A discussion of potential application domains and directions for future research and development.

2. Preliminaries

The alphabet $\mathcal{A}$ consists of three pairwise disjoint sets of symbols: $\mathcal{V}$, which contains variables denoted by $x, y, z, \ldots$; $\mathcal{F}_{\mathsf{u}}$, which includes unordered function symbols represented by $f_{\mathsf{u}}, g_{\mathsf{u}}, h_{\mathsf{u}}, \ldots$; and $\mathcal{F}_{\mathsf{o}}$, which comprises ordered function symbols denoted by $f_{\mathsf{o}}, g_{\mathsf{o}}, h_{\mathsf{o}}, \ldots$.

Function symbols, denoted by $f, g, h, \ldots$, are elements of the set $\mathcal{F} = \mathcal{F}_{\mathsf{u}} \cup \mathcal{F}_{\mathsf{o}}$. Both ordered and unordered function symbols have a fixed arity, meaning they accept a specific, predetermined number of arguments. Unordered symbols naturally generalize the commutativity property of terms and can be used to encode multisets, while ordered symbols are well-suited for modeling lists.

Terms are inductively defined over $\mathcal{A}$ as $t ::= x \mid f(t_1, \ldots, t_n)$. The set of terms over $\mathcal{V}$ and $\mathcal{F}$ is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. We represent terms using the letters t, r, and s. The set of variables of a term t is denoted by $var(t)$

For a given set S, we denote by S^* the set of finite, possibly empty, sequences of elements of S, and by S^n the set of sequences of length n of elements of S. The empty sequence of symbols from any set S is denoted by ϵ. Given a sequence $\overline{s} = (s_1, s_2, \ldots, s_n) \in S^n$, we denote by $perm(s_1, s_2, \ldots, s_n)$ the set of sequences $\{(s_{\pi(1)}, s_{\pi(2)}, \ldots, s_{\pi(n)}) \mid \pi$ is a permutation of $\{1, 2, \ldots, n\}\}$.

A substitution is a mapping from variables to terms such that all but finitely many variables are mapped to themselves. We use lowercase Greek letters σ, θ to denote them.

The application of a substitution σ to a term t, denoted by $t\sigma$, is $\sigma(x)$ if $t = x$ and is $f(t_1\sigma, \dots, t_n\sigma)$ if $t = f(t_1, \dots, t_n)$.

A binary fuzzy relation on a set S is a mapping from $S \times S$ to the real interval $[0, 1]$. If $\mathcal{R}$ is a fuzzy relation on S and λ is a number $0 < \lambda \leq 1$ (called cut value), then the λ-cut of $\mathcal{R}$ on S, denoted $\mathcal{R}_\lambda$, is an ordinary (crisp) relation on S defined as $\mathcal{R}_\lambda := \{(s_1, s_2) \mid \mathcal{R}(s_1, s_2) \geq \lambda\}$.

A fuzzy relation $\mathcal{R}$ on a set S is called a proximity relation, if it is reflexive and symmetric:

- **Reflexivity:** $\mathcal{R}(s, s) = 1$ for all $s \in S$;
- **Symmetry:** $\mathcal{R}(s_1, s_2) = \mathcal{R}(s_2, s_1)$ for all $s_1, s_2 \in S$.

Let $\wedge$ be a T-norm: an associative, commutative, non-decreasing binary operation on $[0, 1]$ with 1 as the unit element. A proximity relation (on S) is called a similarity relation (on S) iff it is transitive:

- **Transitivity** $\mathcal{R}(s_1, s_2) \geq \mathcal{R}(s_1, s) \wedge \mathcal{R}(s, s_2)$ for any $s_1, s_2, s \in S$.

In this paper, in the role of T-norm we take the minimum of two numbers, and write min instead of $\wedge$. In the role of S we take a syntactic domain, defined in the next section.

3. Fuzzy Proximity and Similarity Relations on Terms

We assume that a fuzzy proximity relation $\mathcal{R}_\mathcal{F}$ is defined on $\mathcal{F}$, satisfying the following conditions:

$$\mathcal{R}_\mathcal{F}(f, g) = 0 \quad \text{if} \begin{cases} f \in \mathcal{F}_\mathsf{o} \text{ and } g \in \mathcal{F}_\mathsf{u}, \\ f \in \mathcal{F}_\mathsf{u} \text{ and } g \in \mathcal{F}_\mathsf{o}, \\ f, g \in \mathcal{F}_\mathsf{o} \text{ and have different arities.} \end{cases}$$

Using the fuzzy proximity relation $\mathcal{R}_\mathcal{F}$, we define a fuzzy relation $\mathcal{R}_\text{prox}$ on terms as follows:

$$\mathcal{R}_{\text{prox}}(x,x) = 1, \qquad \text{where } x \in \mathcal{V}.$$

$$\mathcal{R}_{\text{prox}}\big(f_{\mathsf{o}}(t_1,\dots,t_n), g_{\mathsf{o}}(s_1,\dots,s_n)\big) =$$
$$\min\big(\mathcal{R}_{\mathcal{F}}(f_{\mathsf{o}},g_{\mathsf{o}}), \mathcal{R}_{\text{prox}}(t_1,s_1),\dots,\mathcal{R}_{\text{prox}}(t_n,s_n)\big), \quad \text{where } f_{\mathsf{o}}, g_{\mathsf{o}} \in \mathcal{F}_{\mathsf{o}}.$$

$$\mathcal{R}_{\text{prox}}\big(f_{\mathsf{u}}(t_1,\dots,t_n), g_{\mathsf{u}}(s_1,\dots,s_m)\big) =$$
$$\max_{\substack{\bar{t}\in perm(t'_1,\dots,t'_m)\\ \{t'_1,\dots,t'_m\}\subseteq\{t_1,\dots,t_n\}}} \min\big(\mathcal{R}_{\mathcal{F}}(f_{\mathsf{u}},g_{\mathsf{u}}), \mathcal{R}_{\text{prox}}\big(f_{\mathsf{o}}(\bar{t}), f_{\mathsf{o}}(s_1,\dots,s_m)\big)\big),$$

where $n \geq m$, $f_{\mathsf{u}}, g_{\mathsf{u}} \in \mathcal{F}_{\mathsf{u}}$ and $f_{\mathsf{o}} \in \mathcal{F}_{\mathsf{o}}$ is fresh.

$$\mathcal{R}_{\text{prox}}\big(f_{\mathsf{u}}(t_1,\dots,t_n), g_{\mathsf{u}}(s_1,\dots,s_m)\big) =$$
$$\mathcal{R}_{\text{prox}}\big(g_{\mathsf{u}}(s_1,\dots,s_m), f_{\mathsf{u}}(t_1,\dots,t_n)\big), \qquad \text{if } n < m.$$

$$\mathcal{R}_{\text{prox}}(t_1,t_2) = 0, \quad \text{for all } t_1,t_2 \in \mathcal{T}(\mathcal{F},\mathcal{V}), \text{ otherwise.}$$

Theorem 3.1. *The relation $\mathcal{R}_{prox}$ is a proximity relation on terms.*

Proof. We prove that the relation $\mathcal{R}_{\text{prox}}$, defined on terms, is a proximity relation—that is, it satisfies reflexivity, and symmetry. The proof proceeds by structural induction on the terms, using the definition of $\mathcal{R}_{\text{prox}}$ to establish each of the required properties.

□

Remark 1. *The fuzzy relation $\mathcal{R}_{prox}$ it is not transitive. To see this, take a fuzzy relation $\mathcal{R}_{\mathcal{F}}$ defined as $\mathcal{R}_{\mathcal{F}}(f_{\mathsf{u}}, g_{\mathsf{u}}) = 0.7$. Then $\mathcal{R}_{prox}(f_{\mathsf{u}}(a,b), g_{\mathsf{u}}(a)) = 0.7$ and $\mathcal{R}_{prox}(f_{\mathsf{u}}(a,c), g_{\mathsf{u}}(a)) = 0.7$, but $\mathcal{R}_{prox}(f_{\mathsf{u}}(a,b), f_{\mathsf{u}}(a,c)) = 0$. Note that $\mathcal{R}_{\mathcal{F}}$ is indeed a similarity relation. This example demonstrates that, in general, similarity does not extend from the alphabet to terms. However, as we will see below, it does extend if similarity is restricted to function symbols of the same arity within the alphabet.*

Based on this observation, we now assume that $\mathcal{R}_{\mathcal{F}}$ is a fuzzy similarity relation satisfying the following conditions:

$$\mathcal{R}_{\mathcal{F}}(f,g) = 0 \quad \text{if} \begin{cases} f \in \mathcal{F}_{\mathsf{o}} \text{ and } g \in \mathcal{F}_{\mathsf{u}}, \text{ or} \\ f \in \mathcal{F}_{\mathsf{u}} \text{ and } g \in \mathcal{F}_{\mathsf{o}}, \text{ or} \\ f, g \in \mathcal{F} \text{ and have different arities.} \end{cases}$$

We now define a fuzzy relation $\mathcal{R}_{\text{sim}}$ over terms, derived from the similarity relation $\mathcal{R}_{\mathcal{F}}$, as follows:

$$\mathcal{R}_{\text{sim}}(x,x) = 1, \qquad \text{where } x \in \mathcal{V}.$$

$$\mathcal{R}_{\text{sim}}\big(f_{\mathsf{o}}(t_1,\dots,t_n), g_{\mathsf{o}}(s_1,\dots,s_n)\big) =$$
$$\min\big(\mathcal{R}_{\mathcal{F}}(f_{\mathsf{o}},g_{\mathsf{o}}), \mathcal{R}_{\text{sim}}(t_1,s_1),\dots,\mathcal{R}_{\text{sim}}(t_n,s_n)\big), \quad \text{where } f_{\mathsf{o}}, g_{\mathsf{o}} \in \mathcal{F}_{\mathsf{o}}.$$

$$\mathcal{R}_{\text{sim}}\big(f_{\mathsf{u}}(t_1,\dots,t_n), g_{\mathsf{u}}(s_1,\dots,s_n)\big) =$$
$$\max_{\bar{t} \in perm(t_1,\dots,t_n)} \min\big(\mathcal{R}_{\mathcal{F}}(f_{\mathsf{u}},g_{\mathsf{u}}), \mathcal{R}_{\text{sim}}\big(f_{\mathsf{o}}(\bar{t}), f_{\mathsf{o}}(s_1,\dots,s_n)\big)\big),$$

where $f_{\mathsf{u}}, g_{\mathsf{u}} \in \mathcal{F}_{\mathsf{u}}$ and $f_{\mathsf{o}} \in \mathcal{F}_{\mathsf{o}}$ is a fresh symbol with arity n.

$$\mathcal{R}_{\text{sim}}(t_1,t_2) = 0, \quad \text{for all } t_1, t_2 \in \mathcal{T}(\mathcal{F},\mathcal{V}), \text{ otherwise.}$$

Theorem 3.2. *The relation $\mathcal{R}_{sim}$ is a similarity relation on terms.*

Proof. We prove that the relation $\mathcal{R}_{\text{sim}}$, defined on terms, is a similarity relation—that is, it satisfies reflexivity, symmetry, and transitivity. The proof proceeds by structural induction on the terms, using the definition of $\mathcal{R}_{\text{sim}}$ to establish each of the required properties.

□

4. Fuzzy Proximity and Similarity Constraints

Primitive constraints $\mathcal{P}$ are defined by the grammar

$$\mathcal{P} ::= \mathsf{true} \mid \mathsf{false} \mid t \simeq_{\mathcal{R},\lambda} r,$$

where $t, r \in \mathcal{T}(\mathcal{F},\mathcal{V})$, $0 < \lambda \leq 1$ is a real number, and $\mathcal{R}$ is a fuzzy relation. A primitive constraint of the form $t \simeq_{\mathcal{R},\lambda} r$ is called a similarity primitive constraint if $\mathcal{R}$ is a fuzzy similarity relation $\mathcal{R}_{\text{sim}}$, and a proximity primitive constraint if $\mathcal{R}$ is a fuzzy proximity relation $\mathcal{R}_{\text{prox}}$. A constraint $\mathcal{C}$ over $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is defined as $\mathcal{C} ::= \mathcal{P} \mid \mathcal{C} \wedge \mathcal{C} \mid \mathcal{C} \vee \mathcal{C} \mid \exists x.\mathcal{C}$. We denote by $\mathcal{K}$ a conjunction of primitive constraints $t_1 \simeq_{\mathcal{R},\lambda} r_1 \wedge \cdots \wedge t_n \simeq_{\mathcal{R},\lambda} r_n$, all with the same relation $\mathcal{R}$ and the same λ-cut.

The domain D for the intended interpretation of our constraint language is the Herbrand universe, which is the set of ground terms. Each n-ary ordered function symbol $f_{\mathsf{o}} \in \mathcal{F}_{\mathsf{o}}$ is interpreted as a function that maps $(t_1, \ldots, t_n) \in D^n$ to a ground term $f(t_1, \ldots, t_n) \in D$, and each n-ary unordered function symbol $f_{\mathsf{u}} \in \mathcal{F}_{\mathsf{u}}$ is interpreted as a function that maps $(t_1, \ldots, t_n) \in D^n$ to a ground term $f(\overline{s}) \in D$ where $\overline{s} \in \mathit{perm}(t_1, \ldots, t_n)$. The relation $\simeq_{\mathcal{R},\lambda}$ is interpreted as a fuzzy relation on the domain as defined in the previous section.

Substitution composition is defined as the composition of mappings. We write $\sigma\vartheta$ for the composition of σ with ϑ. This operation is associative but not commutative. A substitution σ is $(\mathcal{R}, \lambda)$ more general than ϑ, written as $\sigma \preceq_{\mathcal{R},\lambda} \vartheta$, if there exists a substitution φ such that $x\sigma\varphi \simeq_{\mathcal{R},\lambda} x\vartheta$.

Definition 4.1. *A substitution σ is called a $(\mathcal{R}, \lambda)$ solution of a primitive constraint $t \simeq_{\mathcal{R},\lambda} r$ with a degree $\mathfrak{d}$, if $\mathcal{R}(t\sigma, r\sigma) \geq \lambda$ and $\mathfrak{d} = \mathcal{R}(t\sigma, r\sigma)$.*

Any substitution is a solution of true, while false has no solution. A substitution σ is a $(\mathcal{R}, \lambda)$ solution of a conjunction of primitive constraints $t_1 \simeq_{\mathcal{R},\lambda} r_1 \wedge \cdots \wedge t_n \simeq_{\mathcal{R},\lambda} r_n$ with degree $\mathfrak{d}$ iff $\mathcal{R}(t_i\sigma, r_i\sigma) \geq \lambda$ for each $1 \leq i \leq n$ and $\mathfrak{d} = \min(\mathcal{R}(t_1\sigma, r_1\sigma), \ldots, \mathcal{R}(t_n\sigma, r_n\sigma))$. We denote the set of all solutions of $\mathcal{K}$ by $\mathsf{solve}(\mathcal{K})$. For a constraint $\mathcal{C} = \mathcal{K}_1 \vee \cdots \vee \mathcal{K}_m$ in disjunctive normal form (DNF), we define $\mathsf{solve}(\mathcal{C}) = \bigcup_{i=1}^{m} \mathsf{solve}(\mathcal{K}_i)$.

Definition 4.2. *A substitution σ is called a $(\mathcal{R}, \lambda)$ maximal solution of $\mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ with a degree $\mathfrak{d}'$, if there exists some $\mathcal{K}_i$ such that σ solves $\mathcal{K}_i$ with degree $\mathfrak{d}'$, and there is no $\mathcal{K}_j$ for which σ solves $\mathcal{K}_j$ with a strictly greater degree, i.e., $\mathfrak{d}'' > \mathfrak{d}'$.*

5. Similarity-Based Constraint Solving

In this section, we consider constraints built over similarity primitive constraints. The similarity constraint-solving algorithm, $\mathsf{SolveSim}$, described below, operates on pairs consisting of a conjunction of primitive constraints and a real number from the interval $(0, 1]$. In particular, the rules have the form $\langle \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle \mathcal{K}''_1; \mathfrak{d}''\rangle \vee \cdots \vee \langle \mathcal{K}''_n; \mathfrak{d}''\rangle$, which defines the transformation $\langle \mathcal{K}; \mathfrak{d}\rangle \vee \mathcal{C} \rightsquigarrow_{\mathfrak{d}'} \langle \mathcal{K}''_1; \mathfrak{d}''\rangle \vee \cdots \vee \langle \mathcal{K}''_n; \mathfrak{d}''\rangle \vee \mathcal{C}$ where $\mathcal{C}$ has a form $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_m; \mathfrak{d}_m\rangle$. The rules are applied considering the associativity, commutativity, and idempotence of $\vee$ and $\wedge$, with false as the identity element for $\vee$. In the rules, the superscript ? indicates that the constraints are supposed to be solved.

Del-Var-Sim: **Variable Deletion, Similarity**

$\langle x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} x \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_1 \langle \mathcal{K}; \mathfrak{d}\rangle, \qquad \text{where } x \in \mathcal{V}.$

Dec-Ord-Sim: **Ordered Decomposition, Similarity**

$\langle f_{\mathsf{o}}(t_1,\ldots,t_n) \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} g_{\mathsf{o}}(r_1,\ldots,r_n) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'}$
$\langle t_1 \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_1 \wedge \cdots \wedge t_n \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_n \wedge \mathcal{K}; \min(\mathfrak{d}, \mathfrak{d}')\rangle,$
where $n \geq 0$, $\mathcal{R}_{\text{sim}}(f_{\mathsf{o}}, g_{\mathsf{o}}) \geq \lambda$ and $\mathfrak{d}' = \mathcal{R}_{\text{sim}}(f_{\mathsf{o}}, g_{\mathsf{o}})$.

Dec-Urd-Sim: **Unordered Decomposition, Similarity**

$\langle f_{\mathsf{u}}(t_1,\ldots,t_n) \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} g_{\mathsf{u}}(r_1,\ldots,r_n) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'}$
$\bigvee_{(t'_1,\ldots,t'_n)\in perm(t_1,\ldots,t_n)} \langle t'_1 \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_1 \wedge \cdots \wedge t'_n \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_n \wedge \mathcal{K}; \min(\mathfrak{d}, \mathfrak{d}')\rangle,$
where $n \geq 0$, $\mathcal{R}_{\text{sim}}(f_{\mathsf{u}}, g_{\mathsf{u}}) \geq \lambda$ and $\mathfrak{d}' = \mathcal{R}_{\text{sim}}(f_{\mathsf{u}}, g_{\mathsf{u}})$.

Ori-Var-Sim: **Variable Orient, Similarity**

$\langle t \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} x \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_1 \langle x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}; \mathfrak{d}\rangle, \qquad \text{where } t \notin \mathcal{V}.$

Conf-FS-Sim: **Function Symbol Conflict, Similarity**

$\langle f(t_1,\ldots,t_n) \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} g(r_1,\ldots,r_m) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \mathsf{false},$
where $\mathfrak{d}' = \mathcal{R}_{\text{sim}}(f, g) < \lambda$.

Occ-Ch-Sim: **Occur Check, Similarity**

$\langle x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_0 \mathsf{false}, \qquad \text{where } x \in var(t) \text{ and } x \neq t\ .$

Sol-Con-Sim: **Solve Constraint, Similarity**

$\langle x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_1 \langle x \simeq_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}\{x \mapsto t\}; \mathfrak{d}\rangle,$
where $x \notin var(t)$, $x \neq t$.

In order to solve $t \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r$, we create an initial system $\langle t \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r; 1\rangle$ and apply successively SolveSim rules. The SolveSim rules are constructed such that only one rule is applicable to each system.

Example 1. *Let* $\mathcal{C} = f_u(f_o(x,b), h_o(x,y), z) \simeq_{\mathcal{R},0.4} g_u(g_o(a,y), b, f_o(x,z))$ *and let* $\mathcal{R}$ *be defined as* $\mathcal{R}(f_u,g_u) = 0.6$*,* $\mathcal{R}(f_o,g_o) = 0.5$*,* $\mathcal{R}(f_o,h_o) = 0.4$*, and for every* $s \in \{f_u, f_o, g_u, g_o, h_o\}$*,* $\mathcal{R}(b,s) = 0.3$*. Then* SolveSim *performs the following derivation:*

$$\langle \mathcal{C}; 1\rangle \rightsquigarrow_{\textit{(Dec\_Urd\_Sim)}}$$
$$\langle f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y) \wedge \underline{h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} b} \wedge z \simeq^{?}_{\mathcal{R},0.4} f_o(x,z); 0.6\rangle \vee$$
$$\langle f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y) \wedge h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z) \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.6\rangle \vee$$
$$\langle \underline{f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} b} \wedge h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y) \wedge z \simeq^{?}_{\mathcal{R},0.4} f_o(x,z); 0.6\rangle \vee$$
$$\langle \underline{f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} b} \wedge h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z) \wedge z \simeq^{?}_{\mathcal{R},0.4} g_o(a,y); 0.6\rangle \vee$$
$$\langle f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z) \wedge h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y) \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.6\rangle \vee$$
$$\langle f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z) \wedge \underline{h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} b} \wedge z \simeq^{?}_{\mathcal{R},0.4} g_o(a,y); 0.6\rangle \rightsquigarrow^{*}_{\textit{Conf\_FS\_Sim}}$$
$$\langle \underline{f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y)} \wedge h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z) \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.6\rangle \vee$$
$$\langle f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z) \wedge h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y) \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.6\rangle \rightsquigarrow_{\textit{Dec\_Ord\_Sim}}$$
$$\langle x \simeq^{?}_{\mathcal{R},0.4} a \wedge b \simeq^{?}_{\mathcal{R},0.4} y \wedge \underline{h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z)} \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.5\rangle \vee$$
$$\langle f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z) \wedge h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y) \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.5\rangle \rightsquigarrow_{\textit{Dec\_Ord\_Sim}}$$
$$\langle x \simeq^{?}_{\mathcal{R},0.4} a \wedge \underline{b \simeq^{?}_{\mathcal{R},0.4} y} \wedge x \simeq^{?}_{\mathcal{R},0.4} x \wedge y \simeq^{?}_{\mathcal{R},0.4} z \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.5\rangle \vee$$
$$\langle f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z) \wedge h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y) \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.5\rangle \rightsquigarrow_{\textit{Ori\_Var\_Sim}}$$
$$\langle \underline{x \simeq^{?}_{\mathcal{R},0.4} a} \wedge \underline{y \simeq^{?}_{\mathcal{R},0.4} b} \wedge x \simeq^{?}_{\mathcal{R},0.4} x \wedge y \simeq^{?}_{\mathcal{R},0.4} z \wedge \underline{z \simeq^{?}_{\mathcal{R},0.4} b}; 0.5\rangle \vee$$
$$\langle f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z) \wedge h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y) \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.5\rangle \rightsquigarrow^{*}_{\textit{Sol\_Con\_Sim}}$$
$$\langle x \simeq_{\mathcal{R},0.4} a \wedge y \simeq_{\mathcal{R},0.4} b \wedge \underline{x \simeq^{?}_{\mathcal{R},0.4} x} \wedge \underline{b \simeq^{?}_{\mathcal{R},0.4} b} \wedge z \simeq_{\mathcal{R},0.4} b; 0.5\rangle \vee$$
$$\langle f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z) \wedge h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y) \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.5\rangle \rightsquigarrow^{*}_{\textit{Del\_Var\_Sim}}$$
$$\langle x \simeq_{\mathcal{R},0.4} a \wedge y \simeq_{\mathcal{R},0.4} b \wedge z \simeq_{\mathcal{R},0.4} b; 0.5\rangle \vee$$
$$\langle \underline{f_o(x,b) \simeq^{?}_{\mathcal{R},0.4} f_o(x,z)} \wedge \underline{h_o(x,y) \simeq^{?}_{\mathcal{R},0.4} g_o(a,y)} \wedge z \simeq^{?}_{\mathcal{R},0.4} b; 0.5\rangle \rightsquigarrow^{*}_{\textit{Dec\_Ord\_Sim}}$$
$$\langle x \simeq_{\mathcal{R},0.4} a \wedge y \simeq_{\mathcal{R},0.4} b \wedge z \simeq_{\mathcal{R},0.4} b; 0.5\rangle$$
$$\langle x \simeq^{?}_{\mathcal{R},0.4} x \wedge b \simeq^{?}_{\mathcal{R},0.4} z \wedge \underline{x \simeq^{?}_{\mathcal{R},0.4} a} \wedge y \simeq^{?}_{\mathcal{R},0.4} y \wedge \underline{z \simeq^{?}_{\mathcal{R},0.4} b}; 0.5\rangle \rightsquigarrow_{\textit{Sol\_Con\_Sim}}$$
$$\langle x \simeq_{\mathcal{R},0.4} a \wedge y \simeq_{\mathcal{R},0.4} b \wedge z \simeq_{\mathcal{R},0.4} b; 0.5\rangle$$
$$\langle \underline{x \simeq^{?}_{\mathcal{R},0.4} x} \wedge \underline{b \simeq^{?}_{\mathcal{R},0.4} b} \wedge x \simeq_{\mathcal{R},0.4} a \wedge \underline{y \simeq^{?}_{\mathcal{R},0.4} y} \wedge z \simeq_{\mathcal{R},0.4} b; 0.5\rangle \rightsquigarrow_{\textit{Del\_Var\_Sim}}$$
$$\langle x \simeq_{\mathcal{R},0.4} a \wedge y \simeq_{\mathcal{R},0.4} b \wedge z \simeq_{\mathcal{R},0.4} b; 0.5\rangle \vee$$
$$\langle x \simeq_{\mathcal{R},0.4} a \wedge z \simeq_{\mathcal{R},0.4} b; 0.5\rangle$$

Below we study the properties of the similarity-based constraint solving algorithm.

Theorem 5.1. *(Termination)*
The algorithm SolveSim *terminates on any input constraint.*

Proof. We define a complexity measure for a system $cm(\mathcal{K}_1; \mathfrak{d}_1 \vee \cdots \vee \mathcal{K}_n; \mathfrak{d}_n)$ where $\mathcal{K}_i$ is a conjunction of primitive constraints and $\mathfrak{d}_i$ is a degree, and show that $cm(\mathcal{K}_1; \mathfrak{d}_1 \vee \cdots \vee \mathcal{K}_n; \mathfrak{d}_n) > cm(\mathcal{K}'_1; \mathfrak{d}'_1 \vee \cdots \vee \mathcal{K}'_m; \mathfrak{d}'_m)$ holds whenever $\langle \mathcal{K}_1; \mathfrak{d}_1 \vee \cdots \vee \mathcal{K}_n; \mathfrak{d}_n \rangle \rightsquigarrow \langle \mathcal{K}'_1; \mathfrak{d}'_1 \vee \cdots \vee \mathcal{K}'_m; \mathfrak{d}'_m \rangle$.

The complexity measure $cm(\mathcal{K})$ of a conjunction of primitive constraints $\mathcal{K}$ is the tuple $\langle N_1, N_2, N_3 \rangle$ defined as follows ($\{\!|\,|\!\}$ stands for a multiset):

- N_1 is the number of unsolved variables in $\mathcal{K}$.
- $N_2 := \{\!| \mathit{size}(t) + \mathit{size}(r) \mid t \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r \in \mathcal{K} |\!\}$, where *size* of a term denotes the denotational length of t.
- N_3 the number of equations in the form t $\simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} x$.

The complexity measure $cm(\mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n)$ is defined as the multiset $\{\!| cm(\mathcal{K}_1), \ldots, cm(\mathcal{K}_n) |\!\}$. Measures are compared by multiset extension of the lexicographic ordering on tuples. The table below shows which rule reduces which component of the measure, which implies termination of the algorithm solve.

Rule	$\mathbf{N_1}$	$\mathbf{N_2}$	$\mathbf{N_3}$
Sol-Con-Sim	$>$		
Del-Var-Sim, Dec-OFS-Sim, Dec-UFS-Sim, Conf-FS-Sim, Occ-Var-Sim	$\geq$	$>$	
Ori-Var-Sim	$\geq$	$\geq$	$>$

□

Definition 5.1. *A conjunction of similarity primitive constraints $\mathcal{K}$ is in solved form if $\mathcal{K}$ is either* true *or each primitive constraint in $\mathcal{K}$ has a form $x \simeq_{\mathcal{R}_{sim},\lambda} t$, where x appears only once in $\mathcal{K}$. A constraint in DNF $\mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ is in solved form if each $\mathcal{K}_i$ is in solved form. For a conjunction of primitive constraints $\mathcal{K}$ in solved form, its corresponding substitution is denoted by $\sigma_{\mathcal{K}}$.*

Theorem 5.2. *If* $\mathsf{SolveSim}$ *reduces* $\langle \mathcal{K}; 1\rangle \rightsquigarrow^* \langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ *then each* $\mathcal{K}_j$ *is either in solved form or is* false.

Proof. By Theorem 6.1, each $\mathcal{K}_j$ is in a normal form. Assume by contradiction that $\mathcal{K}_j$ is not in solved form. By inspection of the solver rules, based on the definition of solved constraints, we can see that there is a rule that applies to $\mathcal{K}_j$. But this contradicts the fact that $\mathcal{K}_j$ is in a normal form. Hence, $\mathcal{K}_j$ is solved form.

□

Lemma 5.3. *(Soundness lemma for* $\mathsf{SolveSim}$*)*
Let $\mathcal{R}_{sim}$ *be a similarity relation,* λ *a cut value and* $\langle \mathcal{K}; \mathfrak{d}\rangle$ *a system. If* $\langle \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ *is a step performed by the solver* $\mathsf{SolveSim}$ *and* σ *is a maximal solution of* $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ *with degree* $\mathfrak{d}''$*, then* σ *is a solution of* $\mathcal{K}$ *with degree* $\min(\mathfrak{d}'', \mathfrak{d}')$.

Proof. By case distinction on the inference rules of the solver. We illustrate here three cases, when the selected rules are `Dec-OFS-Sim`, `Dec-UFS-Sim` and `Sol-Con-Sim`. For other rules the lemma can be shown in a similar manner.

- For failing rules it is trivial as false has no solution.
- Assume $\mathsf{SolveSim}$ transforms the given system by `Dec-OFS-Sim` rule, i.e., we have $\langle f_{\mathsf{o}}(t_1, \ldots, t_n) \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} g_{\mathsf{o}}(r_1, \ldots, r_n) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle t_1 \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r_1 \wedge \cdots \wedge t_n \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r_n \wedge \mathcal{K}; \min(\mathfrak{d}, \mathfrak{d}')\rangle$

  Then $\mathfrak{d}' = \mathcal{R}_{\mathrm{sim}}(f_{\mathsf{o}}, g_{\mathsf{o}}) \geq \lambda$. Let σ be a $(\mathcal{R}_{\mathrm{sim}}, \lambda)$ solution of $t_1 \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r_1 \wedge \cdots \wedge t_n \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r_n \wedge \mathcal{K}$ with degree $\mathfrak{d}''$ and $S = \{e \mid \text{e is part of } \mathcal{K}\}$. This means $\min(\bigcup_{i=1}^n \mathcal{R}_{\mathrm{sim}}(t_i\sigma, r_i\sigma) \cup \bigcup_{(t \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r) \in S} \mathcal{R}_{\mathrm{sim}}(t\sigma, r\sigma)) = \mathfrak{d}'' \geq \lambda$. Since, $\mathfrak{d}' \geq \lambda$ and $\mathfrak{d}'' \geq \lambda$ we have $\min(\mathfrak{d}'', \mathfrak{d}') \geq \lambda$. Moreover,

$$\begin{aligned}
\min(\mathfrak{d}'', \mathfrak{d}') = \quad & \min\left(\bigcup_{i=1}^{n} \mathcal{R}_{\mathrm{sim}}(t_i\sigma, r_i\sigma) \cup \right. \\
& \left. \bigcup_{(t \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r) \in S} \mathcal{R}_{\mathrm{sim}}(t\sigma, r\sigma), \mathfrak{d}' \right) \\
= \quad & \min\left(\bigcup_{i=1}^{n} \mathcal{R}_{\mathrm{sim}}(t_i\sigma, r_i\sigma) \cup \right. \\
& \left. \bigcup_{(t \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r) \in S} \mathcal{R}_{\mathrm{sim}}(t\sigma, r\sigma), \mathcal{R}_{\mathrm{sim}}(f_{\mathsf{o}}, g_{\mathsf{o}}) \right) \\
= \quad & \min\left(\{\mathcal{R}_{\mathrm{sim}}(f_{\mathsf{o}}(t_1, \dots, t_n)\sigma, \right. \\
& g_{\mathsf{o}}(r_1, \dots, r_m)\sigma)\} \cup \\
& \left. \bigcup_{(t \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r) \in S} \mathcal{R}_{\mathrm{sim}}(t\sigma, r\sigma) \right)
\end{aligned}$$

which implies that σ is a solution of
$f_{\mathsf{o}}(t_1, \dots, t_n) \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} g_{\mathsf{o}}(r_1, \dots, r_n) \wedge \mathcal{K}$ with the degree $\min(\mathfrak{d}'', \mathfrak{d}')$

- Assume SolveSim transforms given system by `Dec-UFS-Sim` rule, i.e., we have

$$\langle f_{\mathsf{u}}(t_1, \dots, t_n) \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} g_{\mathsf{u}}(r_1, \dots, r_n) \wedge \mathcal{K}; \mathfrak{d} \rangle \rightsquigarrow_{\mathfrak{d}'}$$
$$\bigvee_{(t'_1, \dots, t'_n) \in perm(t_1, \dots, t_n)} \langle t'_1 \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r_1 \wedge \dots \wedge t'_m \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r_m \wedge \mathcal{K}; \min(\mathfrak{d}, \mathfrak{d}') \rangle$$

Then $\mathfrak{d}' = \mathcal{R}_{\mathrm{sim}}(f_{\mathsf{u}}, g_{\mathsf{u}}) \geq \lambda$. Let σ be a $(\mathcal{R}, \lambda)$ maximal solution of $\bigvee_{(t'_1, \dots, t'_n) \in perm(t_1, \dots, t_n)} \langle t'_1 \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r_1 \wedge \dots \wedge t'_m \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r_m \wedge \mathcal{K} \rangle$ with degree $\mathfrak{d}''$. According to the Definition 4.2, σ is a $(\mathcal{R}_{\mathrm{sim}}, \lambda)$ solution of a conjuct $t'_1 \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r_1 \wedge \dots \wedge t'_n \simeq^?_{\mathcal{R}_{\mathrm{sim}},\lambda} r_n \wedge \mathcal{K}$ with degree $\mathfrak{d}''$. This means there is no other conjuct for which σ solves it with a strictly greater degree. Therefore, according to the Definition of $\mathcal{R}_{\mathrm{sim}}$ for terms where top symbols are unordered fuction symbols, we conclude that σ is $(\mathcal{R}_{\text{sim}}, \lambda)$ solution of $f_{\mathsf{u}}(t_1, \ldots, t_n) \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} g_{\mathsf{u}}(r_1, \ldots, r_n) \wedge \mathcal{K}$ with degree $\min(\mathfrak{d}'', \mathfrak{d}')$.

- Assume SolveSim transforms given system by `Sol-Con-Sim` rule, i.e., we have $\langle x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}\{x \mapsto t\}; \mathfrak{d}\rangle$ where $\mathfrak{d}' = 1$. Let σ be a $(\mathcal{R}_{\text{sim}}, \lambda)$ solution $x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}\{x \mapsto t\}$ with degree $\mathfrak{d}''$. This means $\mathfrak{d}'' = \min(\mathcal{R}_{\text{sim}}(x\sigma, t\sigma) \cup \bigcup_{s \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r \in S} \mathcal{R}_{\text{sim}}(s\sigma, r\sigma))$, where $S = \{e \mid \text{e is a conjunct of } \mathcal{K}\{x \mapsto t\}\}$. It follows from the soundness of weak unification algorithm given in Sessa (2002), that σ is $(\mathcal{R}_{\text{sim}}, \lambda)$ solution of $x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}$ with degree $\mathfrak{d}''$. Since $\mathfrak{d}' = 1$, we conclude σ is $(\mathcal{R}_{\text{sim}}, \lambda)$ solution of $x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}$ with degree $\min(\mathfrak{d}'', \mathfrak{d}')$. Since $\mathfrak{d}'' \geq \lambda$ and $\mathfrak{d}' \geq \lambda$ we have $\min(\mathfrak{d}', \mathfrak{d}'') \geq \lambda$

$\square$

Theorem 5.4. *(Soundness theorem for* SolveSim*)*
If SolveSim *reduces* $\langle \mathcal{K}; 1\rangle \rightsquigarrow^* \langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ *where each* $\mathcal{K}_i$ *is in solved form and* σ *is a maximal solution of* $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ *with degree* $\mathfrak{d}'$ *then* σ *is a solution* $\mathcal{K}$ *with degree* $\min(\mathfrak{d}', \mathfrak{d}_i)$.

Proof. By induction on the length of a rule application sequence leading from $\langle \mathcal{K}; 1\rangle$ to $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ using the soundness lemmas. $\square$

Lemma 5.5. *(Completeness Lemma for* SolveSim*) Let* $\mathcal{K}_0$ *be a conjunction of primitive constraints and* θ *is a* $(\mathcal{R}_{sim}, \lambda)$ *solution of* $\mathcal{K}_0$ *with degree* $\mathfrak{d}$. *Then we can make a step* $\langle \mathcal{K}_0; \mathfrak{d}_0\rangle \rightsquigarrow_{\mathfrak{d}'} \langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$, *such that there exists a* $(\mathcal{R}_{sim}, \lambda)$ *substitution* $\sigma \preceq_{\mathcal{R}_{sim},\lambda} \theta$ *that is a maximal solution of* $\mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ *with a degree* $\mathfrak{d}''$ *such that* $\mathfrak{d} = \min(\mathfrak{d}'', \mathfrak{d}')$.

Proof. By case distinction on the inference rules of the solver. We illustrate here three cases, when the selected rules are Dec-OFS-Sim, Dec-UFS-Sim and Sol-Con-Sim. For other rules the lemma can be shown in a similar manner.

- For failing rules it is trivial as false has no solution.
- Assume SolveSim transforms given system by `Dec-OFS-Sim` rule, i.e., we have $\langle f_{\mathsf{o}}(t_1, \ldots, t_n) \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} g_{\mathsf{o}}(r_1, \ldots, r_n) \wedge \mathcal{K}; \mathfrak{d}_1\rangle \rightsquigarrow_{\mathfrak{d}'} \langle t_1 \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_1 \wedge \cdots \wedge t_n \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_n \wedge \mathcal{K}; \min(\mathfrak{d}_1, \mathfrak{d}')\rangle$. Then $\mathfrak{d}' = \mathcal{R}_{\text{sim}}(f_{\mathsf{o}}, g_{\mathsf{o}})$. Let θ be a $(\mathcal{R}_{\text{sim}}, \lambda)$ solution of $f_{\mathsf{o}}(t_1, \ldots, t_n) \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} g_{\mathsf{o}}(r_1, \ldots, r_n) \wedge \mathcal{K}$ with degree

$\mathfrak{d}$. Since $\mathcal{K}$ does not change during transformation, we can assume θ is a $(\mathcal{R}_{\text{sim}}, \lambda)$ solution of $f_{\mathsf{o}}(t_1, \dots, t_n) \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} g_{\mathsf{o}}(r_1, \dots, r_n)$ That means $\mathcal{R}_{\text{sim}}(f_{\mathsf{o}}(t_1\theta, \dots, t_n\theta), g_{\mathsf{o}}(r_1\theta, \dots, r_n\theta)) = \mathfrak{d} \geq \lambda$. Therefore, according to the definition of $\mathcal{R}_{\text{sim}}$ we have $\mathcal{R}_{\text{sim}}(f_{\mathsf{o}}(t_1\theta, \dots, t_n\theta), g_{\mathsf{o}}(r_1\theta, \dots, r_n\theta))$ $= \min(\mathcal{R}_{\text{sim}}(f_{\mathsf{o}}, g_{\mathsf{o}}), \mathcal{R}_{\text{sim}}(t_1\theta, r_1\theta), \dots, \mathcal{R}_{\text{sim}}(t_n\theta, r_n\theta))$. This means θ also solves $t_1 \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_1, \dots, t_n \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_n$ with a degree $\mathfrak{d}''$, such that $\mathfrak{d}''$ satisfies the equality $\mathfrak{d} = \min(\mathfrak{d}'', \mathfrak{d}')$ and we conclude this case.

- Assume $\mathsf{SolveSim}$ transforms given system by `Dec-UFS-Sim` rule, i.e., we have

$$\langle f_{\mathsf{u}}(t_1, \dots, t_n) \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} g_{\mathsf{u}}(r_1, \dots, r_n) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \bigvee_{(t'_1,\dots,t'_n)\in perm(t_1,\dots,t_n)} \langle t'_1 \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_1 \wedge \cdots \wedge t'_n \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_n \wedge \mathcal{K}; \min(\mathfrak{d}, \mathfrak{d}')\rangle$$

Then $\mathfrak{d}' = \mathcal{R}_{\text{sim}}(f_{\mathsf{u}}, g_{\mathsf{u}})$. Let θ be a $(\mathcal{R}_{\text{sim}}, \lambda)$ solution of $f_{\mathsf{u}}(t_1, \dots, t_n) \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} g_{\mathsf{u}}(r_1, \dots, r_n) \wedge \mathcal{K}$ with degree $\mathfrak{d}$. Since $\mathcal{K}$ does not change during transformation, we can assume θ is a $(\mathcal{R}_{\text{sim}}, \lambda)$ solution of $f_{\mathsf{u}}(t_1, \dots, t_n) \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} g_{\mathsf{u}}(r_1, \dots, r_n)$. That means $\mathcal{R}_{\text{sim}}(f_{\mathsf{u}}(t_1\theta, \dots, t_n\theta),$ $g_{\mathsf{u}}(r_1\theta, \dots, r_n\theta)) = \mathfrak{d} \geq \lambda$. Therefore, according to the definition of $\mathcal{R}_{\text{sim}}$ we have

$$\mathcal{R}_{\text{sim}}(f_{\mathsf{u}}(t_1\theta, \dots, t_n\theta), g_{\mathsf{u}}(r_1\theta, \dots, r_n\theta)) = \max_{\bar{t}\in perm(t_1,\dots,t_n)} \min\left(\mathcal{R}_{\text{sim}}(f_{\mathsf{u}}, g_{\mathsf{u}}),\, \mathcal{R}_{\text{sim}}\left(f_{\mathsf{o}}(\bar{t}\theta), f_{\mathsf{o}}(r_1\theta, \dots, r_n\theta)\right)\right)$$

This means, that θ is a $(\mathcal{R}_{\text{sim}}, \lambda)$ maximal solution of

$$\bigvee_{(t'_1,\dots,t'_n)\in perm(t_1,\dots,t_n)} \langle t'_1 \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_1 \wedge \cdots \wedge t'_n \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} r_n\rangle$$

with degree $\mathfrak{d}''$, where $\bar{t} = (t'_1, \dots, t'_n)$. We know, that $\mathfrak{d}' = \mathcal{R}_{\text{sim}}(f_{\mathsf{u}}, g_{\mathsf{u}})$ and therefore we conclude, that $\mathfrak{d}''$ satisfies the equality $\mathfrak{d} = \min(\mathfrak{d}'', \mathfrak{d}')$,

- Assume $\mathsf{SolveSim}$ transforms given system by `Sol-Con-Sim` rule, i.e., we have $\langle x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}\{x \mapsto t\}; \mathfrak{d}\rangle$, where $\mathfrak{d}' = 1$. Let θ be a $(\mathcal{R}_{\text{sim}}, \lambda)$ solution of $x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}$ with degree $\mathfrak{d}$. It follows from the completeness of weak unification algorithm given in Sessa (2002), that $\sigma \preceq_{\mathcal{R}_{\text{sim}},\lambda} \theta$ is $(\mathcal{R}_{\text{sim}}, \lambda)$ solution of $x \simeq^?_{\mathcal{R}_{\text{sim}},\lambda} t \wedge \mathcal{K}\{x \mapsto t\}$ with degree $\mathfrak{d}$.

□

Theorem 5.6. *(Completeness theorem for* SolveSim*)*
Let $\mathcal{K}$ be a conjunction of primitive constraints and θ is a $(\mathcal{R}_{sim}, \lambda)$ solution of $\mathcal{K}$ with degree $\mathfrak{d}$. Then SolveSim *reduces $\langle \mathcal{K}; 1\rangle$ to $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$, such that there exists a $(\mathcal{R}_{sim}, \lambda)$ substitution $\sigma \preceq_{\mathcal{R}_{sim},\lambda} \theta$ that is a maximal solution of $\mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ with a degree $\mathfrak{d}_i \geq \mathfrak{d}$.*

Proof. By induction on the length of a rule application sequence leading from $\langle \mathcal{K}; 1\rangle$ to $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ using the completeness lemmas.

□

6. Proximity-Based Constraint Solving

In this section, we consider constraints built over proximity primitive constraints. The proximity constraint-solving algorithm, SolveProx, described below, operates on pairs consisting of a conjunction of primitive constraints and a real number from the interval $(0, 1]$. In particular, the rules have the form $\langle \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$, which defines the transformation $\langle \mathcal{K}; \mathfrak{d}\rangle \vee \mathcal{C} \rightsquigarrow_{\mathfrak{d}'} \langle \mathcal{K}_1; \mathfrak{d}'\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}'\rangle \vee \mathcal{C}$ where $\mathcal{C}$ has a form $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_m; \mathfrak{d}_m\rangle$. The rules are applied considering the associativity, commutativity, and idempotence of $\vee$ and $\wedge$, with false as the identity element for $\vee$. In the rules, the superscript ? indicates that the constraints are supposed to be solved.

Del-Var-Prox: **Variable Deletion, Proximity**

$\langle x \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} x \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle \mathcal{K}; \mathfrak{d}\rangle$, where $x \in \mathcal{V}$ and $\mathfrak{d}' = 1$.

Dec-Ord-Prox: **Ordered Decomposition, Proximity**

$$\langle f_{\mathsf{o}}(t_1, \ldots, t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{o}}(r_1, \ldots, r_n) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'}$$
$$\langle t_1 \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_1 \wedge \cdots \wedge t_n \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_n \wedge \mathcal{K}; \min(\mathfrak{d}, \mathfrak{d}')\rangle,$$

where $n \geq 0$, $\mathcal{R}_{\text{prox}}(f_{\mathsf{o}}, g_{\mathsf{o}}) \geq \lambda$ and $\mathfrak{d}' = \mathcal{R}_{\text{prox}}(f_{\mathsf{o}}, g_{\mathsf{o}})$.

Dec-Urd-Prox: **Unordered Decomposition, Proximity**

$$\langle f_{\mathsf{u}}(t_1, \ldots, t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{u}}(r_1, \ldots, r_m) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'}$$
$$\bigvee_{\substack{(t''_1,\ldots,t''_m)\in perm(t'_1,\ldots,t'_m) \\ \{t'_1,\ldots,t'_m\}\subseteq\{t_1,\ldots,t_n\}}} \langle t''_1 \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_1 \wedge \cdots \wedge$$
$$t''_m \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_m \wedge \mathcal{K}; \min(\mathfrak{d}, \mathfrak{d}')\rangle,$$

where $n \geq m \geq 0$, $\mathcal{R}_{\text{prox}}(f_{\mathsf{u}}, g_{\mathsf{u}}) \geq \lambda$ and $\mathfrak{d}' = \mathcal{R}_{\text{prox}}(f_{\mathsf{u}}, g_{\mathsf{u}})$.

Ori-UFS-Prox: **Unordered Function Symbol Orient**

$$\langle f_{\mathsf{u}}(t_1,\dots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{u}}(r_1,\dots,r_m) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'}$$
$$\langle g_{\mathsf{u}}(r_1,\dots,r_m) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} f_{\mathsf{u}}(t_1,\dots,t_n) \wedge \mathcal{K}; \mathfrak{d}\rangle,$$

where $n < m$ and $\mathfrak{d}' = 1$.

Ori-Var-Prox: **Variable Orient, Proximity**

$$\langle t \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} x \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle x \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} t \wedge \mathcal{K}; \mathfrak{d}\rangle,$$

where $t \notin \mathcal{V}$ and $\mathfrak{d}' = 1$.

Conf-FS-Prox: **Function Symbol Conflict, Proximity**

$$\langle f(t_1,\dots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g(r_1,\dots,r_m) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \mathsf{false},$$

where $n \geq m \geq 0$, $\mathfrak{d}' = 0$, and $f \in \mathcal{F}_{\mathsf{u}},\ g \in \mathcal{F}_{\mathsf{o}}$ or $f \in \mathcal{F}_{\mathsf{o}},\ g \in \mathcal{F}_{\mathsf{u}}$.

Conf-UFS-Prox: **Unordered Function Symbol Conflict, Proximity**

$$\langle f_{\mathsf{u}}(t_1,\dots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{u}}(r_1,\dots,r_m) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \mathsf{false},$$

where $n \geq m \geq 0$, $\mathfrak{d}' = 0$, and $\mathcal{R}_{\text{prox}}(f_{\mathsf{u}}, g_{\mathsf{u}}) < \lambda$.

Conf-OFS-Prox: **Ordered Function Symbol Conflict, Proximity**

$$\langle f_{\mathsf{o}}(t_1,\dots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{o}}(r_1,\dots,r_m) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \mathsf{false},$$

where $n \neq m$, and $\mathfrak{d}' = 0$ or $\mathcal{R}_{\text{prox}}(f_{\mathsf{o}}, g_{\mathsf{o}}) < \lambda$.

Check-Occ-Prox: **Occur Check, Proximity**

$$\langle x \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} t \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \mathsf{false}, \qquad \text{where } x \in \mathit{var}(t) \text{ and } x \neq t\ .$$

Sol-Con-Prox: **Solve Constraint, Proximity**

$$\langle x \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g(r_1,\dots,r_n) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow$$
$$\bigvee_{\mathcal{R}_{\text{prox}}(f,g)\geq\lambda} \langle x \simeq_{\mathcal{R},\lambda} f(y_1,\dots,y_n) \wedge f(y_1,\dots,y_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g(r_1,\dots,r_n)$$
$$\wedge\, \mathcal{K}\{x \mapsto f(y_1,\dots,y_n)\}; \min(\mathfrak{d}, \mathcal{R}_{\text{prox}}(f,g))\rangle,$$

where $\mathcal{R}_{\text{prox}}(f,g) \geq \lambda$, and $y_1,\dots,y_n$ are fresh variables.

In order to solve $t \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r$, we create an initial system $\langle t \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r; 1\rangle$ and apply successively SolveProx rules as is demonstrated in the following example:

Example 2. *Let* $\mathcal{C} = f_u(x, f_o(x,b), y, h_o(x,y)) \simeq_{\mathcal{R},0.4} g_u(g_o(a,y), b)$ *and let* $\mathcal{R}_{prox}$ *be defined as* $\mathcal{R}_{prox}(f_{\mathsf{u}}, g_{\mathsf{u}}) = 0.6$, $\mathcal{R}_{prox}(f_{\mathsf{o}}, g_{\mathsf{o}}) = 0.5$, $\mathcal{R}_{prox}(h_{\mathsf{o}}, g_{\mathsf{o}}) = 0.3$. *Then* SolveSim *performs the following derivation (the subscript* $\{\mathsf{Rule_1}, \dots, \mathsf{Rule_n}\}$ *of* $\rightsquigarrow$ *specifies which set of rules are applied in the transformation):*

$$\langle \mathcal{C} \rangle \rightsquigarrow_{\mathit{Dec\_Urd\_Prox}}$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge \underline{f_o(x,b) \simeq^?_{\mathcal{R},0.4} b}; 0.6 \right\rangle \vee$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} b \wedge f_o(x,b) \simeq^?_{\mathcal{R},0.4} g_o(a,y); 0.6 \right\rangle \vee$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \vee$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} b \wedge \underline{\underline{y \simeq^?_{\mathcal{R},0.4} g_o(a,y)}}; 0.6 \right\rangle \vee$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge \underline{h_o(x,y) \simeq^?_{\mathcal{R},0.4} b}; 0.6 \right\rangle \vee$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} b \wedge \underline{h_o(x,y) \simeq^?_{\mathcal{R},0.4} g_o(a,y)}; 0.6 \right\rangle \vee$$
$$\left\langle f_o(x,b) \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \vee$$
$$\left\langle f_o(x,b) \simeq^?_{\mathcal{R},0.4} b \wedge \underline{\underline{y \simeq^?_{\mathcal{R},0.4} g_o(a,y)}}; 0.6 \right\rangle \vee$$
$$\left\langle f_o(x,b) \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge \underline{h_o(x,y) \simeq^?_{\mathcal{R},0.4} b}; 0.6 \right\rangle \vee$$
$$\left\langle f_o(x,b) \simeq^?_{\mathcal{R},0.4} b \wedge \underline{h_o(x,y) \simeq^?_{\mathcal{R},0.4} g_o(a,y)}; 0.6 \right\rangle \vee$$
$$\left\langle y \simeq^?_{\mathcal{R},0.4} g_o(a,b) \wedge \underline{h_o(x,y) \simeq^?_{\mathcal{R},0.4} b}; 0.6 \right\rangle \vee$$
$$\left\langle y \simeq^?_{\mathcal{R},0.4} b \wedge \underline{h_o(x,y) \simeq^?_{\mathcal{R},0.4} g_o(a,b)}; 0.6 \right\rangle \rightsquigarrow^*_{\mathit{Check\_Occ\_Prox\ 2x,\ Conf\_OFS\_Prox\ 7x}}$$
$$\left\langle \underline{x \simeq^?_{\mathcal{R},0.4} b} \wedge f_o(x,b) \simeq^?_{\mathcal{R},0.4} g_o(a,y); 0.6 \right\rangle \vee$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \vee$$
$$\left\langle f_o(x,b) \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \rightsquigarrow_{\mathit{Sol\_Con\_Prox}}$$
$$\langle x \simeq_{\mathcal{R},0.4} b \wedge \underline{f_o(b,b) \simeq^?_{\mathcal{R},0.4} g_o(a,y)}; 0.6 \rangle \vee$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \vee$$
$$\left\langle f_o(x,b) \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \rightsquigarrow_{\mathit{Dec\_Ord\_Prox}}$$
$$\left\langle x \simeq_{\mathcal{R},0.4} b \wedge \underline{b \simeq^?_{\mathcal{R},0.4} a} \wedge b \simeq^?_{\mathcal{R},0.4} y; 0.5 \right\rangle \vee$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \vee$$
$$\left\langle f_o(x,b) \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \rightsquigarrow_{\mathit{Conf\_FS\_Prox}}$$
$$\left\langle x \simeq_{\mathcal{R},0.4} b \wedge b \simeq^?_{\mathcal{R},0.4} y; 0.5 \right\rangle \vee$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \vee$$
$$\left\langle \underline{f_o(x,b) \simeq^?_{\mathcal{R},0.4} g_o(a,y)} \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \rightsquigarrow_{\mathit{Dec\_Ord\_Prox}}$$
$$\left\langle x \simeq_{\mathcal{R},0.4} b \wedge b \simeq^?_{\mathcal{R},0.4} y; 0.5 \right\rangle \vee$$
$$\left\langle x \simeq^?_{\mathcal{R},0.4} g_o(a,y) \wedge y \simeq^?_{\mathcal{R},0.4} b; 0.6 \right\rangle \vee$$

$\langle x \simeq_{\mathcal{R},0.4} a \wedge b \simeq_{\mathcal{R},0.4} y \wedge \underline{y \simeq^?_{\mathcal{R},0.4} b}; 0.5\rangle \rightsquigarrow_{\mathit{Sol\_Con\_Prox}}$
$\big\langle x \simeq_{\mathcal{R},0.4} b \wedge b \simeq^?_{\mathcal{R},0.4} y; 0.5\big\rangle \vee$
$\langle x \simeq_{\mathcal{R},0.4} g_o(a, y) \wedge y \simeq_{\mathcal{R},0.4} b; 0.6\rangle \vee$
$\langle x \simeq_{\mathcal{R},0.4} a \wedge \underline{b \simeq_{\mathcal{R},0.4} b} \wedge y \simeq_{\mathcal{R},0.4} b; 0.5\rangle \rightsquigarrow_{\mathit{Del\_Var\_Prox}}$
$\langle x \simeq_{\mathcal{R},0.4} b \wedge \underline{b \simeq^?_{\mathcal{R},0.4} y}; 0.5\rangle \vee$
$\langle x \simeq_{\mathcal{R},0.4} g_o(a, y) \wedge y \simeq_{\mathcal{R},0.4} b; 0.6\rangle \vee$
$\langle x \simeq_{\mathcal{R},0.4} a \wedge y \simeq_{\mathcal{R},0.4} b; 0.5\rangle \rightsquigarrow^*_{\mathit{Ori\_Var\_Prox,\ Sol\_Con\_Prox}}$
$\langle x \simeq_{\mathcal{R},0.4} b \wedge y \simeq_{\mathcal{R},0.4} b; 0.5\rangle \vee$
$\langle x \simeq_{\mathcal{R},0.4} g_o(a, y) \wedge y \simeq_{\mathcal{R},0.4} b; 0.6\rangle \vee$
$\langle x \simeq_{\mathcal{R},0.4} a \wedge y \simeq_{\mathcal{R},0.4} b; 0.5\rangle$

Example 3. *We note that solving a primitive constraint $t \simeq_{\mathcal{R},\lambda} r$, when $\mathcal{R}$ is a proximity relation, can be more complex and may yield more solutions than when $\mathcal{R}$ is a similarity relation. Let us define $\mathcal{R}$ as follows:*

$$\mathcal{R}(f_\circ, g_\circ) = 0.9, \quad \mathcal{R}(h_\circ, p_\circ) = 0.7, \quad \mathcal{R}(a, b) = 0.6.$$

Now consider the following similarity constraint:

$$f_\circ(x, h_\circ(a)) \simeq^?_{\mathcal{R},\,0.6} g_\circ(h_\circ(a), p_\circ(b)).$$

Our SolveSim *algorithm solves this as follows:*

$\langle f_\circ(x, h_\circ(a)) \simeq^?_{\mathcal{R},0.6} g_\circ(h_\circ(a), p_\circ(b)); 1\rangle \rightsquigarrow_{\mathit{(Dec\_Ord\_Sim)}}$
$\langle x \simeq^?_{\mathcal{R},0.6} h_\circ(a) \wedge \underline{h_\circ(y) \simeq^?_{\mathcal{R},0.6} p_\circ(b)}; 0.9\rangle \rightsquigarrow_{\mathit{(Dec\_Ord\_Sim)}}$
$\langle \underline{x \simeq^?_{\mathcal{R},0.6} h_\circ(a)} \wedge \underline{y \simeq^?_{\mathcal{R},0.6} b; 0.7}\rangle \rightsquigarrow^*_{\mathit{(Sol\_Con\_Sim)}}$
$\langle x \simeq_{\mathcal{R},0.6} h_\circ(a) \wedge y \simeq_{\mathcal{R},0.6} b; 0.7\rangle$

Next, we consider the same constraint:

$$f_\circ(x, h_\circ(a)) \simeq^?_{\mathcal{R},\,0.6} g_\circ(h_\circ(a), p_\circ(b)),$$

where $\mathcal{R}$ is treated as a proximity relation. The SolveProx *algorithm processes it and produces:*

$\langle \underline{f_\circ(x, h_\circ(a)) \simeq^?_{\mathcal{R},0.6} g_\circ(h_\circ(a), p_\circ(b))}; 1\rangle \rightsquigarrow_{\mathit{(Dec\_Ord\_Prox)}}$

$\langle x \simeq^?_{\mathcal{R},1} h_\circ(a) \wedge \underline{h_\circ(y) \simeq^?_{\mathcal{R},0.6} p_\circ(b)}; 0.9\rangle \rightsquigarrow_{\mathit{(Dec\_Ord\_Prox)}}$

$\langle \underline{x \simeq^?_{\mathcal{R},0.6} h_\circ(a)} \wedge y \simeq^?_{\mathcal{R},0.6} b; 0.7\rangle \rightsquigarrow_{\mathit{(Sol\_Con\_Prox)}}$

$\langle x \simeq_{\mathcal{R},0.6} p_\circ(a) \wedge \underline{p_\circ(x_1) \simeq^?_{\mathcal{R},0.6} h_\circ(a)} \wedge y \simeq^?_{\mathcal{R},0.6} b; 0.7\rangle \vee$

$\langle x \simeq_{\mathcal{R},0.6} h_\circ(a) \wedge \underline{h_\circ(x_1) \simeq^?_{\mathcal{R},0.6} h_\circ(a)} \wedge y \simeq^?_{\mathcal{R},0.6} b; 0.7\rangle \rightsquigarrow^*_{\mathit{(Dec\_Ord\_Prox)}}$

$\langle x \simeq_{\mathcal{R},0.6} p_\circ(a) \wedge \underline{x_1 \simeq^?_{\mathcal{R},0.6} a} \wedge y \simeq^?_{\mathcal{R},0.6} b; 0.7\rangle \vee$

$\langle x \simeq_{\mathcal{R},0.6} h_\circ(a) \wedge \underline{x_1 \simeq^?_{\mathcal{R},0.6} a} \wedge y \simeq^?_{\mathcal{R},0.6} b; 0.7\rangle \rightsquigarrow^*_{\mathit{(Sol\_Con\_Prox)}}$

$\langle x \simeq_{\mathcal{R},0.6} p_\circ(a) \wedge x_1 \simeq_{\mathcal{R},0.6} b \wedge a \simeq^?_{\mathcal{R},0.6} b \wedge \underline{y \simeq^?_{\mathcal{R},0.6} b}; 0.7\rangle \vee$

$\langle x \simeq_{\mathcal{R},0.6} h_\circ(a) \wedge x_1 \simeq_{\mathcal{R},0.6} b \wedge b \simeq^?_{\mathcal{R},0.6} a \wedge \underline{y \simeq^?_{\mathcal{R},0.6} b}; 0.7\rangle \rightsquigarrow^*_{\mathit{(Sol\_Con\_Prox)}}$

$\langle x \simeq_{\mathcal{R},0.6} p_\circ(a) \wedge x_1 \simeq_{\mathcal{R},0.6} a \wedge \underline{a \simeq^?_{\mathcal{R},0.6} b} \wedge y \simeq_{\mathcal{R},0.6} a \wedge \underline{a \simeq^?_{\mathcal{R},0.6} b}; 0.7\rangle \vee$

$\langle x \simeq_{\mathcal{R},0.6} p_\circ(a) \wedge x_1 \simeq_{\mathcal{R},0.6} b \wedge \underline{b \simeq^?_{\mathcal{R},0.6} b} \wedge y \simeq_{\mathcal{R},0.6} b \wedge \underline{b \simeq^?_{\mathcal{R},0.6} b}; 0.7\rangle \vee$

$\langle x \simeq_{\mathcal{R},0.6} h_\circ(a) \wedge x_1 \simeq_{\mathcal{R},0.6} a \wedge \underline{a \simeq^?_{\mathcal{R},0.6} a} \wedge y \simeq_{\mathcal{R},0.6} a \wedge \underline{a \simeq^?_{\mathcal{R},0.6} b}; 0.7\rangle \vee$

$\langle x \simeq_{\mathcal{R},0.6} h_\circ(a) \wedge x_1 \simeq_{\mathcal{R},0.6} b \wedge$

$\underline{b \simeq^?_{\mathcal{R},0.6} a} \wedge y \simeq_{\mathcal{R},0.6} b \wedge \underline{b \simeq^?_{\mathcal{R},0.6} b}; 0.7\rangle \rightsquigarrow^*_{\mathit{(Dec\_Ord\_Prox)}}$

$\langle x \simeq_{\mathcal{R},0.6} p_\circ(a) \wedge x_1 \simeq_{\mathcal{R},0.6} a \wedge y \simeq_{\mathcal{R},0.6} a; 0.7\rangle \vee$

$\langle x \simeq_{\mathcal{R},0.6} p_\circ(a) \wedge x_1 \simeq_{\mathcal{R},0.6} a \wedge y \simeq_{\mathcal{R},0.6} b; 0.7\rangle \vee$

$\langle x \simeq_{\mathcal{R},0.6} h_\circ(a) \wedge x_1 \simeq_{\mathcal{R},0.6} a \wedge y \simeq_{\mathcal{R},0.6} a; 0.7\rangle \vee$

$\langle x \simeq_{\mathcal{R},0.6} h_\circ(a) \wedge x_1 \simeq_{\mathcal{R},0.6} a \wedge y \simeq_{\mathcal{R},0.6} b; 0.7\rangle$

From this example, we observe that solving the primitive constraint using the SolveSim *algorithm returns a single solution, whereas solving the same constraint using the* SolveProx *algorithm yields four solutions.*

Below we study the properties of the proximity-based constraint solving algorithm.

Theorem 6.1. *(Termination)*
The algorithm SolveProx *terminates on any input constraint.*

Proof. We define a <u>complexity measure</u> for a system $cm(\mathcal{K}_1; \mathfrak{d}_1 \vee \cdots \vee \mathcal{K}_n; \mathfrak{d}_n)$ where $\mathcal{K}_i$ is a conjunction of primitive constraints and $\mathfrak{d}_i$ is a degree, and show that $cm(\mathcal{K}_1; \mathfrak{d}_1 \vee \cdots \vee \mathcal{K}_n; \mathfrak{d}_n) < cm(\mathcal{K}'_1; \mathfrak{d}'_1 \vee \cdots \vee \mathcal{K}'_m; \mathfrak{d}'_m)$ holds whenever $\langle \mathcal{K}_1; \mathfrak{d}_1 \vee \cdots \vee \mathcal{K}_n; \mathfrak{d}_n \rangle \rightsquigarrow \langle \mathcal{K}'_1; \mathfrak{d}'_1 \vee \cdots \vee \mathcal{K}'_m; \mathfrak{d}'_m \rangle$.

The complexity measure $cm(\mathcal{K})$ of a conjunction of primitive constraints $\mathcal{K}$ is the tuple $\langle N_1, N_2, N_3, N_4 \rangle$ defined as follows ($\{\!|\,|\!\}$ stands for a multiset):

- Let $\mathrm{depth}_x(t) = \max\{d \mid \text{there exists a path of length } d \text{ from the root of } t \text{ to an occurrence of } x\}$.

  $$\mathrm{depth}_x(\mathcal{C}) = \max\{\, \mathrm{depth}_x(t) \mid t \text{ is a term in } \mathcal{C} \text{ and } x \text{ occurs in } t \,\}.$$

  We also define the activity of a variable as follows:

  $$\mathrm{active}(x) = \begin{cases} 1, & \text{if } x \text{ does not occur in solved form;} \\ 0, & \text{otherwise.} \end{cases}$$

  The multiset N_1 is then defined as:

  $$N_1 := \{\!|\, (-\mathrm{depth}_x(C), \mathrm{active}(x)) \mid x \in \mathcal{C} \,|\!\}$$

  where $\mathcal{C}$ is the set of constraints.

- $N_2 := \{\!|\, \mathit{size}(t) + \mathit{size}(r) \mid t \simeq^?_{\mathcal{R}_{\mathrm{prox}},\lambda} r \in \mathcal{K} |\!\}$, where *size* of a term denotes the denotational length of t.

- N_3 the number of equations in the form t $\simeq^?_{\mathcal{R},\lambda} x$.

- N_4 the number of equations in the form $f_{\mathsf{u}}(t_1, \ldots, t_n) \simeq^?_{\mathcal{R},\lambda} g_{\mathsf{u}}(r_1, \ldots, r_m)$ where $n < m$

The complexity measure $cm(\mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n)$ is defined as the multiset $\{\!|\, cm(\mathcal{K}_1), \ldots, cm(\mathcal{K}_n) |\!\}$. Measures are compared by multiset extension of the lexicographic ordering on tuples. The table below shows which rule reduces which component of the measure, which implies termination of the algorithm solve.

□

Definition 6.1. *A conjunction of proximity primitive constraints $\mathcal{K}$ is in partially solved form, if $\mathcal{K}$ is either* true *or each primitive constraint in $\mathcal{K}$ has a form $x \simeq_{\mathcal{R},\lambda} t$ or $y \simeq^?_{\mathcal{R},\lambda} z$, where x appears only once in $\mathcal{K}$. A constraint in DNF $\mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ is in partially solved form if each $\mathcal{K}_i$ is in partially solved form.*

Rule	$\mathbf{N_1}$	$\mathbf{N_2}$	$\mathbf{N_3}$	$\mathbf{N_4}$
Sol-Con-Prox	$>$			
Del-Var-Prox,Dec-OFS-Prox,Dec-UFS-Prox,Conf-FS-Prox,Conf-UFS-Prox,Conf-OFS-Prox,Check-Occ-Prox	$\geq$	$>$		
Ori-Var	$\geq$	$\geq$	$>$	
Ori-UFS	$\geq$	$\geq$	$\geq$	$>$

Theorem 6.2. *If* SolveProx *reduces* $\langle \mathcal{K}; 1\rangle \rightsquigarrow^* \langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ *then each* $\mathcal{K}_j$ *is either in partially solved form or is* false*.*

Proof. By Theorem 6.1, each $\mathcal{K}_j$ is in a normal form. Assume by contradiction that $\mathcal{K}_j$ is not in partially solved form. By inspection of the solver rules, based on the definition of partially solved constraints, we can see that there is a rule that applies to $\mathcal{K}_j$. But this contradicts the fact that $\mathcal{K}_j$ is in a normal form. Hence, $\mathcal{K}_j$ is partially solved form.

□

Lemma 6.3. *(Soundness lemma for* SolveProx*)*
Let $\mathcal{R}$ *be a similarity relation,* λ *a cut value and* $\langle \mathcal{K}; \mathfrak{d}\rangle$ *a system. If* $\langle \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ *is a step performed by the solver* SolveProx *and* σ *is a maximal solution of* $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ *with degree* $\mathfrak{d}''$*, then* σ *is a solution of* $\mathcal{K}$ *with degree* $\min(\mathfrak{d}'', \mathfrak{d}')$*.*

Proof. By case distinction on the inference rules of the solver. We illustrate here three cases, when the selected rules are `Dec-OFS-Prox`, `Dec-UFS-Prox` and `Sol-Con-Prox`. For other rules the lemma can be shown in a similar manner.

- For failing rules it is trivial as false has no solution.
- Assume SolveProx transforms given system by `Dec-OFS-Prox` rule, i.e., we have $\langle f_{\mathsf{o}}(t_1, \dots, t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{o}}(r_1, \dots, r_n) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle t_1 \simeq^?_{\mathcal{R},\lambda} r_1 \wedge \cdots \wedge t_n \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_n \wedge \mathcal{K}; \min(\mathfrak{d}, \mathfrak{d}')\rangle$

  Then $\mathfrak{d}' = \mathcal{R}_{\text{prox}}(f_{\mathsf{o}}, g_{\mathsf{o}}) \geq \lambda$. Let σ be a $(\mathcal{R}, \lambda)$ solution of $t_1 \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_1 \wedge \cdots \wedge t_n \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_n \wedge \mathcal{K}$ with degree $\mathfrak{d}''$ and $S = \{e \mid$ e is a conjunct of

$\mathcal{K}\}$. This means $\min(\bigcup_{i=1}^{n} \mathcal{R}_{\text{prox}}(t_i\sigma, r_i\sigma) \cup \bigcup_{(t \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r)\in S} \mathcal{R}_{\text{prox}}(t\sigma, r\sigma))$ $= \mathfrak{d}'' \geq \lambda$. Since, $\mathfrak{d}' \geq \lambda$ and $\mathfrak{d}'' \geq \lambda$ we have $\min(\mathfrak{d}'', \mathfrak{d}') \geq \lambda$. Moreover,

$$\begin{aligned}
\min(\mathfrak{d}'', \mathfrak{d}') = \quad & \min\left(\bigcup_{i=1}^{n} \mathcal{R}_{\text{prox}}(t_i\sigma, r_i\sigma) \cup \right. \\
& \left. \bigcup_{(t \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r)\in S} \mathcal{R}_{\text{prox}}(t\sigma, r\sigma), \mathfrak{d}' \right) \\
= \quad & \min\left(\bigcup_{i=1}^{n} \mathcal{R}_{\text{prox}}(t_i\sigma, r_i\sigma) \cup \right. \\
& \left. \bigcup_{(t \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r)\in S} \mathcal{R}_{\text{prox}}(t\sigma, r\sigma), \mathcal{R}_{\text{prox}}(f_{\mathsf{o}}, g_{\mathsf{o}}) \right) \\
= \quad & \min\left(\{\mathcal{R}_{\text{prox}}(f_{\mathsf{o}}(t_1, \ldots, t_n)\sigma, \right. \\
& g_{\mathsf{o}}(r_1, \ldots, r_m)\sigma)\} \cup \\
& \left. \bigcup_{(t \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r)\in S} \mathcal{R}_{\text{prox}}(t\sigma, r\sigma) \right)
\end{aligned}$$

which implies that σ is a solution of
$f_{\mathsf{o}}(t_1, \ldots, t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{o}}(r_1, \ldots, r_n) \wedge \mathcal{K}$ with the degree $\min(\mathfrak{d}'', \mathfrak{d}')$

- Assume $\mathsf{SolveProx}$ transforms given system by `Dec-UFS-Prox` rule, i.e., we have

$$\begin{aligned}
& \langle f_{\mathsf{u}}(t_1, \ldots, t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{u}}(r_1, \ldots, r_n) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \\
& \bigvee_{\substack{(t''_1,\ldots,t''_m)\in perm(t'_1,\ldots,t'_m) \\ \{t'_1,\ldots,t'_m\}\subseteq\{t_1,\ldots,t_n\}}} \langle t''_1 \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_1 \wedge \cdots \wedge t''_m \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_m \wedge \mathcal{K}; \min(\mathfrak{d}, \mathfrak{d}')\rangle
\end{aligned}$$

Then $\mathfrak{d}' = \mathcal{R}_{\text{prox}}(f_{\mathsf{u}}, g_{\mathsf{u}}) \geq \lambda$. Let σ be a $(\mathcal{R}, \lambda)$ maximal solution of $\bigvee_{\substack{(t''_1,\ldots,t''_m)\in perm(t'_1,\ldots,t'_m) \\ \{t'_1,\ldots,t'_m\}\subseteq\{t_1,\ldots,t_n\}}} \langle t''_1 \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_1 \wedge \cdots \wedge t''_m \simeq^?_{\mathcal{R},\lambda} r_m \wedge \mathcal{K}\rangle$ with degree $\mathfrak{d}''$. According to the Definition 4.2, σ is a $(\mathcal{R},\lambda)$ solution of a conjuct $t''_1 \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_1 \wedge \cdots \wedge t''_m \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_m \wedge \mathcal{K}$ with degree $\mathfrak{d}''$. This means there is no other conjuct for which σ solves it with a strictly greater degree. Therefore, according to the Definition of $\mathcal{R}_{\text{prox}}$ for terms where top symbols are unordered fuction symbols, we conclude that σ is $(\mathcal{R}_{\text{prox}},\lambda)$ solution of $f_{\mathsf{u}}(t_1,\ldots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{u}}(r_1,\ldots,r_n) \wedge \mathcal{K}$ with degree $\min(\mathfrak{d}'',\mathfrak{d}')$.

- Assume $\mathsf{SolveProx}$ transforms given system by `Sol-Con-Prox` rule, i.e., we have $\langle x \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g(r_1,\ldots,r_n) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \bigvee_{\mathcal{R}_{\text{prox}}(f,g)\geq\lambda} \langle x \simeq_{\mathcal{R}_{\text{prox}},\lambda} f(y_1,\ldots,y_n) \wedge f(y_1,\ldots,y_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g(r_1,\ldots,r_n) \wedge \mathcal{K}\{x \mapsto f(y_1,\ldots,y_n)\}; \min(\mathfrak{d}, \mathcal{R}_{\text{prox}}(f,g))\rangle$ where $\mathcal{R}_{\text{prox}}(f,g) \geq \lambda$, and $y_1,\ldots,y_n$ are fresh variables. $\mathfrak{d}' = 1$. let denoted by $\deg(\mathcal{K}\sigma)$ the approximation degree of the constraint solving problem of $\mathcal{K}$ by σ. let σ be a $(\mathcal{R}_{\text{prox}},\lambda)$ solution with degree of $\mathfrak{d}''$ of the right part. $\mathfrak{d}'' = \min(\deg(\mathcal{K}\sigma), \mathcal{R}_{\text{prox}}(g(r_1,\ldots, r_n)\sigma, f(y_1,\ldots,y_n)\sigma)))$. Since $\mathfrak{d}'' \geq \lambda$ and $\mathfrak{d}' \geq \lambda$ we have $\min(\mathfrak{d}',\mathfrak{d}'') \geq \lambda$. We conclude that σ solves the left part with degree of $\min(\mathfrak{d}',\mathfrak{d}'') = \min(1,\mathfrak{d}'') = \mathfrak{d}''$

□

Theorem 6.4. *(Soundness theorem for* $\mathsf{SolveProx}$*)*
If $\mathsf{SolveSim}$ *reduces* $\langle \mathcal{K}; 1\rangle \rightsquigarrow^* \langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ *where each* $\mathcal{K}_i$ *is in partially solved form and* σ *is a maximal solution of* $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ *with degree* $\mathfrak{d}'$ *then* σ *is a solution* $\mathcal{K}$ *with degree* $\min(\mathfrak{d}',\mathfrak{d}_i)$*.*

Proof. By induction on the length of a rule application sequence leading from $\langle \mathcal{K}; 1\rangle$ to $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ using the soundness lemmas. □

Lemma 6.5. *(Completness Lemma for* $\mathsf{SolveProx}$*) Let* $\mathcal{K}_0$ *be a conjunction of primitive constraints and* θ *is a* $(\mathcal{R},\lambda)$ *solution of* $\mathcal{K}_0$ *with degree* $\mathfrak{d}$*. Then we can make a step* $\langle \mathcal{K}_0; \mathfrak{d}_0\rangle \rightsquigarrow_{\mathfrak{d}'} \langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \cdots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$*, such that there exists a* $(\mathcal{R}_{prox},\lambda)$ *substitution* $\sigma \preceq_{\mathcal{R}_{prox},\lambda} \theta$ *that is a maximal solution of* $\mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ *with a degree* $\mathfrak{d}''$ *such that* $\mathfrak{d} = \min(\mathfrak{d}'',\mathfrak{d}')$*.*

Proof. By case distinction on the inference rules of the solver. We illustrate here three cases, when the selected rules are Dec-OFS-Prox, Dec-UFS-Prox and Sol-Con-Prox. For other rules the lemma can be shown in a similar manner.

- For failing rules it is trivial as false has no solution.

- Assume SolveSim transforms given system by `Dec-OFS-Prox` rule, i.e., we have $\langle f_{\mathsf{o}}(t_1,\dots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{o}}(r_1,\dots,r_n) \wedge \mathcal{K}; \mathfrak{d}_1\rangle \rightsquigarrow_{\mathfrak{d}'} \langle t_1 \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_1 \wedge \dots \wedge t_n \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_n \wedge \mathcal{K}; \min(\mathfrak{d}_1,\mathfrak{d}')\rangle$. Then $\mathfrak{d}' = \mathcal{R}_{\text{prox}}(f_{\mathsf{o}}, g_{\mathsf{o}})$. Let θ be a $(\mathcal{R}_{\text{prox}},\lambda)$ solution of $f_{\mathsf{o}}(t_1,\dots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{o}}(r_1,\dots,r_n) \wedge \mathcal{K}$ with degree $\mathfrak{d}$. Since $\mathcal{K}$ does not change during transformation, we can assume θ is a $(\mathcal{R}_{\text{prox}},\lambda)$ solution of $f_{\mathsf{o}}(t_1,\dots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{o}}(r_1,\dots,r_n)$ That means $\mathcal{R}_{\text{prox}}(f_{\mathsf{o}}(t_1\theta,\dots,t_n\theta), g_{\mathsf{o}}(r_1\theta,\dots,r_n\theta)) = \mathfrak{d} \geq \lambda$. Therefore, according to the definition of $\mathcal{R}_{\text{prox}}$ we have $\mathcal{R}_{\text{prox}}(f_{\mathsf{o}}(t_1\theta,\dots,t_n\theta), g_{\mathsf{o}}(r_1\theta,\dots,r_n\theta)) = \min(\mathcal{R}_{\text{prox}}(f_{\mathsf{o}}, g_{\mathsf{o}}), \mathcal{R}_{\text{prox}}(t_1\theta, r_1\theta),\dots,\mathcal{R}_{\text{prox}}(t_n\theta, r_n\theta))$. This means θ also solves $t_1 \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_1,\dots,t_n \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_n$ with a degree $\mathfrak{d}''$, such that $\mathfrak{d}''$ satisfies the equality $\mathfrak{d} = \min(\mathfrak{d}'',\mathfrak{d}')$ and we conclude this case.

- Assume SolveSim transforms given system by `Dec-UFS-Prox` rule, i.e., we have $\langle f_{\mathsf{u}}(t_1,\dots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{u}}(r_1,\dots,r_n) \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'}$ $\bigvee_{\substack{(t''_1,\dots,t''_m)\in perm(t'_1,\dots,t'_m)\\ \{t'_1,\dots,t'_m\}\subseteq\{t_1,\dots,t_n\}}} \langle t''_1 \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_1 \wedge\dots\wedge t''_m \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_m \wedge \mathcal{K}; \min(\mathfrak{d},\mathfrak{d}')\rangle$. Then $\mathfrak{d}' = \mathcal{R}_{\text{prox}}(f_{\mathsf{u}}, g_{\mathsf{u}})$. Let θ is a $(\mathcal{R},\lambda)$ solution of $f_{\mathsf{u}}(t_1,\dots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{u}}(r_1,\dots,r_m) \wedge \mathcal{K}$ with degree $\mathfrak{d}$. Since $\mathcal{K}$ does not change during transformation, we can assume θ is a $(\mathcal{R}_{\text{prox}},\lambda)$ solution of $f_{\mathsf{u}}(t_1,\dots,t_n) \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} g_{\mathsf{u}}(r_1,\dots,r_m)$. That means $\mathcal{R}_{\text{prox}}(f_{\mathsf{u}}(t_1\theta,\dots,t_n\theta)$ $, g_{\mathsf{u}}(r_1\theta,\dots,r_m\theta)) = \mathfrak{d} \geq \lambda$. Therefore, according to the definition of $\mathcal{R}_{\text{prox}}$ we have
$\mathcal{R}_{\text{prox}}(f_{\mathsf{u}}(t_1\theta,\dots,t_n\theta), g_{\mathsf{u}}(r_1\theta,\dots,r_m\theta)) =$
$\max_{\bar{t}\in perm(t'_1,\dots,t'_m)\{t'_1,\dots,t'_m\}\subseteq\{t_1,\dots,t_n\}} \min\big(\mathcal{R}_{\text{prox}}(f_{\mathsf{u}}, g_{\mathsf{u}}), \mathcal{R}_{\text{prox}}\big(f_{\mathsf{o}}(\bar{t}\theta),$ $f_{\mathsf{o}}(r_1\theta,\dots,r_m\theta)\big)\big)$. This means, that θ is a $(\mathcal{R}_{\text{prox}},\lambda)$ maximal solution of
$\bigvee_{\substack{(t''_1,\dots,t''_m)\in perm(t'_1,\dots,t'_m)\\ \{t'_1,\dots,t'_m\}\subseteq\{t_1,\dots,t_n\}}} \langle t''_1 \simeq^?_{\mathcal{R},\lambda} r_1 \wedge \dots \wedge t''_m \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} r_m$ with degree $\mathfrak{d}''$, where $\bar{t} = (t''_1,\dots,t''_m)$. We know, that $\mathfrak{d}' = \mathcal{R}_{\text{prox}}(f_{\mathsf{u}}, g_{\mathsf{u}})$ and therefore we conclude, that $\mathfrak{d}''$ satisfies the equality $\mathfrak{d} = \min(\mathfrak{d}'',\mathfrak{d}')$,

- Assume SolveProx transforms given system by `Sol-Con-Prox` rule, i.e., we have $\langle x \simeq^?_{\mathcal{R},\lambda} t \wedge \mathcal{K}; \mathfrak{d}\rangle \rightsquigarrow_{\mathfrak{d}'} \langle x \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} t \wedge \mathcal{K}\{x \mapsto t\}; \mathfrak{d}\rangle$, where $\mathfrak{d}' = 1$. Let θ be a $(\mathcal{R}_{\text{prox}},\lambda)$ solution of $x \simeq^?_{\mathcal{R}_{\text{prox}},\lambda} t \wedge \mathcal{K}$ with degree $\mathfrak{d}''$. x $\theta = f(t_1,\dots,t_n)$, where $\mathcal{R}(f,g) \geq \lambda$ and $f(t_1,\dots,t_n) \simeq_{\mathcal{R}_{\text{prox}},\lambda} g(r_1,\dots,r_n)$ under θ. the substitution for the right part with degree of

$\mathfrak{d}''$ is $\sigma := \theta \cup \{y_1 \mapsto t_1, \dots, y_n \mapsto t_n\}$. Then x$\sigma = f(t_1, \dots, t_n) = x\theta$, $f(y_1, \dots, y_n)\sigma = f(t_1, \dots, t_n) \simeq_{\mathcal{R}_{\text{prox}},\lambda} g(r_1, \dots, r_n)\sigma$ and the rest of the constraint $\mathcal{K}\{x \mapsto f(y_1, \dots, y_n)\}\sigma = \mathcal{K}\theta$, since x$\theta = f(t_1, \dots, t_n)$. We conclude $\sigma \preceq_{\mathcal{R}_{\text{prox}},\lambda} \theta$ with degree of $\min(\mathfrak{d}'', \mathfrak{d}') = \mathfrak{d}''$, since $\mathfrak{d}' = 1$.

□

Theorem 6.6. *(Completeness theorem for* SolveProx*)*
Let $\mathcal{K}$ be a conjunction of primitive constraints and θ is a $(\mathcal{R}_{prox}, \lambda)$ solution of $\mathcal{K}$ with degree $\mathfrak{d}$. Then SolveProx *reduces $\langle \mathcal{K}; 1\rangle$ to $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \dots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$, such that there exists a $(\mathcal{R}_{prox}, \lambda)$ substitution $\sigma \preceq_{\mathcal{R}_{prox},\lambda} \theta$ that is a maximal solution of $\mathcal{K}_1 \vee \dots \vee \mathcal{K}_n$ with a degree $\mathfrak{d}_i \geq \mathfrak{d}$.*

Proof. By induction on the length of a rule application sequence leading from $\langle \mathcal{K}; 1\rangle$ to $\langle \mathcal{K}_1; \mathfrak{d}_1\rangle \vee \dots \vee \langle \mathcal{K}_n; \mathfrak{d}_n\rangle$ using the completeness lemmas.

□

7. Conclusion

In this paper, we advanced the theory and practice of constraint solving by developing novel algorithms for equational reasoning under fuzzy similarity and proximity relations modulo commutativity. Our framework generalizes classical unification by introducing graded, approximate notions of equality, thereby enabling more flexible and expressive forms of symbolic reasoning in domains where strict syntactic equality is too rigid or inadequate.

We presented two distinct algorithms. The first addresses similarity-based constraint solving modulo commutativity, allowing controlled mismatches between function symbol names. The second, more general algorithm, supports constraint solving under fuzzy proximity relations and accommodates mismatches in both symbol names and function arities. This additional flexibility significantly broadens the applicability of our approach, particularly in contexts involving heterogeneity, noise, or structural divergence.

To ensure the soundness and practical reliability of our methods, we formally proved the key properties of both algorithms, including their termination, soundness, and completeness. These results establish a solid theoretical foundation and guarantee that the algorithms behave predictably across a wide range of input problems. In addition, we implemented both algorithms in a user-friendly, web-based interfaceDundua et al. (2025a), designed to facilitate experimentation, evaluation, and deployment in practical settings.

8. Discussion and Future Work

XML has become the de facto standard for information representation and data exchange on the Web, enabling platform-independent and structured communication between systems. Due to its inherently hierarchical nature, XML documents are most naturally represented and processed using tree-based data structures, which facilitate efficient navigation, querying, and transformation of data Hosoya and Pierce (2003); Coelho et al. (2010); Bry et al. (2004). XML is widely used for data exchange in web services, storing configuration files, and representing structured data. It promotes interoperability between systems and supports standards such as RSS, SVG, and SOAP. Additionally, XML is used in publishing to separate content from presentation, and in databases for data import and export.

The ability to accurately measure proximity between XML documents is crucial in a wide range of applications, such as data integration, document versioning, digital libraries, and information retrieval Oliveira et al. (2018). XML documents are inherently semi-structured, and their schema and content often evolve over time due to system updates, heterogeneous tool usage, or inconsistent modeling practices. Consequently, proximity assessment must account for both structural (schema-level) and content-level differences to ensure robust and meaningful comparisons.

Unordered XML is particularly important because it accurately models data-centric applications where the order of elements is irrelevant—such as bibliographic databases, metadata repositories, and configuration files Boneva et al. (2013, 2015). The significance of this structural model has also motivated research on efficient XML compression, demonstrating that omitting order can enable significantly more compact representations of XML trees Lohrey et al. (2017). Traditional ordered XML schemas often result in overly complex or permissive definitions. In contrast, unordered XML allows simpler and more precise schema design through formalisms such as disjunctive multiplicity schemas (DMS) and multiplicity schemas (MS), enabling efficient validation and static analysis. This approach is particularly valuable in data integration, web services, and query optimization, where flexibility and semantic precision are essential.

Beyond schema design, expressive query languages have been developed specifically for unordered data trees. Notably, Active XML (AXML) and its generalizations support powerful data tree transformations by embedding computation directly within the data Abiteboul et al. (2012). These languages combine basic tree pattern queries with both deterministic and nondeterministic control constructs and are capable of Turing-complete computation. However, their expressive power remains limited when dealing with structural mismatches between XML documents—an aspect that can be naturally addressed by our proposed framework.

In the context of aligning protein-protein interaction (PPI) networks, Koyutürk et al. (2006) propose a biologically motivated framework that evaluates network proximity by extending the notions of match, mismatch, and duplication—originally used in sequence alignment—to the graph level. Their scoring function is flexible and evolution-aware, enabling the identification of conserved and divergent modular structures in PPI networks. However, the proximity function used in this framework is crisp and inherently limited in expressing approximate structural relationships that are common in biological systems.

We propose that our commutative unification algorithm with fuzzy proximity relations provides a natural and expressive extension to such frameworks. Specifically, the use of fuzzy proximity allows the model to account for partial or approximate matches between interaction substructures, which often arise due to noise, incomplete data, or evolutionary divergence. The commutative property ensures that the proximity score between any two substructures is symmetric, maintaining consistency throughout the alignment process. By integrating fuzzy semantics, our approach can enhance the interpretability and biological relevance of alignments by quantifying proximity on a graded scale, rather than relying on strict unification or predefined penalties.

Furthermore, we suggest that our framework can complement and extend recent initiatives such as SURF Nippa et al. (2025), which standardizes the representation of chemical reaction data in a tabular, machine-readable format to promote FAIR data principles. While SURF facilitates structured and interoperable documentation, our approach can support more flexible querying, comparison, and integration of reaction data in cases where approximate matching of components or conditions is required. In particular, the combination of fuzzy proximity with commutative reasoning enables more robust identification of reaction patterns across datasets that differ slightly in structure or nomenclature, thereby enhancing data mining and machine learning workflows in chemical synthesis.

These potential applications motivate us to further extend our framework to support function symbols modulo not only commutativity but also associativity. Incorporating both properties would significantly enhance the expressiveness and flexibility of the framework, enabling it to more accurately capture a broader spectrum of real-world structures and computations.

It is well established that solving constraints modulo commutativity and associativity-commutativity (AC) is NP-complete Benanav et al. (1987). In Eker (1995), an algorithm for AC unification was proposed, which constructs a hierarchy of bipartite graph matching problems to systematically explore the solution space. This approach introduces the notion of semi-pure AC systems, which significantly reduces the search space through structural refinements and achieves improved performance on non-pathological instances. We consider this framework a natural starting point for extending constraint solving to incorporate fuzzy proximity relations under commutative or AC theories. As part of our future work, we plan to build on this foundation to develop constraint-solving algorithms for fuzzy unification in AC settings.

Furthermore, drawing on foundational work such as Jaffar and Maher (1994), we plan to integrate the extended solver into the Constraint Logic Programming (CLP) framework. In doing so, we aim to introduce a novel extension that combines proximity- and similarity-based reasoning with constraint solving modulo associativity and commutativity. This enriched framework is particularly well-suited for domains where exact constraint satisfaction is too rigid, and where graded, proximity-aware solutions are more appropriate—especially in the contexts discussed above, including XML document transformation, biological network alignment, and chemical reaction data integration.

References

Abiteboul, S., Bourhis, P., Vianu, V., 2012. Highly expressive query languages for unordered data trees, in: Proceedings of the 15th International Conference on Database Theory, pp. 46–60.

Aït-Kaci, H., Pasi, G., 2020. Fuzzy lattice operations on first-order terms over signatures with similar constructors: A constraint-based approach. Fuzzy Sets Syst. 391, 1–46. URL: `https://doi.org/10.1016/j.fss.2019.03.019`, doi:`10.1016/J.FSS.2019.03.019`.

Baader, F., 1989. Unification in commutative theories. Journal of Symbolic Computation 8, 479–497.

Baader, F., Nipkow, T., 1998. Term rewriting and all that. Cambridge university press.

Benanav, D., Kapur, D., Narendran, P., 1987. Complexity of matching problems. Journal of symbolic computation 3, 203–216.

Boneva, I., Ciucanu, R., Staworko, S., 2013. Simple schemas for unordered xml. arXiv preprint arXiv:1303.4277 .

Boneva, I., Ciucanu, R., Staworko, S., 2015. Schemas for unordered xml on a dime. Theory of Computing Systems 57, 337–376.

Bry, F., Patranjan, P., Schaffert, S., 2004. Xcerpt and xchange - logic programming languages for querying and evolution on the web, in: Demoen, B., Lifschitz, V. (Eds.), Logic Programming, 20th International Conference, ICLP 2004, Saint-Malo, France, September 6-10, 2004, Proceedings, Springer. pp. 450–451. URL: `https://doi.org/10.1007/978-3-540-27775-0_33`, doi:`10.1007/978-3-540-27775-0\_33`.

Coelho, J., Dundua, B., Florido, M., Kutsia, T., 2010. A rule-based approach to XML processing and web reasoning, in: Hitzler, P., Lukasiewicz, T. (Eds.), Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings, Springer. pp. 164–172. URL: `https://doi.org/10.1007/978-3-642-15918-3_13`, doi:`10.1007/978-3-642-15918-3\_13`.

Cornejo, M.E., Moreno, J.M., Rubio-Manzano, C., 2018. Towards a full fuzzy unification in the bousi prolog system, in: 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE. pp. 1–7.

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

Dundua, B., Labadze, D., Tsereteli, T., 2025a. Fuzzy similarity and proximity constraint solving in commutative theories. `https://www3.risc.jku.at/projects/stout/library.html`.

Dundua, B., Labadze, D., Tsereteli, T., 2025b. Similarity-based constraint solving modulo commutativity: A unified framework, in: 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE. pp. 1–6.

Ehling, G., Kutsia, T., 2024. Solving quantitative equations, in: Benzmüller, C., Heule, M.J.H., Schmidt, R.A. (Eds.), Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part II, Springer. pp. 381–400. URL: `https://doi.org/10.1007/978-3-031-63501-4_20`, doi:`10.1007/978-3-031-63501-4\_20`.

Eker, S.M., 1995. Associative-commutative matching via bipartite graph matching. The Computer Journal 38, 381–399.

Hosoya, H., Pierce, B.C., 2003. Xduce: A statically typed xml processing language. ACM Transactions on Internet Technology (TOIT) 3, 117–148.

Jaffar, J., Maher, M.J., 1994. Constraint logic programming: A survey. J. Log. Program. 19/20, 503–581. URL: `https://doi.org/10.1016/0743-1066(94)90033-7`, doi:`10.1016/0743-1066(94)90033-7`.

Julián-Iranzo, P., Moreno, G., Riaza, J.A., 2020. The fuzzy logic programming language fasill: design and implementation. International Journal of Approximate Reasoning 125, 139–168.

Julián-Iranzo, P., Rubio-Manzano, C., 2015. Proximity-based unification theory. Fuzzy Sets and Systems 262, 21–43.

Julián-Iranzo, P., Sáenz-Pérez, F., 2018. An efficient proximity-based unification algorithm, in: 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE. pp. 1–8.

Koyutürk, M., Kim, Y., Topkara, U., Subramaniam, S., Szpankowski, W., Grama, A., 2006. Pairwise alignment of protein interaction networks. Journal of Computational Biology 13, 182–199.

Kutsia, T., Pau, C., 2019a. Matching and generalization modulo proximity and tolerance relations, in: Özgün, A., Zinova, Y. (Eds.), Language, Logic, and Computation - 13th International Tbilisi Symposium, TbiLLC 2019, Batumi, Georgia, September 16-20, 2019, Revised Selected Papers, Springer. pp. 323–342. URL: `https://doi.org/10.1007/978-3-030-98479-3_16`, doi:`10.1007/978-3-030-98479-3\_16`.

Kutsia, T., Pau, C., 2019b. Solving proximity constraints, in: Gabbrielli, M. (Ed.), Logic-Based Program Synthesis and Transformation - 29th International Symposium, LOPSTR 2019, Porto, Portugal, October 8-10, 2019, Revised Selected Papers, Springer. pp. 107–122. URL: `https://doi.org/10.1007/978-3-030-45260-5_7`, doi:`10.1007/978-3-030-45260-5\_7`.

Lohrey, M., Maneth, S., Reh, C.P., 2017. Compression of unordered xml trees, in: 20th International Conference on Database Theory, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Germany. pp. 1–17.

Nippa, D.F., Müller, A.T., Atz, K., Konrad, D.B., Grether, U., Martin, R.E., Schneider, G., 2025. Simple user-friendly reaction format. Molecular Informatics 44, e202400361.

Oliveira, A., Tessarolli, G., Ghiotto, G., Pinto, B., Campello, F., Marques, M., Oliveira, C., Rodrigues, I., Kalinowski, M., Souza, U., et al., 2018. An efficient similarity-based approach for comparing xml documents. Information Systems 78, 40–57.

Pau, C., Kutsia, T., 2021. 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. pp. 1–6. URL: `https://doi.org/10.1109/FUZZ45933.2021.9494438`, doi:`10.1109/FUZZ45933.2021.9494438`.

Sessa, M.I., 2002. Approximate reasoning by similarity-based SLD resolution. Theor. Comput. Sci. 275, 389–426. URL: `https://doi.org/10.1016/S0304-3975(01)00188-8`, doi:`10.1016/S0304-3975(01)00188-8`.

Stickel, M.E., 1981. A unification algorithm for associative-commutative functions. Journal of the ACM (JACM) 28, 423–434.