Solving Proximity Constraints*

Temur Kutsia and Cleo Pau

RISC, Johannes Kepler University Linz, Austria

Abstract. Proximity relations are binary fuzzy relations, which satisfy reflexivity and symmetry properties, but are not transitive. This relation induces the notion of distance between function symbols, which is further extended to terms. For two given terms we aim at bringing them "sufficiently close" to each other, by finding an appropriate substitution. A similar problem has been addressed by Julian-Iranzo and Rubio-Manzano. Unlike their work, we consider unrestricted proximal candidates, by allowing them to be close to two terms that themselves are not close to each other. We present an algorithm which works in two phases: first reducing the unification problem to constraints over sets of function symbols, and then solving the obtained constraints. The problem is decidable and finitary.

1 Introduction

Proximity relations are reflexive and symmetric fuzzy binary relations. They generalize similarity relations, which are a fuzzy version of equivalence. Proximity relations help to represent fuzzy information in situations, where similarity is not adequate. For unification, working modulo proximity or similarity means to treat different function symbols as if they were the same when the given relation asserts they are "close enough" to each other.

Unification for similarity relations was studied in [2] in the context of fuzzy logic programming. This work was generalized for proximity relations in [1], where the authors introduced the notion of proximity-based unification under a certain restriction imposed on the proximity relation. The restriction requires that the same symbol can not be close to two symbols at the same time, when those symbols are not close to each other. One should choose one of them as the proximal candidate to the given symbol. Looking at the proximity relation as an undirected graph, this restriction implies that cliques in the graph are disjoint. From the unification point of view, it means that $p(x, x)$ can not be unified with $p(a, c)$ when a and c are not close to each other, even if there exists a b which is close both to a and c.

In this paper we consider the general case: unification for a proximity relation without any restriction, and develop an algorithm which computes a compact representation of the set of solutions.

2 Preliminaries

Proximity relations. We define basic notions about proximity relations following [1]. 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 $0 \leq \lambda \leq 1$, then the λ-*cut* of $\mathcal{R}$ on S, denoted $\mathcal{R}_\lambda$, is an ordinary (crisp) relation on S defined as $\mathcal{R}_\lambda := \{(s_1, s_2) \mid \mathcal{R}(s_1, s_1) \geq \lambda\}$. In the role of T-norm $\wedge$ we take the minimum.

A fuzzy relation $\mathcal{R}$ on a set S is called a *proximity relation* on S iff it is reflexive and symmetric. The *proximity class of level* λ *of* $s \in S$ (a λ-*neighborhood* of s) is a set $\mathbf{pc}(s, \mathcal{R}, \lambda) = \{s' \mid \mathcal{R}(s, s') \geq \lambda\}$. When $\mathcal{R}$ and λ are fixed, we simply write $\mathbf{pc}(s)$.

Terms. Given a set of variables $\mathcal{V}$ and a fixed arity signature Σ, *terms* over Σ and $\mathcal{V}$ are defined as usual, by the grammar $t := x \mid f(t_1, \ldots, t_n)$, where $x \in \mathcal{V}$ and $f \in \Sigma$ is n-ary. We assume to have Σ partitioned into a set of function symbols $\mathcal{F}$ and a set of names $\mathcal{N}$, having symbols of each arity in each of them.

The set of terms over $\mathcal{V}$ and Σ (resp. over $\mathcal{F}$, $\mathcal{N}$), is denoted by $\mathcal{T}(\Sigma, \mathcal{V})$ (resp. $\mathcal{T}(\mathcal{F}, \mathcal{V})$, $\mathcal{T}(\mathcal{N}, \mathcal{V})$). We denote variables by x, y, z, arbitrary function symbols by f, g, h, constants by a, b, c, names by $\mathrm{N}, \mathrm{M}, \mathrm{K}$, and terms by s, t, r. The elements of $\mathcal{T}(\mathcal{F}, \mathcal{V})$ are called $\mathcal{F}$-terms. The elements of $\mathcal{T}(\mathcal{N}, \mathcal{V})$ are $\mathcal{N}$-terms.

The *head* of an $\mathcal{F}$-term is defined as $\mathit{head}(x) := x$ and $\mathit{head}(f(t_1, \ldots, t_n)) := f$.

$\mathcal{F}$-*Substitutions* (resp. $\mathcal{N}$-substitutions) are mappings from variables to $\mathcal{F}$-terms (resp. to $\mathcal{N}$-terms), where all but finitely many variables are mapped to themselves. The letters $\sigma, \vartheta, \varphi$ are used for $\mathcal{F}$-substitutions, and

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

their upright versions $\sigma, \vartheta, \varphi$ for $\mathcal{N}$-substitutions. The identity substitution is denoted by Id. We use the usual set notation for substitutions, writing, e.g., σ as $\sigma = \{x \mapsto \sigma(x) \mid x \neq \sigma(x)\}$. Substitution application and composition are defined in the standard way.

For simplicity, below we refer to $\mathcal{F}$-terms and $\mathcal{F}$-substitutions just by terms and substitutions, and only mention $\mathcal{N}$-terms and $\mathcal{N}$-substitutions explicitly.

A *name-class mapping* Φ is a finite mapping from names to sets of function symbols such that if $\mathrm{N} \in dom(\Phi)$ (where *dom* is the domain of mapping), then N and each $f \in \Phi(\mathrm{N})$ have the same arity.

Proximity relations over terms and substitutions. Each proximity relation $\mathcal{R}$ we consider in this paper obeys the following restrictions: (a) It is defined on $\mathcal{F} \cup \mathcal{V}$; (b) $\mathcal{R}(f, g) = 0$ if $arity(f) \neq arity(g)$; (c) $\mathcal{R}(f, x) = 0$ if $f \in \mathcal{F}$ and $x \in \mathcal{V}$, (d) $\mathcal{R}(x, y) = 0$ if $x \neq y$, for all $x, y \in \mathcal{V}$. We extend such a relation $\mathcal{R}$ to terms: (i) $\mathcal{R}(s, t) := 0$ if $\mathcal{R}(head(s), head(t)) = 0$. (ii) $\mathcal{R}(s, t) := 1$ if $s = t$ and $s, t \in \mathcal{V}$. (iii) $\mathcal{R}(s, t) := \mathcal{R}(f, g) \wedge \mathcal{R}(s_1, t_1) \wedge \cdots \wedge \mathcal{R}(s_n, t_n)$, if $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_n)$.

Two terms s and t are $(\mathcal{R}, \lambda)$-*close* to each other, written $s \simeq_{\mathcal{R},\lambda} t$, if $\mathcal{R}(s, t) \geq \lambda$. We say that s is $(\mathcal{R}, \lambda)$-*more general than* t and write $s \precsim_{\mathcal{R},\lambda} t$, if there exists a substitution σ such that $s\sigma \simeq_{\mathcal{R},\lambda} t$. This relation is defined for substitutions as well: $\sigma \precsim_{\mathcal{R},\lambda} \vartheta$ iff there exists φ such that $x\sigma\varphi \simeq_{\mathcal{R},\lambda} x\vartheta$ for all x.

The relation $\precsim_{\mathcal{R},\lambda}$ is not a quasi-order: it is reflexive, but not transitive.

Definition 1 (Approximate unification problem). *Given a proximity relation $\mathcal{R}$ and a cut value λ, an $(\mathcal{R}, \lambda)$-*unification problem *Γ is a finite set of approximate equations (i.e., pairs) of terms, written as $\Gamma := \{s_1 \simeq^?_{\mathcal{R},\lambda} t_1, \ldots, s_n \simeq^?_{\mathcal{R},\lambda} t_n\}$. The question mark indicates that it is supposed to be solved. An* unifier *of Γ is a substitution σ such that $s_i\sigma \simeq_{\mathcal{R},\lambda} t_i\sigma$ for all $1 \leq i \leq n$.*

Instead of writing "a unifier of an $(\mathcal{R}, \lambda)$-unification problem Γ", we often shortly say "an $(\mathcal{R}, \lambda)$-unifier of Γ". If σ is an $(\mathcal{R}, \lambda)$-unifier of Γ and $\sigma \preceq_{\mathcal{R},\lambda} \vartheta$, it might be that ϑ is not an $(\mathcal{R}, \lambda)$-unifier of Γ. For instance, if $\mathcal{R}_\lambda = \{(a, b), (b, c)\}$, then $\sigma = \{x \mapsto b\}$ is an $(\mathcal{R}, \lambda)$-unifier of $\{x \simeq^?_{\mathcal{R},\lambda} a\}$. Besides, $\sigma \preceq_{\mathcal{R},\lambda} \vartheta = \{x \mapsto c\}$, but ϑ is not an $(\mathcal{R}, \lambda)$-unifier of $\{x \simeq^?_{\mathcal{R},\lambda} a\}$.

However, if σ is an $(\mathcal{R}, \lambda)$-unifier of Γ, then $\sigma\varphi$ is also an $(\mathcal{R}, \lambda)$-unifier of Γ for any φ.

It might happen that a minimal complete set of $(\mathcal{R}, \lambda)$-unifiers (defined by using $\precsim_{\mathcal{R},\lambda}$ and $\simeq_{\mathcal{R},\lambda}$) of some Γ contains two substitutions σ and ϑ such that $\sigma \simeq_{\mathcal{R},\lambda} \vartheta$, but it is forbidden that $\vartheta = \sigma\varphi$ for some φ.

Definition 2. *Given $\mathcal{R}$ and λ, an $(\mathcal{R}, \lambda)$-neighborhood equation has one of the following forms: $f \approx^?_{\mathcal{R},\lambda} g$, $\mathrm{N} \approx^?_{\mathcal{R},\lambda} g$, $g \approx^?_{\mathcal{R},\lambda} \mathrm{N}$, or $\mathrm{N} \approx^?_{\mathcal{R},\lambda} \mathrm{M}$, where $f, g \in \mathcal{F}$ and $\mathrm{N}, \mathrm{M} \in \mathcal{N}$. The question mark indicates that they have to be solved.*

We say that a name-class mapping Φ is a solution of an $(\mathcal{R}, \lambda)$-neighborhood equation if

- *the equation is $f \approx^?_{\mathcal{R},\lambda} g$ and $f \approx_{\mathcal{R},\lambda} g$ holds (in this case any Φ would be a solution), or*
- *the equation is $\mathrm{N} \approx^?_{\mathcal{R},\lambda} g$ or $g \approx^?_{\mathcal{R},\lambda} \mathrm{N}$ and $f \approx_{\mathcal{R},\lambda} g$ holds for all $f \in \Phi(\mathrm{N})$, or*
- *the equation is $\mathrm{N} \approx^?_{\mathcal{R},\lambda} \mathrm{M}$ and $f \approx_{\mathcal{R},\lambda} g$ holds for all $f \in \Phi(\mathrm{N})$ and $g \in \Phi(\mathrm{M})$.*

An $(\mathcal{R}, \lambda)$-neighborhood constraint is a finite set of $(\mathcal{R}, \lambda)$-neighborhood equations. A name-class mapping Φ is a solution of an $(\mathcal{R}, \lambda)$-neighborhood constraint C if it is a solution of every $(\mathcal{R}, \lambda)$-neighborhood equation in C.

We shortly write "an $(\mathcal{R}, \lambda)$-solution to C" instead of "a solution to an $(\mathcal{R}, \lambda)$-neighborhood constraint C". Given a name-class mapping Φ and an $\mathcal{N}$-term t, we define a set of terms $\Phi(t)$ as

$$\Phi(x) := \{x\}, \qquad \Phi(\mathrm{N}(t_1, \ldots, t_n)) := \{f(s_1, \ldots, s_n) \mid f \in \Phi(\mathrm{N}), s_i \in \Phi(t_i), 1 \leq i \leq n\}.$$

The notation extends to $\mathcal{N}$-substitutions as well: $\Phi(\sigma) := \{\sigma \mid x\sigma \in \Phi(x\sigma) \text{ for all } x \in \mathcal{V}\}$.

Definition 3. *We say that a set of term equations $\{x \simeq^?_{\mathcal{R},\lambda} t\} \uplus P$ contains an* occurrence cycle *for the variable x if $t \notin \mathcal{V}$ and either*

- *$x \in var(t)$, or*
- *there exist term-pairs $(x_1, t_1[x_2]), \ldots, (x_n, t_n[x])$ such that $x_1 \in var(t)$ and for each $1 \leq i \leq n$, P contains an equation $x_i \simeq^?_{\mathcal{R},\lambda} t_i$ or $t_i \simeq^?_{\mathcal{R},\lambda} x_i$. (The notation $t[x]$ means that x occurs in t.)*

Lemma 1. *If a set of term equations contains an occurrence cycle for some variable, then it has no $(\mathcal{R}, \lambda)$-approximate unifier for any proximity relation $\mathcal{R}$ and cut value λ.*

Given an approximate unification problem Γ, our goal is to obtain a compact representation of its (minimal) complete set of unifiers as a pair of an $\mathcal{N}$-substitution σ and a name-class mapping Φ, so that $\Phi(\sigma)$ (restricted to the variables of Γ) gives the desired set. The algorithms below construct such a representation.

3 The Algorithms

In the rules below we will use the *renaming function* $\rho : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathcal{T}(\mathcal{N}, \mathcal{V})$. Applied to a Σ-term, ρ gives its fresh copy, an $\mathcal{N}$-term, obtained by replacing each occurrence of a symbol from Σ by a new name and each variable occurrence by a fresh variable. For instance, if the term is $f(\mathrm{N}(a, x, x, f(a)))$, where $f, a \in \mathcal{F}$ and $\mathrm{N} \in \mathcal{N}$, then $\rho(f(\mathrm{N}(a, x, x, f(a))) = \mathrm{N}_1(\mathrm{N}_2(\mathrm{N}_3, x_1, x_2, \mathrm{N}_4(\mathrm{N}_5)))$, where $\mathrm{N}_1, \mathrm{N}_2, \mathrm{N}_3, \mathrm{N}_4, \mathrm{N}_5 \in \mathcal{N}$ are new names and x_1, x_2 are new variables.

The algorithm consists of two phases: pre-unification and constraint solving. In the pre-unification phase we either obtain a pre-unifier together with neighborhood constraints which have to be solved in the second phase, or get a failure which indicates that the problem does not have a solution.

3.1 Pre-Unification Rules

An *equational configuration* is a triple $P; C; \sigma$, where

- P is the unification problem to be solved. It is initialized with the equation between the original terms;
- C is a neighborhood constraint;
- σ is the pre-unifier computed so far, initialized by Id.

The pre-unification algorithm takes given terms s and t, creates the initial configuration $\{s \simeq^?_{\mathcal{R},\lambda} t\}; \emptyset; Id$ and applies the rules given below exhaustively.

The rules are very similar to the syntactic unification algorithm with the difference that here the function symbol clash does not happen unless their arities differ, and variables are not replaced by other variables until the very end. (The notation $\overline{exp_n}$ in the rules below abbreviates the sequence $exp_1, \ldots, exp_n$.)

(Tri) Trivial:	$\{x \simeq^?_{\mathcal{R},\lambda} x\} \uplus P; C; \sigma \Longrightarrow P; C; \sigma$.
(Dec) Decomposition:	$\{F(\overline{s_n}) \simeq^?_{\mathcal{R},\lambda} G(\overline{t_n})\} \uplus P; C; \sigma \Longrightarrow \{\overline{s_n \simeq^?_{\mathcal{R},\lambda} t_n}\} \cup P; \{F \approx^?_{\mathcal{R},\lambda} G\} \cup C; \sigma$, where $F, G \in \Sigma$.
(VE) Var. Elim.:	$\{x \simeq^?_{\mathcal{R},\lambda} t\} \uplus P; C; \sigma \Longrightarrow \{t' \simeq^?_{\mathcal{R},\lambda} t\} \cup P\{x \mapsto t'\}; C; \sigma\{x \mapsto t'\}$, where $t \notin \mathcal{V}$, there is no occurrence cycle for x in $\{x \simeq^?_{\mathcal{R},\lambda} t\} \uplus P$, and $t' = \rho(t)$.
(Ori) Orient:	$\{t \simeq^?_{\mathcal{R},\lambda} x\} \uplus P; C; \sigma \Longrightarrow \{x \simeq^?_{\mathcal{R},\lambda} t\} \cup P; C; \sigma$, if $t \notin \mathcal{V}$.
(Cla) Clash:	$F(\overline{s_n}) \simeq^?_{\mathcal{R},\lambda} G(\overline{t_m})\} \uplus P; C; \sigma \Longrightarrow \bot$, where $F, G \in \Sigma$ and $n \neq m$.
(Occ) Occur Check:	$\{x \simeq^?_{\mathcal{R},\lambda} t\} \uplus P; C; \sigma \Longrightarrow \bot$, if there is an occurrence cycle for x in $\{x \simeq^?_{\mathcal{R},\lambda} t\} \uplus P$.
(VO) Vars Only:	$\{x \simeq^?_{\mathcal{R},\lambda} y, \overline{x_n \simeq^?_{\mathcal{R},\lambda} y_n}\}; C; \sigma \Longrightarrow \{\overline{x_n \simeq^?_{\mathcal{R},\lambda} y_n}\}\{x \mapsto y\}; C; \sigma\{x \mapsto y\}$.

It is easy to see that the pre-unification algorithm terminates either with $\bot$ or with $\emptyset; C; \sigma$, where C is a neighborhood constraint and σ is an $\mathcal{N}$-substitution. When the result is $\bot$, there is no unifier of s and t: terms of different number of arguments can not be unified (Clash), and a system with occurrence cycle (Occur Check) has no solutions (Lemma 1).

Theorem 1 (Soundness of pre-unification). *Let s and t be two terms, such that the pre-unification algorithm gives $\{s \simeq^?_{\mathcal{R},\lambda} t\}; \emptyset; Id \Longrightarrow^* \emptyset; C; \sigma$. Let Φ be an $(\mathcal{R}, \lambda)$-solution of C. Then any substitution in the set $\Phi(\sigma)$ is an $(\mathcal{R}, \lambda)$-unifier of s and t.*

Theorem 2 (Completeness of pre-unification). *Let ψ be an $(\mathcal{R}, \lambda)$-unifier of s and t. Then there exists a derivation $\{s \simeq^?_{\mathcal{R},\lambda} t\}; \emptyset; Id \Longrightarrow^* \emptyset; C; \sigma$ such that $\varphi \preceq_{\mathcal{R},\lambda} \psi$ for some $\varphi \in \Phi(\sigma|_{var(s) \cup var(t)})$, where Φ is an $(\mathcal{R}, \lambda)$-solution of C.*

For solving neighborhood constraints, we will introduce an algorithm in the next section. But before that we illustrate the pre-unification rules with a couple of examples:

Example 1. Let $s = p(x, y, x)$ and $t = q(f(a), g(d), y)$. Then the pre-unification algorithm stops with the configuration $\emptyset; C, \sigma$, where $C = \{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} g,\ \mathrm{N}_4 \approx^?_{\mathcal{R},\lambda} d,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_3,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\}$ and $\sigma = \{x \mapsto \mathrm{N}_1(\mathrm{N}_2), y \mapsto \mathrm{N}_3(\mathrm{N}_4)\}$.

Assume that for the given λ-cut, the proximity relation consists of pairs $\mathcal{R}_\lambda = \{(a, b), (b, c), (c, d), (a, b'), (b', c'), (c', d), (f, g), (p, q)\}$. The obtained constraint can be solved, e.g., by the name-class mappings $\Phi = [\mathrm{N}_1 \mapsto \{f, g\}, \mathrm{N}_2 \mapsto \{b\}, \mathrm{N}_3 \mapsto \{f, g\}, \mathrm{N}_4 \mapsto \{c\}]$ and $\Phi' = [\mathrm{N}_1 \mapsto \{f, g\}, \mathrm{N}_2 \mapsto \{b'\}, \mathrm{N}_3 \mapsto \{f, g\}, \mathrm{N}_4 \mapsto \{c'\}]$. From them and σ we can get the sets $\Phi(\sigma)$ and $\Phi'(\sigma)$ of $(\mathcal{R}, \lambda)$-unifiers of s and t. Each of them consists of 4 substitutions.

If we did not have the VO rule and allowed the use of VE rule instead, we might have ended up with the unification problem $\{y \simeq^?_{\mathcal{R},\lambda} f(a), y \simeq^?_{\mathcal{R},\lambda} g(d)\}$, which does not have a solution, because the neighborhoods of a and d do not have a common element. Hence, we would have lost a solution.

Example 2. Let $s = p(x, x)$ and $t = q(f(y, y), f(a, c))$. The pre-unification algorithm stops with the configuration $\emptyset; P; \sigma$, where $P = \{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{M} \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{M}\}$ and $\sigma = \{x \mapsto \mathrm{N}_1(\mathrm{N}_2, \mathrm{N}_3),\ y_1 \mapsto \mathrm{N}_2,\ y_2 \mapsto \mathrm{N}_3,\ y \mapsto \mathrm{M}\}$. Let $\mathcal{R}_\lambda = \{(a, a_1), (a_1, b), (b, c_1), (c_1, c), (p, q)\}$. Then C is solved by $\Phi = [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a_1\}, \mathrm{M} \mapsto \{b\}, \mathrm{N}_3 \mapsto \{c_1\}]$ and $\Phi(\sigma|_{var(s)\cup var(t)})$ contains only one element, an $(\mathcal{R}, \lambda)$-unifier $\sigma = \{x \mapsto f(a_1, c_1),\ y \mapsto b\}$ of s and t. Indeed, $s\sigma = p(f(a_1, c_1), f(a_1, c_1)) \simeq_{\mathcal{R},\lambda} q(f(b, b), f(a, c)) = t\sigma$.

This example illustrates the necessity of introducing a fresh variable for *each occurrence* of a variable by the renaming function in the VE rule. If we used the same new variable, say y', for both occurrences of y in $f(y, y)$ (instead of using y_1 and y_2 as above), we would get the configuration $\emptyset; \{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\}; \{x \mapsto \mathrm{N}_1(\mathrm{N}_2, \mathrm{N}_2),\ y' \mapsto \mathrm{N}_2,\ y \mapsto \mathrm{N}_2\}$. But for the given $\mathcal{R}_\lambda$, the constraint $\{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\}$ does not have a solution (because the neighborhoods of a and c are not close to each other). Hence, we would lose a unifier.

3.2 Rules for Neighborhood Constraints

Let Φ be a *name-class mapping*. Then the function of updating Φ by a mapping $\mathrm{N} \to S$ for $\mathrm{N} \in \mathcal{N}$ and $S \subset \mathcal{F}$, where N and the elements of S have the same arity, is defined as

$$\mathsf{update}(\Phi, \mathrm{N} \mapsto S) = \begin{cases} \Phi \cup [\mathrm{N} \mapsto S], \text{ if } \mathrm{N} \notin \mathit{dom}(\Phi), \\ (\Phi \setminus [\mathrm{N} \mapsto \Phi(\mathrm{N})]) \cup [\mathrm{N} \mapsto S \cap \Phi(\mathrm{N})], \text{ otherwise.} \end{cases}$$

We write $\mathsf{update}(\Phi, \mathrm{N}_1 \to S_1, \ldots, \mathrm{N}_n \to S_n)$ for $\mathsf{update}(\cdots \mathsf{update}(\Phi, \mathrm{N}_1 \to S_1), \ldots, \mathrm{N}_n \to S_n)$.

A *constraint configuration* is a pair $C; \Phi$, where C is a set of $(\mathcal{R}, \lambda)$-neighborhood constraints to be solved, and Φ is a name-class mapping (as a set of rules), representing the $(\mathcal{R}, \lambda)$-solution computed so far. It is initialized by the empty set. The constraint simplification algorithm $\mathcal{CS}$ transforms constraint configurations, exhaustively applying the following rules:

(FFS) Function Symbols:	$\{f \approx^?_{\mathcal{R},\lambda} g\} \uplus C; \Phi \Longrightarrow C; \Phi$, if $\mathcal{R}(f, g) \geq \lambda$.
(NFS) Name vs Function Symbol:	$\{\mathrm{N} \approx^?_{\mathcal{R},\lambda} g\} \uplus C; \Phi \Longrightarrow C; \mathsf{update}(\Phi, \mathrm{N} \mapsto \mathbf{pc}(g, \mathcal{R}, \lambda))$.
(FSN) Function Symbol vs Name:	$\{g \approx^?_{\mathcal{R},\lambda} \mathrm{N}\} \uplus C; \Phi \Longrightarrow \{\mathrm{N} \approx^?_{\mathcal{R},\lambda} g\} \cup C; \Phi$.
(NN1) Name vs Name 1:	$\{\mathrm{N} \approx^?_{\mathcal{R},\lambda} \mathrm{M}\} \uplus C; \Phi \Longrightarrow C; \mathsf{update}(\Phi, \mathrm{N} \mapsto \{f\}, \mathrm{M} \mapsto \mathbf{pc}(f, \mathcal{R}, \lambda))$, where $\mathrm{N} \in \mathit{dom}(\Phi)$, $f \in \Phi(\mathrm{N})$.
(NN2) Name vs Name 2:	$\{\mathrm{M} \approx^?_{\mathcal{R},\lambda} \mathrm{N}\} \uplus C; \Phi \Longrightarrow \{\mathrm{N} \approx^?_{\mathcal{R},\lambda} \mathrm{M}\} \cup C; \Phi$, where $\mathrm{M} \notin \mathit{dom}(\Phi)$, $\mathrm{N} \in \mathit{dom}(\Phi)$.
(Fail1) Failure 1:	$\{f \approx^?_{\mathcal{R},\lambda} g\} \uplus C; \Phi \Longrightarrow \bot$, if $\mathcal{R}(f, g) < \lambda$.
(Fail2) Failure 2:	$C; \Phi \Longrightarrow \bot$, if there exists $\mathrm{N} \in \mathit{dom}(\Phi)$ such that $\Phi(\mathrm{N}) = \emptyset$.

The NN1 rule causes branching, generating n branches where n is the number of elements in $\Phi(\mathrm{N})$ (assuming that the proximity class of each symbol is finite). When the derivation does not fail, the terminal configuration has the form $\{\mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{M}_1, \ldots, \mathrm{N}_n \approx^?_{\mathcal{R},\lambda} \mathrm{M}_n\}; \Phi$, where for each $1 \leq i \leq n$, $\mathrm{N}_i, \mathrm{M}_i \notin \mathit{dom}(\Phi)$. Such a constraint is trivially solvable.

Theorem 3 (Soundness of $\mathcal{CS}$). *Let C be an $(\mathcal{R},\lambda)$-neighborhood constraint such that $\mathcal{CS}$ produces a maximal derivation $C;\emptyset \Longrightarrow^* C';\Phi$. Then Φ is an $(\mathcal{R},\lambda)$-solution of $C \setminus C'$, and C' is a set of constraints between names which is trivially $(\mathcal{R},\lambda)$-satisfiable.*

Theorem 4 (Completeness of $\mathcal{CS}$). *Let C be an $(\mathcal{R},\lambda)$-neighborhood constraint and Φ be its solution such that every name from $dom(\Phi)$ appears in C. Let $dom(\Phi) = \{\mathrm{N}_1,\ldots,\mathrm{N}_n\}$. Then for each n-tuple $c_1 \in \Phi(\mathrm{N}_1),\ldots,c_n \in \Phi(\mathrm{N}_n)$ there exists a maximal $\mathcal{CS}$-derivation $C;\emptyset \Longrightarrow^* C';\Phi'$ such that for each $1 \le i \le n$, either $c_i \in \Phi'(\mathrm{N}_i)$, or there exists $1 \le j \le n$ such that $c_i \in \Phi(\mathrm{N}_j)$ and $\mathrm{N}_i \approx^?_{\mathcal{R},\lambda} \mathrm{N}_j \in C'$.*

Remark 1. When a neighborhood constraint C is produced by the pre-unification algorithm, then every maximal $\mathcal{CS}$-derivation starting from $C;\emptyset$ ends either in $\bot$ or in the pair of the form $\emptyset;\Phi$. This is due to the fact that the VE rule, which introduces names in pre-unification problems, and the subsequent decomposition steps necessarily produce neighborhood equations of the form $\mathrm{N} \approx_{\mathcal{R},\lambda} f$ for each introduced N and for some f.

Example 3. The pre-unification derivation in Example 1 gives the neighborhood constraint $C = \{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} g,\ \mathrm{N}_4 \approx^?_{\mathcal{R},\lambda} d,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_3,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\}$. For $\mathcal{R}_\lambda = \{(a,b),(b,c),(c,d),(a,b'),(b',c'),(c',d),(f,g),(p,q)\}$, the constraint C can be solved by $\mathcal{CS}$ as follows:

$$\{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} g,\ \mathrm{N}_4 \approx^?_{\mathcal{R},\lambda} d,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_3,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\};\emptyset \Longrightarrow_{\mathsf{FFS}}$$
$$\{\mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} g,\ \mathrm{N}_4 \approx^?_{\mathcal{R},\lambda} d,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_3,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\};\emptyset \Longrightarrow_{\mathsf{NFS}}$$
$$\{\mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} g,\ \mathrm{N}_4 \approx^?_{\mathcal{R},\lambda} d,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_3,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\};[\mathrm{N}_1 \mapsto \{f,g\}] \Longrightarrow_{\mathsf{NFS}}$$
$$\{\mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} g,\ \mathrm{N}_4 \approx^?_{\mathcal{R},\lambda} d,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_3,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\};[\mathrm{N}_1 \mapsto \{f,g\},\ \mathrm{N}_2 \mapsto \{a,b,b'\}] \Longrightarrow_{\mathsf{NFS}}$$
$$\{\mathrm{N}_4 \approx^?_{\mathcal{R},\lambda} d,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_3,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\};[\mathrm{N}_1 \mapsto \{f,g\},\ \mathrm{N}_2 \mapsto \{a,b,b'\},\ \mathrm{N}_3 \mapsto \{f,g\}] \Longrightarrow_{\mathsf{NFS}}$$
$$\{\mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_3,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\};[\mathrm{N}_1 \mapsto \{f,g\},\ \mathrm{N}_2 \mapsto \{a,b,b'\},\ \mathrm{N}_3 \mapsto \{f,g\},\ \mathrm{N}_4 \mapsto \{d,c,c'\}].$$

Here the algorithm branches, since the rule $\mathsf{NN1}$ applies. Branch 1 continues with

$$\{\mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_3,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\};[\mathrm{N}_1 \mapsto \{f,g\},\mathrm{N}_2 \mapsto \{a,b,b'\},\mathrm{N}_3 \mapsto \{f,g\},\mathrm{N}_4 \mapsto \{d,c,c'\}] \Longrightarrow_{\mathsf{NN1}}$$
$$\{\mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\};[\mathrm{N}_1 \mapsto \{f\},\mathrm{N}_2 \mapsto \{a,b,b'\},\mathrm{N}_3 \mapsto \{f,g\},\mathrm{N}_4 \mapsto \{d,c,c'\}].$$

Branching again by $\mathsf{NN1}$ produces three subbranches. Branch 1.1 fails: $\emptyset;[\mathrm{N}_1 \mapsto \{f\},\mathrm{N}_2 \mapsto \{a\},\mathrm{N}_3 \mapsto \{f,g\},\mathrm{N}_4 \mapsto \emptyset] \Longrightarrow_{\mathsf{Fail2}} \bot$. Branch 1.2 and branch 1.3 give two solutions, respectively:

$$\Phi_1 = [\mathrm{N}_1 \mapsto \{f\},\mathrm{N}_2 \mapsto \{b\},\ \mathrm{N}_3 \mapsto \{f,g\},\mathrm{N}_4 \mapsto \{c\}]$$
$$\Phi_2 = [\mathrm{N}_1 \mapsto \{f\},\mathrm{N}_2 \mapsto \{b'\},\mathrm{N}_3 \mapsto \{f,g\},\mathrm{N}_4 \mapsto \{c'\}].$$

Branch 2 also expands into three subbranches, the first of which fails. The other two return two more solutions:

$$\Phi_3 = [\mathrm{N}_1 \mapsto \{g\},\mathrm{N}_2 \mapsto \{b\},\ \mathrm{N}_3 \mapsto \{f,g\},\mathrm{N}_4 \mapsto \{c\}]$$
$$\Phi_4 = [\mathrm{N}_1 \mapsto \{g\},\mathrm{N}_2 \mapsto \{b'\},\mathrm{N}_3 \mapsto \{f,g\},\mathrm{N}_4 \mapsto \{c'\}].$$

Referring to the name-class mappings Φ and Φ' and the substitution σ in Example 1, it is easy to observe that $\Phi(\sigma) \cup \Phi'(\sigma) = \Phi_1(\sigma) \cup \Phi_2(\sigma) \cup \Phi_3(\sigma) \cup \Phi_4(\sigma)$.

4 Final Remarks

We described our work in progress towards solving equational constraints over proximity relations. The immediate next step is to incorporate the computation of unification degree into the procedure and use it to return only those unifiers which bring the original terms as close as possible to each other.

References

1. P. Julián-Iranzo and C. Rubio-Manzano. Proximity-based unification theory. *Fuzzy Sets and Systems*, 262:21–43, 2015.
2. M. I. Sessa. Approximate reasoning by similarity-based SLD resolution. *Theor. Comput. Sci.*, 275(1-2):389–426, 2002.

A Examples

The selected equations are underlined.

Example 4. Let $s = p(x, y, x)$ and $t = q(f(a), g(d), y)$. The following is a pre-unification derivation, which shows how the neighborhood constraint and the substitution shown in Example 1 are computed:

$\{\underline{p(x,y,x) \simeq^?_{\mathcal{R},\lambda} q(f(a),g(d),y)}\}$; $\emptyset$; Id $\Longrightarrow_{\mathsf{Dec}}$

$\{\underline{x \simeq^?_{\mathcal{R},\lambda} f(a)}, y \simeq^?_{\mathcal{R},\lambda} g(d), x \simeq^?_{\mathcal{R},\lambda} y\}$; $\{p \approx^?_{\mathcal{R},\lambda} q\}$; Id $\Longrightarrow_{\mathsf{VE}}$

$\{\underline{\mathrm{N}_1(\mathrm{N}_2) \simeq^?_{\mathcal{R},\lambda} f(a)}, y \simeq^?_{\mathcal{R},\lambda} g(d), \mathrm{N}_1(\mathrm{N}_2) \simeq^?_{\mathcal{R},\lambda} y\}$; $\{p \approx^?_{\mathcal{R},\lambda} q\}$; $\{x \mapsto \mathrm{N}_1(\mathrm{N}_2)\}$ $\Longrightarrow_{\mathsf{Dec}^2}$

$\{\underline{y \simeq^?_{\mathcal{R},\lambda} g(d)}, \mathrm{N}_1(\mathrm{N}_2) \simeq^?_{\mathcal{R},\lambda} y\}$; $\{p \approx^?_{\mathcal{R},\lambda} q, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f, \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a\}$; $\{x \mapsto \mathrm{N}_1(\mathrm{N}_2)\}$ $\Longrightarrow_{\mathsf{VE}}$

$\{\underline{\mathrm{N}_3(\mathrm{N}_4) \simeq^?_{\mathcal{R},\lambda} g(d)}, \mathrm{N}_1(\mathrm{N}_2) \simeq^?_{\mathcal{R},\lambda} \mathrm{N}_3(\mathrm{N}_4)\}$; $\{p \approx^?_{\mathcal{R},\lambda} q, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f, \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a\}$;
$\{x \mapsto \mathrm{N}_1(\mathrm{N}_2), y \mapsto \mathrm{N}_3(\mathrm{N}_4)\}$ $\Longrightarrow_{\mathsf{Dec}^2}$

$\{\underline{\mathrm{N}_1(\mathrm{N}_2) \simeq^?_{\mathcal{R},\lambda} \mathrm{N}_3(\mathrm{N}_4)}\}$; $\{p \approx^?_{\mathcal{R},\lambda} q, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f, \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a, \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} g, \mathrm{N}_4 \approx^?_{\mathcal{R},\lambda} d\}$;
$\{x \mapsto \mathrm{N}_1(\mathrm{N}_2), y \mapsto \mathrm{N}_3(\mathrm{N}_4)\}$ $\Longrightarrow_{\mathsf{Dec}^2}$

$\emptyset$; $\{p \approx^?_{\mathcal{R},\lambda} q, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f, \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a, \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} g, \mathrm{N}_4 \approx^?_{\mathcal{R},\lambda} d, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_3, \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_4\}$;
$\{x \mapsto \mathrm{N}_1(\mathrm{N}_2), y \mapsto \mathrm{N}_3(\mathrm{N}_4)\}$.

Note that we could have chosen the second equation in $\{y \simeq^?_{\mathcal{R},\lambda} g(d), \mathrm{N}_1(\mathrm{N}_2) \simeq^?_{\mathcal{R},\lambda} y\}$ to eliminate y, but the obtained result would differ from the above computed one by the choice of names only.

Together with the output from Example 3, we report the solutions for $\mathcal{R}_\lambda = \{(a,b),(b,c),(c,d),(a,b'),(b',c'),(c',d),(f,g),(p,q)\}$,

$\Phi_1 = [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{b\}, \mathrm{N}_3 \mapsto \{f,g\}, \mathrm{N}_4 \mapsto \{c\}]$,
$\Phi_2 = [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{b'\}, \mathrm{N}_3 \mapsto \{f,g\}, \mathrm{N}_4 \mapsto \{c'\}]$,
$\Phi_3 = [\mathrm{N}_1 \mapsto \{g\}, \mathrm{N}_2 \mapsto \{b\}, \mathrm{N}_3 \mapsto \{f,g\}, \mathrm{N}_4 \mapsto \{c\}]$,
$\Phi_4 = [\mathrm{N}_1 \mapsto \{g\}, \mathrm{N}_2 \mapsto \{b'\}, \mathrm{N}_3 \mapsto \{f,g\}, \mathrm{N}_4 \mapsto \{c'\}]$,
$\sigma = \{x \mapsto \mathrm{N}_1(\mathrm{N}_2), y \mapsto \mathrm{N}_3(\mathrm{N}_4)\}$.

Example 5. Now we illustrate the steps made for computations in Example 2. Let $s = p(x, x)$ and $t = q(f(y, y), f(a, c))$. Then the pre-unification derivation looks as follows:

$\{\underline{p(x,x) \simeq^?_{\mathcal{R},\lambda} q(f(y,y),f(a,c))}\}$; $\emptyset$; Id $\Longrightarrow_{\mathsf{Dec}}$

$\{\underline{x \simeq^?_{\mathcal{R},\lambda} f(y,y)}, x \simeq^?_{\mathcal{R},\lambda} f(a,c)\}$; $\{p \approx^?_{\mathcal{R},\lambda} q\}$; Id $\Longrightarrow_{\mathsf{VE}}$

$\{\underline{\mathrm{N}_1(y_1,y_2) \simeq^?_{\mathcal{R},\lambda} f(y,y)}, \mathrm{N}_1(y_1,y_2) \simeq^?_{\mathcal{R},\lambda} f(a,c)\}$; $\{p \approx^?_{\mathcal{R},\lambda} q\}$; $\{x \mapsto \mathrm{N}_1(y_1,y_2)\}$ $\Longrightarrow_{\mathsf{Dec}}$

$\{y_1 \simeq^?_{\mathcal{R},\lambda} y, y_2 \simeq^?_{\mathcal{R},\lambda} y, \underline{\mathrm{N}_1(y_1,y_2) \simeq^?_{\mathcal{R},\lambda} f(a,c)}\}$; $\{p \approx^?_{\mathcal{R},\lambda} q, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f\}$; $\{x \mapsto \mathrm{N}_1(y_1,y_2)\}$ $\Longrightarrow_{\mathsf{Dec}}$

$\{y_1 \simeq^?_{\mathcal{R},\lambda} y, y_2 \simeq^?_{\mathcal{R},\lambda} y, \underline{y_1 \simeq^?_{\mathcal{R},\lambda} a}, y_2 \simeq^?_{\mathcal{R},\lambda} c\}$; $\{p \approx^?_{\mathcal{R},\lambda} q, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f\}$;
$\{x \mapsto \mathrm{N}_1(y_1,y_2)\}$ $\Longrightarrow_{\mathsf{VE,Dec}}$

$\{\mathrm{N}_2 \simeq^?_{\mathcal{R},\lambda} y, y_2 \simeq^?_{\mathcal{R},\lambda} y, \underline{y_2 \simeq^?_{\mathcal{R},\lambda} c}\}$; $\{p \approx^?_{\mathcal{R},\lambda} q, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f, \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a\}$;
$\{x \mapsto \mathrm{N}_1(\mathrm{N}_2, y_2), y_1 \mapsto \mathrm{N}_2\}$ $\Longrightarrow_{\mathsf{VE,Dec}}$

$\{\underline{\mathrm{N}_2 \simeq^?_{\mathcal{R},\lambda} y}, \mathrm{N}_3 \simeq^?_{\mathcal{R},\lambda} y\}$; $\{p \approx^?_{\mathcal{R},\lambda} q, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f, \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a, \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c\}$;
$\{x \mapsto \mathrm{N}_1(\mathrm{N}_2, \mathrm{N}_3), y_1 \mapsto \mathrm{N}_2, y_2 \mapsto \mathrm{N}_3\}$ $\Longrightarrow_{\mathsf{Ori,VE}}$

$\{\underline{\mathrm{N}_3 \simeq^?_{\mathcal{R},\lambda} \mathrm{M}}\}$; $\{p \approx^?_{\mathcal{R},\lambda} q, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f, \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a, \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c, \mathrm{M} \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\}$;
$\{x \mapsto \mathrm{N}_1(\mathrm{N}_2, \mathrm{N}_3), y_1 \mapsto \mathrm{N}_2, y_2 \mapsto \mathrm{N}_3, y \mapsto \mathrm{M}\}$ $\Longrightarrow_{\mathsf{Dec}}$

$\emptyset$; $\{p \approx^?_{\mathcal{R},\lambda} q, \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f, \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a, \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c, \mathrm{M} \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2, \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{M}\}$;

$\{x \mapsto \mathrm{N}_1(\mathrm{N}_2, \mathrm{N}_3),\ y_1 \mapsto \mathrm{N}_2,\ y_2 \mapsto \mathrm{N}_3,\ y \mapsto \mathrm{M}\}$.

If $\mathcal{R}_\lambda = \{(a, a_1), (a_1, b), (b, c_1), (c_1, c), (p, q)\}$, then the obtained constraint is satisfied by the assignment $[\mathrm{N}_1 \mapsto \{f\},\ \mathrm{N}_2 \mapsto \{a_1\},\ \mathrm{M} \mapsto \{b\},\ \mathrm{N}_3 \mapsto \{c_1\}]$, computed by $\mathcal{CS}$ as follows (where NN1 causes branching, we display only the success branch):

$\{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{M} \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{M}\};\ \emptyset \Longrightarrow_{\mathsf{FFS}}$
$\{\mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{M} \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{M}\};\ \emptyset \Longrightarrow_{\mathsf{NFS}}$
$\{\mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{M} \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{M}\};\ [\mathrm{N}_1 \mapsto \{f\}] \Longrightarrow_{\mathsf{NFS}}$
$\{\mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{M} \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{M}\};\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a, a_1\}] \Longrightarrow_{\mathsf{NFS}}$
$\{\mathrm{M} \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{M}\};\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a, a_1\}, \mathrm{N}_3 \mapsto \{c, c_1\}] \Longrightarrow_{\mathsf{NN2}}$
$\{\mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} \mathrm{M},\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{M}\};\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a, a_1\}, \mathrm{N}_3 \mapsto \{c, c_1\}] \Longrightarrow_{\mathsf{NN1}}$
$\{\mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{M}\};\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a_1\}, \mathrm{N}_3 \mapsto \{c, c_1\}, \mathrm{M} \mapsto \{b\}] \Longrightarrow_{\mathsf{NN1}}$
$\emptyset;\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a_1\}, \mathrm{N}_3 \mapsto \{c_1\}, \mathrm{M} \mapsto \{b\}]$.

Hence, the computed solution is

$\Phi = [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a_1\}, \mathrm{N}_3 \mapsto \{c_1\}, \mathrm{M} \mapsto \{b\}]$,
$\sigma = \{x \mapsto \mathrm{N}_1(\mathrm{N}_2, \mathrm{N}_3),\ y_1 \mapsto \mathrm{N}_2,\ y_2 \mapsto \mathrm{N}_3,\ y \mapsto \mathrm{M}\}$.
$\sigma|_{var(s) \cup var(t)} = \{x \mapsto \mathrm{N}_1(\mathrm{N}_2, \mathrm{N}_3),\ y \mapsto \mathrm{M}\}$.

If we used the same new variable, say y', for both occurrences of y in $f(y, y)$ (instead of using y_1 and y_2 as above), we would get the following pre-unification derivation:

$\{\underline{p(x, x) \simeq^?_{\mathcal{R},\lambda} q(f(y, y), f(a, c))}\};\ \emptyset;\ Id \Longrightarrow_{\mathsf{Dec}}$
$\{\underline{x \simeq^?_{\mathcal{R},\lambda} f(y, y)},\ x \simeq^?_{\mathcal{R},\lambda} f(a, c)\};\ \{p \approx^?_{\mathcal{R},\lambda} q\};\ Id \Longrightarrow_{\mathsf{VE}}$
$\{\underline{\mathrm{N}_1(y', y') \simeq^?_{\mathcal{R},\lambda} f(y, y)},\ \mathrm{N}_1(y', y') \simeq^?_{\mathcal{R},\lambda} f(a, c)\};\ \{p \approx^?_{\mathcal{R},\lambda} q\};\ \{x \mapsto \mathrm{N}_1(y', y')\} \Longrightarrow_{\mathsf{Dec}}$
$\{y' \simeq^?_{\mathcal{R},\lambda} y,\ \underline{\mathrm{N}(y', y') \simeq^?_{\mathcal{R},\lambda} f(a, c)}\};\ \{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f\};\ \{x \mapsto \mathrm{N}_1(y', y')\} \Longrightarrow_{\mathsf{Dec}}$
$\{y' \simeq^?_{\mathcal{R},\lambda} y,\ \underline{y' \simeq^?_{\mathcal{R},\lambda} a},\ y' \simeq^?_{\mathcal{R},\lambda} c\};\ \{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f\};\ \{x \mapsto \mathrm{N}_1(y', y')\} \Longrightarrow_{\mathsf{VE,Dec}}$
$\{\underline{\mathrm{N}_2 \simeq^?_{\mathcal{R},\lambda} y},\ \mathrm{N}_2 \simeq^?_{\mathcal{R},\lambda} c\};\ \{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a\};\ \{x \mapsto \mathrm{N}_1(\mathrm{N}_2, \mathrm{N}_2),\ y' \mapsto \mathrm{N}_2\} \Longrightarrow_{\mathsf{VE,Dec}}$
$\{\underline{\mathrm{N}_2 \simeq^?_{\mathcal{R},\lambda} \mathrm{N}_3}\};\ \{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c\};$
$\quad \{x \mapsto \mathrm{N}_1(\mathrm{N}_2, \mathrm{N}_3),\ y' \mapsto \mathrm{N}_2,\ y \mapsto \mathrm{N}_2\} \Longrightarrow_{\mathsf{Dec}}$
$\emptyset;\ \{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\};\ \{x \mapsto \mathrm{N}_1(\mathrm{N}_2, \mathrm{N}_2),\ y' \mapsto \mathrm{N}_2,\ y \mapsto \mathrm{N}_2\}$.

If $\mathcal{R}_\lambda = \{(a, a_1), (a_1, b), (b, c_1), (c_1, c), (p, q)\}$, the obtained neighborhood constraint does not have a solution:

$\{p \approx^?_{\mathcal{R},\lambda} q,\ \mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\};\ \emptyset \Longrightarrow_{\mathsf{FFS}}$
$\{\mathrm{N}_1 \approx^?_{\mathcal{R},\lambda} f,\ \mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\};\ \emptyset \Longrightarrow_{\mathsf{NFS}}$
$\{\mathrm{N}_2 \approx^?_{\mathcal{R},\lambda} a,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\};\ [\mathrm{N}_1 \mapsto \{f\}] \Longrightarrow_{\mathsf{NFS}}$
$\{\mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} c,\ \mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\};\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a, a_1\}] \Longrightarrow_{\mathsf{NFS}}$
$\{\mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\};\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a, a_1\}, \mathrm{N}_3 \mapsto \{c, c_1\}]$

From here, NN1 generates two branches. First:

$\{\mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\};\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a, a_1\}, \mathrm{N}_3 \mapsto \{c, c_1\}] \Longrightarrow_{\mathsf{NN1}}$
$\emptyset;\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a\}, \mathrm{N}_3 \mapsto \emptyset] \Longrightarrow_{\mathsf{Fail2}} \bot$.

Second:

$\{\mathrm{N}_3 \approx^?_{\mathcal{R},\lambda} \mathrm{N}_2\};\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a, a_1\}, \mathrm{N}_3 \mapsto \{c, c_1\}] \Longrightarrow_{\mathsf{NN1}}$
$\emptyset;\ [\mathrm{N}_1 \mapsto \{f\}, \mathrm{N}_2 \mapsto \{a_1\}, \mathrm{N}_3 \mapsto \emptyset] \Longrightarrow_{\mathsf{Fail2}} \bot$.