PROXIMITY-BASED UNIFICATION FOR FULLY FUZZY SIGNATURES

Cleo Pau and Temur Kutsia
RISC, Johannes Kepler University Linz

JOHANNES KEPLER UNIVERSITY LINZ

Der Wissenschaftsfonds.

Unification

Solving equations between terms.

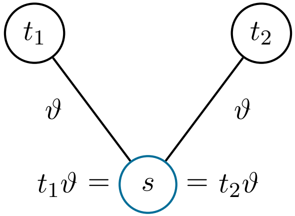
Fundamental technique, used in automated reasoning, rewriting, declarative programming, type inference, NLP, etc.

Syntactic unification

Unifying two terms t_1 and t_2:

Syntactic unification

Unifying two terms t_1 and t_2:



ϑ: a most general unifier of t_1 and t_2.

ϑ solves the unification problem $t_1 =^? t_2$.

Syntactic unification

$$\begin{tikzcd}
\boxed{f(x, g(x), g(y))} \arrow[dr, dash, "{\{x \mapsto a, z \mapsto g(y)\}}"'] & & \boxed{f(a, g(a), z)} \arrow[dl, dash, "{\{x \mapsto a, z \mapsto g(y)\}}"] \\
& \boxed{f(a, g(a), g(y))} &
\end{tikzcd}$$

$\{x \mapsto a, z \mapsto g(y)\}$ is a most general unifier of $f(x, g(x), g(y))$ and $f(a, g(a), z)$.

Syntactic unification

$f(x)$ and $g(a)$ are not unifiable because f and g are distinct function symbols.

From equalities to proximities

In these examples, the given information was precise.

Two symbols, terms, etc. are either equal or not.

How to deal with cases when the information is not perfect?

From equalities to proximities

Reasoning with incomplete, imperfect information is very common in human communication.

Its modeling is a highly nontrivial task.

There are various notions associated to such information (e.g., uncertainty, imprecision, vagueness, fuzziness).

From equalities to proximities

Reasoning with incomplete, imperfect information is very common in human communication.

Its modeling is a highly nontrivial task.

There are various notions associated to such information (e.g., uncertainty, imprecision, vagueness, fuzziness).

Different methodologies have been proposed to deal with them (e.g., approaches based on default logic, probability, fuzzy sets, etc.)

From equalities to proximities

For many problems in this area, exact equality is replaced by its approximation.

Tolerance relations are a tool to express the approximation, modeling the corresponding imprecise information.

They are reflexive and symmetric but not necessarily transitive relations, expressing the idea of closeness or resemblance.

From equalities to proximities

For many problems in this area, exact equality is replaced by its approximation.

Tolerance relations are a tool to express the approximation, modeling the corresponding imprecise information.

They are reflexive and symmetric but not necessarily transitive relations, expressing the idea of closeness or resemblance.

In the original version, tolerance relations were crisp
(two objects are either close to each other or not).

Later, their graded counterparts appeared which led, among others, to fuzzy tolerance relations, called proximity relations.

From equalities to proximities

$$\begin{tikzcd}
\text{crisp tolerance} \arrow[r] & \boxed{\text{proximity (fuzzy tolerance)}} \\
\text{crisp equivalence} \arrow[u] \arrow[r] & \text{similarity (fuzzy equivalence)} \arrow[u]
\end{tikzcd}$$

Fuzzy tolerances and equivalences

A fuzzy relation on a set S: a mapping from S to $[0, 1]$.

Fuzzy tolerances and equivalences

A fuzzy relation on a set S: a mapping from S to $[0, 1]$.

A fuzzy relation $\mathcal{R}$ on S is a proximity (fuzzy tolerance) relation on S iff 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 $s_1, s_2 \in S$.

Fuzzy tolerances and equivalences

A fuzzy relation on a set S: a mapping from S to $[0,1]$.

A fuzzy relation $\mathcal{R}$ on S is a proximity (fuzzy tolerance) relation on S iff 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 $s_1, s_2 \in S$.

A proximity relation on S is a similarity (fuzzy equivalence) relation on S if it is transitive:

$$\mathcal{R}(s_1,s_2) \geq \mathcal{R}(s_1,s) \wedge \mathcal{R}(s,s_2) \text{ for any } s_1, s_2, s \in S,$$

where $\wedge$ is a T-norm: an associative, commutative, non-decreasing binary operation on $[0,1]$ with 1 as the unit element.

In this talk: T-norm is $\min$.

Fuzzy tolerances and equivalences

Given $0 \leq \lambda \leq 1$, the λ-cut of $\mathcal{R}$ on S is the crisp relation

$$\mathcal{R}_\lambda := \{(s_1, s_2) \mid \mathcal{R}(s_1, s_2) \geq \lambda\}.$$

Fuzzy tolerances and equivalences

Given $0 \leq \lambda \leq 1$, the λ-cut of $\mathcal{R}$ on S is the crisp relation

$$\mathcal{R}_\lambda := \{(s_1, s_2) \mid \mathcal{R}(s_1, s_2) \geq \lambda\}.$$

λ-cut of a proximity relation is a crisp tolerance relation.

λ-cut of a similarity relation is a crisp equivalence relation.

$$\begin{tikzcd}
\text{crisp tolerance} \arrow[r] & \text{proximity (fuzzy tolerance)} \\
\text{crisp equivalence} \arrow[u] \arrow[r] & \text{similarity (fuzzy equivalence)} \arrow[u]
\end{tikzcd}$$

Proximity between terms: a restricted case

Only function symbols of the same arity are allowed to be proximal.

Mismatch is allowed in the name, not in the arity.

Proximity is extended between terms:

$\mathcal{R}(x, x) = 1$ for any x.

$$\mathcal{R}(f(t_1, \ldots, t_n),\ g(s_1, \ldots, s_n)) = \\ \min\{\mathcal{R}(f, g),\ \mathcal{R}(t_1, s_1), \ldots, \mathcal{R}(t_n, s_n)\},\ \ n \geq 0.$$

Otherwise, $\mathcal{R}(t, s) = 0$.

Proximity-based unification

The idea:

- $f(x)$ and $g(a)$ are not syntactically unifiable.
- However, if f and g are sufficiently close to each other ($(f, g) \in \mathcal{R}_\lambda$), then $\{x \mapsto a\}$ is an approximate unifier of $f(x)$ and $g(a)$ with respect $\mathcal{R}$ and λ.

Proximity-based unification

$(\mathcal{R}, \lambda)$-unifying two terms t_1 and t_2:

Proximity-based unification

$(\mathcal{R},\lambda)$-unifying two terms t_1 and t_2:

$$\begin{tikzcd}
t_1 \arrow[dr, dash, "\vartheta"'] & & & t_2 \arrow[dl, dash, "\vartheta"] \\
& t_1\vartheta \arrow[r, phantom, "\simeq_{\mathcal{R},\lambda}"] & t_2\vartheta &
\end{tikzcd}$$

ϑ: a most general $(\mathcal{R},\lambda)$-unifier of t_1 and t_2.

ϑ solves the approximate unification problem $t_1 \simeq^?_{\mathcal{R},\lambda} t_2$.

Block-based and class-based approaches

Two approaches to proximity-based unification.

Proximity relation: undirected graph.

block-based vs class-based

Block-based and class-based approaches

Two approaches to proximity-based unification.

Proximity relation: undirected graph.

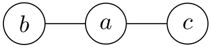
block-based	vs	class-based
block of a:		class of a:
max. clique to which a belongs		the neighborhood of a

Block-based and class-based approaches

Two approaches to proximity-based unification.

Proximity relation: undirected graph.

block-based	vs	class-based
block of a:		class of a:
max. clique to which a belongs		the neighborhood of a

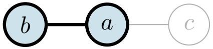


Block-based and class-based approaches

Two approaches to proximity-based unification.

Proximity relation: undirected graph.

block-based	vs	class-based
block of a:		class of a:
max. clique to which a belongs		the neighborhood of a

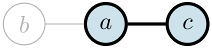


Block-based and class-based approaches

Two approaches to proximity-based unification.

Proximity relation: undirected graph.

block-based	vs	class-based
block of a:		class of a:
max. clique to which a belongs		the neighborhood of a

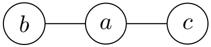


Block-based and class-based approaches

Two approaches to proximity-based unification.

Proximity relation: undirected graph.

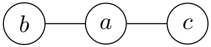
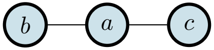
block-based	vs	class-based
block of a:		class of a:
max. clique to which a belongs		the neighborhood of a



Block-based and class-based approaches

Two approaches to proximity-based unification.

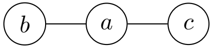
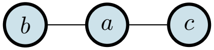
Proximity relation: undirected graph.

block-based	vs	class-based
block of a: max. clique to which a belongs		class of a: the neighborhood of a
		

Block-based and class-based approaches

Two approaches to proximity-based unification.

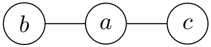
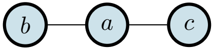
Proximity relation: undirected graph.

block-based	vs	class-based
block of a: max. clique to which a belongs		class of a: the neighborhood of a
		
$f(x,x) \simeq^?_{\mathcal{R},\lambda} f(b,c)$ not solvable		$f(x,x) \simeq^?_{\mathcal{R},\lambda} f(b,c)$ solved by $\{x \mapsto a\}$

Block-based and class-based approaches

Two approaches to proximity-based unification.

Proximity relation: undirected graph.

block-based	vs	class-based
block of a:		class of a:
max. clique to which a belongs		the neighborhood of a
		
$f(x,x) \simeq^?_{\mathcal{R},\lambda} f(b,c)$		$f(x,x) \simeq^?_{\mathcal{R},\lambda} f(b,c)$
not solvable		solved by $\{x \mapsto a\}$

We follow the class-based approach.

Peculiarities of proximity-based unification

These syntactic unification problems

$$\{f(x,y) \doteq^? f(y,b)\} \text{ and } \{f(x,y) \doteq^? f(b,b)\}$$

have the same solutions.

In proximity-based unification they have different solutions as well.

Peculiarities of proximity-based unification

Take $\mathcal{R}_\lambda = \{(a,b),(b,c),(c,d)\}$ and the problems

$$P_1 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(y,b)\}, \quad P_2 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(b,b)\}.$$

Peculiarities of proximity-based unification

Take $\mathcal{R}_\lambda = \{(a,b),(b,c),(c,d)\}$ and the problems

$$P_1 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(y,b)\}, \quad P_2 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(b,b)\}.$$

Let $\sigma = \{x \mapsto d, y \mapsto c\}$ and $\vartheta = \{x \mapsto a, y \mapsto c\}$.

Peculiarities of proximity-based unification

Take $\mathcal{R}_\lambda = \{(a,b),(b,c),(c,d)\}$ and the problems

$$P_1 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(y,b)\}, \quad P_2 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(b,b)\}.$$

Let $\sigma = \{x \mapsto d, y \mapsto c\}$ and $\vartheta = \{x \mapsto a, y \mapsto c\}$.

- σ is a unifier of P_1: $f(d,c) \simeq_{\mathcal{R},\lambda} f(c,b)$.

Peculiarities of proximity-based unification

Take $\mathcal{R}_\lambda = \{(a,b),(b,c),(c,d)\}$ and the problems

$$P_1 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(y,b)\}, \quad P_2 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(b,b)\}.$$

Let $\sigma = \{x \mapsto d, y \mapsto c\}$ and $\vartheta = \{x \mapsto a, y \mapsto c\}$.

- σ is a unifier of P_1: $f(d,c) \simeq_{\mathcal{R},\lambda} f(c,b)$.

  But σ is not a unifier of P_2: $f(d,c) \not\simeq_{\mathcal{R},\lambda} f(b,b)$.

Peculiarities of proximity-based unification

Take $\mathcal{R}_\lambda = \{(a,b),(b,c),(c,d)\}$ and the problems

$$P_1 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(y,b)\}, \quad P_2 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(b,b)\}.$$

Let $\sigma = \{x \mapsto d, y \mapsto c\}$ and $\vartheta = \{x \mapsto a, y \mapsto c\}$.

- σ is a unifier of P_1: $f(d,c) \simeq_{\mathcal{R},\lambda} f(c,b)$.

  But σ is not a unifier of P_2: $f(d,c) \not\simeq_{\mathcal{R},\lambda} f(b,b)$.

- ϑ is not a unifier of P_1: $f(a,c) \not\simeq_{\mathcal{R},\lambda} f(c,b)$.

Peculiarities of proximity-based unification

Take $\mathcal{R}_\lambda = \{(a,b),(b,c),(c,d)\}$ and the problems

$$P_1 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(y,b)\}, \quad P_2 = \{f(x,y) \simeq^?_{\mathcal{R},\lambda} f(b,b)\}.$$

Let $\sigma = \{x \mapsto d, y \mapsto c\}$ and $\vartheta = \{x \mapsto a, y \mapsto c\}$.

- σ is a unifier of P_1: $f(d,c) \simeq_{\mathcal{R},\lambda} f(c,b)$.

  But σ is not a unifier of P_2: $f(d,c) \not\simeq_{\mathcal{R},\lambda} f(b,b)$.

- ϑ is not a unifier of P_1: $f(a,c) \not\simeq_{\mathcal{R},\lambda} f(c,b)$.

  But ϑ is a unifier of P_2: $f(a,c) \simeq_{\mathcal{R},\lambda} f(b,b)$.

Proximity between terms: the general case

Fully fuzzy signatures: proximity between function symbols of different arities are allowed.

Mismatch is possible both in the names and the arities.

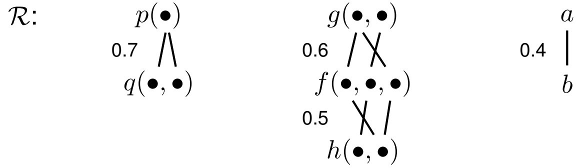
Proximity relations are defined on function symbols together with the proximity between their arguments.

Proximity between terms: the general case

Fully fuzzy signatures: proximity between function symbols of different arities are allowed.

Mismatch is possible both in the names and the arities.

Proximity relations are defined on function symbols together with the proximity between their arguments.

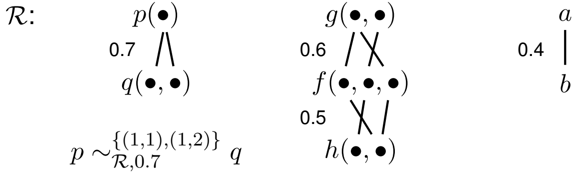


Proximity between terms: the general case

Fully fuzzy signatures: proximity between function symbols of different arities are allowed.

Mismatch is possible both in the names and the arities.

Proximity relations are defined on function symbols together with the proximity between their arguments.

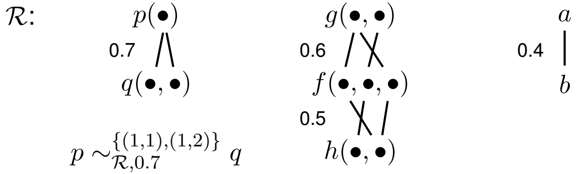


Proximity between terms: the general case

Fully fuzzy signatures: proximity between function symbols of different arities are allowed.

Mismatch is possible both in the names and the arities.

Proximity relations are defined on function symbols together with the proximity between their arguments.



$$p \sim_{\mathcal{R},0.7}^{\{(1,1),(1,2)\}} q$$

We have $f \sim_{\mathcal{R},1}^{Id} f$ for all f.

Proximity between terms: the general case

Proximity over the signature is extended to proximity over terms:

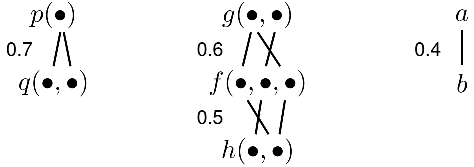
$\mathcal{R}(x,x) = 1$ for all variables x.

$$\mathcal{R}(f(t_1,\dots,t_n), g(s_1,\dots,s_m)) = \min\{\mathcal{R}(f,g), \min_{(i,j)\in\rho}\{\mathcal{R}(t_i,s_j)\}\}, \text{ where } f \sim^{\rho}_{\mathcal{R},\lambda} g$$

$\mathcal{R}(t,s) = 0$ in all other cases.

Unification problem

$\mathcal{R}$:

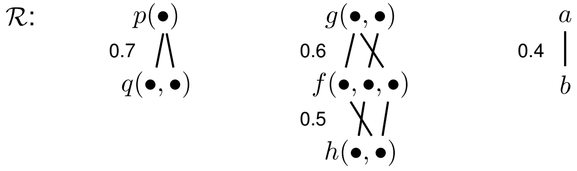


Unification problem: $p(x) \simeq^?_{\mathcal{R},0.4} q(g(y,a),h(z,y))$.

Find a substitution ϑ such that

$$\mathcal{R}(p(x)\vartheta,\ q(g(y,a),h(z,y))\vartheta) \geq 0.4.$$

Unification problem

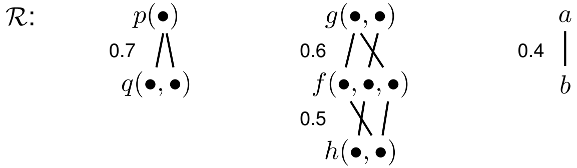


Unification problem: $P = \{p(x) \simeq^?_{\mathcal{R},0.4} q(g(y,a),h(z,y))\}$.

One of the computed answers:

$$\{v \simeq_{\mathcal{R},0.4} y,\ u \simeq_{\mathcal{R},0.4} y\};\ \{x \mapsto f(v,a,u), z \mapsto a\};\ 0.5$$

Unification problem



Unification problem: $P = \{p(x) \simeq^{?}_{\mathcal{R},0.4} q(g(y,a),h(z,y))\}$.

One of the computed answers:

$$\{v \simeq_{\mathcal{R},0.4} y,\ u \simeq_{\mathcal{R},0.4} y\};\ \{x \mapsto f(v,a,u), z \mapsto a\};\ 0.5$$

Any solution of $\{v \simeq_{\mathcal{R},0.4} y,\ u \simeq_{\mathcal{R},0.4} y\}$ unifies

$p(x)\{x \mapsto f(v,a,u), z \mapsto a\} = p(f(v,a,u))$ and

$q(g(y,a),h(z,y))\{x \mapsto f(v,a,u), z \mapsto a\} = q(g(y,a),h(a,y))$.

Unifiability

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

Unifiability

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

In fact, already a special case (well-moded) is NP-hard.

Unification algorithm: the idea

When solving the unification constraint

$$\{f(t_1, \dots, t_n) \simeq^?_{\mathcal{R},\lambda} g(s_1, \dots, s_m)\} \uplus P :$$

- Check whether $f \sim^{\rho}_{\mathcal{R},\beta} g$ with $\beta \geq \lambda$.
- If yes, try to solve $\{t_i \simeq^?_{\mathcal{R},\lambda} s_j \mid (i, j) \in \rho\} \cup P$.
- Otherwise fail.

If the approximation degree computed so far is α, update it with $\alpha \wedge \beta$.

Unification algorithm: the idea

When solving the unification constraint

$$\{x \simeq^{?}_{\mathcal{R},\lambda} g(s_1, \ldots, s_m)\} \uplus P :$$

Unification algorithm: the idea

When solving the unification constraint

$$\{x \simeq^{?}_{\mathcal{R},\lambda} g(s_1, \ldots, s_m)\} \uplus P :$$

- If there is an occurrence cycle for x in the problem, fail.

Unification algorithm: the idea

When solving the unification constraint

$$\{x \simeq^?_{\mathcal{R},\lambda} g(s_1, \ldots, s_m)\} \uplus P :$$

- If there is an occurrence cycle for x in the problem, fail.
- Otherwise, take $\sigma = \{x \mapsto f(y_1, \ldots, y_n)\}$ for an n-ary f with $f \sim^{\rho}_{\mathcal{R},\beta} g$ and $\beta \geq \lambda$, where the y's are fresh variables.

Unification algorithm: the idea

When solving the unification constraint

$$\{x \simeq^?_{\mathcal{R},\lambda} g(s_1, \ldots, s_m)\} \uplus P :$$

- If there is an occurrence cycle for x in the problem, fail.
- Otherwise, take $\sigma = \{x \mapsto f(y_1, \ldots, y_n)\}$ for an n-ary f with $f \sim^{\rho}_{\mathcal{R},\beta} g$ and $\beta \geq \lambda$, where the y's are fresh variables.
- Try to solve $\{y_i \simeq^?_{\mathcal{R},\lambda} s_j \mid (i, j) \in \rho\} \cup P\sigma$.

Unification algorithm: the idea

When solving the unification constraint

$$\{x \simeq^{?}_{\mathcal{R},\lambda} g(s_1, \ldots, s_m)\} \uplus P :$$

- If there is an occurrence cycle for x in the problem, fail.
- Otherwise, take $\sigma = \{x \mapsto f(y_1, \ldots, y_n)\}$ for an n-ary f with $f \sim^{\rho}_{\mathcal{R},\beta} g$ and $\beta \geq \lambda$, where the y's are fresh variables.
- Try to solve $\{y_i \simeq^{?}_{\mathcal{R},\lambda} s_j \mid (i, j) \in \rho\} \cup P\sigma$.

If the approximation degree computed so far is α, update it with $\alpha \wedge \beta$.

Unification algorithm: the idea

To make sure that failing with occurrence cycles does not lead to losing a solution, we require that all argument relations are correspondence (i.e., left- and right-total) relations.

Correspondence relations guarantee that proximal terms have the same set of variables and no term is close to its proper subterm.

The unification algorithm is sound, complete, and terminating.

Final comments

Our argument correspondence relations can be represented as a version of regular, collapse-free, shallow theories, which have been studied quite intensively in (crisp) equational unification.

Our work opens a way towards studying approximate unification modulo background theories.

Final comments

Our argument correspondence relations can be represented as a version of regular, collapse-free, shallow theories, which have been studied quite intensively in (crisp) equational unification.

Our work opens a way towards studying approximate unification modulo background theories.

Paper:

- Cleo Pau and Temur Kutsia. Proximity-Based Unification and Matching for Fully Fuzzy Signatures. In: Proceedings of FUZZ-IEEE 2021 - 30th IEEE International Conference on Fuzzy Systems. IEEE 2021. 1–6.