SYMBOLIC CONSTRAINTS AND QUANTITATIVE EXTENSIONS OF EQUALITY

Temur Kutsia

April 25, 2023. FMAR seminar

JOHANNES KEPLER UNIVERSITY LINZ

Symbolic constraints

Usually: conjunctions of primitive (atomic) constraints in some logic language.

Examples of primitive constraints:

- equations,
- disequations,
- atomic formulas expressing e.g., ordering, membership, generalization, or dominance relations,
- etc.

Solutions: variable substitutions that satisfy the given formula.

Symbolic constraints

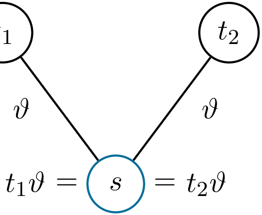
Our focus: equational and generalization constraints.

Solving methods: unification, matching, anti-unification.

Appear in many areas of computational logic:

- automated reasoning
- term rewriting
- declarative programming
- pattern-based calculi
- unification theory
- …

Dual problems: unification / anti-unification



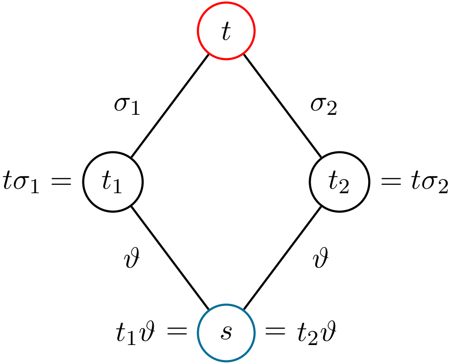
s: most general instance

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

Dual problems: unification / anti-unification

t: least general generalization

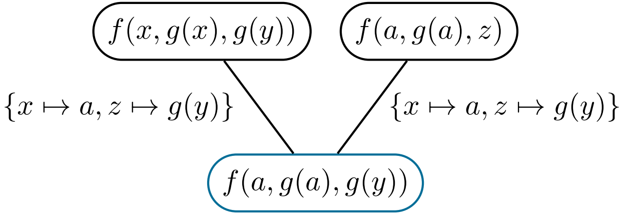
$X = t$ solves the anti-unification problem $X : t_1 \triangleq t_2$



s: most general instance

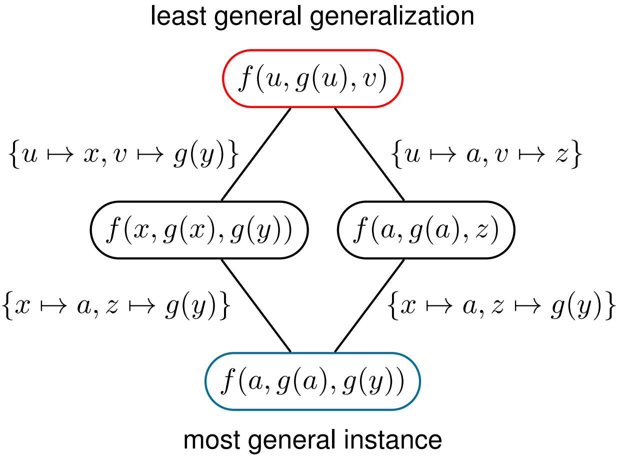
ϑ solves the unification problem $t_1 =^? t_2$

Dual problems: unification / anti-unification



most general instance

Dual problems: unification / anti-unification



Precise vs imprecise

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?

Outline

Quantitative extensions of equalities

Fuzzy proximities

Quantitative equational logic

Future research directions

Outline

Quantitative extensions of equalities

Fuzzy proximities

Quantitative equational logic

Future research directions

From equalities to tolerances

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

Its modeling is a highly nontrivial task.

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 tolerances

Examples of tolerance relations include some well-known mathematical notions, e.g.,

- a and b are vertices of the same edge in an undirected graph,

From equalities to tolerances

Examples of tolerance relations include some well-known mathematical notions, e.g.,

- a and b are vertices of the same edge in an undirected graph,
- a and b are points in a metric space that are within a given positive distance from each other,

From equalities to tolerances

Examples of tolerance relations include some well-known mathematical notions, e.g.,

- a and b are vertices of the same edge in an undirected graph,
- a and b are points in a metric space that are within a given positive distance from each other,
- Two binary sequences a and b differ from each other in at most e positions for some given error level e.

From equalities to tolerances

Examples of tolerance relations include some well-known mathematical notions, e.g.,

- a and b are vertices of the same edge in an undirected graph,
- a and b are points in a metric space that are within a given positive distance from each other,
- Two binary sequences a and b differ from each other in at most e positions for some given error level e.
- For a topological space T and its fixed covering ω, the relation "a and b are points in T that belong to the same element of ω".

From equalities to tolerances

Examples of approximating the equality by quantitative tolerance relations:

- Using fuzzy proximity relations, expressing the degree of closeness / resemblance:

  $t \simeq_\lambda s, \quad \lambda \in [0,1]:$

  t and s are proximal with degree λ

- Using quantitative equations, expressing the distance between the objects:

  $t \simeq_\lambda s, \quad \lambda \in \mathbb{Q}^{\geq 0}:$

  t and s are at most λ apart

Quantitative relations over terms

Our objects are first-order terms.

We need to define quantitative counterparts of equality for terms, and then design methods to solve symbolic constraints over them.

Outline

Quantitative extensions of equalities

Fuzzy proximities

Quantitative equational logic

Future research directions

Fuzzy proximities

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 $s_1, s_2 \in S$.

$\mathcal{R}(s_1,s_2)$: the degree of proximity between s_1 and s_2.

Fuzzy proximities

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 (monotonic) binary operation on $[0, 1]$ with 1 as the unit element.

Fuzzy proximities

T-norm (triangular norm) generalizes intersection in a lattice and conjunction in logic.

Some well-known T-norms:

- Minimum T-norm (aka Gödel T-norm): $s \wedge t = \min(s, t)$.
- Product T-norm: $s \wedge t = s * t$.
- Łukasiewicz T-norm: $s \wedge t = \max\{0, s + t - 1\}$.

In the rest, we use the $\min$ T-norm.

Fuzzy proximities

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

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

Notation: $s_1 \simeq_{\mathcal{R},\lambda} s_2$ means $(s_1, s_2) \in \mathcal{R}_\lambda$.

The cut value λ provides a threshold: defines which objects are treated proximal to each other ($(\mathcal{R}, \lambda)$-proximal) and which are not.

Fuzzy proximities

$\mathcal{R}$: a given proximity relation on a set of function symbols $\mathcal{F}$.

No restriction: symbols of different arity might be proximal with a positive degree (fully fuzzy signature).

To be able to extend proximity from alphabet symbols to terms, we need to know which arguments of proximal symbols are related to each other (argument relations).

We assume that this information is provided.

Fuzzy proximities

If $\mathcal{R}(f, g) = \alpha > 0$ and the argument relation between f and g is ρ, we write $f \sim^{\rho}_{\mathcal{R},\alpha} g$.

Assumptions:

- for each pair (f, g), there is at most one argument relation;
- if $f \sim^{\rho}_{\mathcal{R},\alpha} g$, then $g \sim^{\rho^{-1}}_{\mathcal{R},\alpha} f$.

Fuzzy proximities

If $\mathcal{R}(f, g) = \alpha > 0$ and the argument relation between f and g is ρ, we write $f \sim^{\rho}_{\mathcal{R},\alpha} g$.

Assumptions:

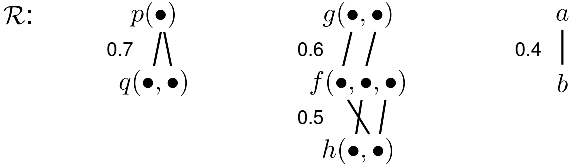
- for each pair (f, g), there is at most one argument relation;
- if $f \sim^{\rho}_{\mathcal{R},\alpha} g$, then $g \sim^{\rho^{-1}}_{\mathcal{R},\alpha} f$.

Basic signatures: a special case with ρ required to be a (left and right) total identity relation.

Argument relations should satisfy certain extra properties in order a similarity relation on the signature to be extendable to a similarity relation over terms.

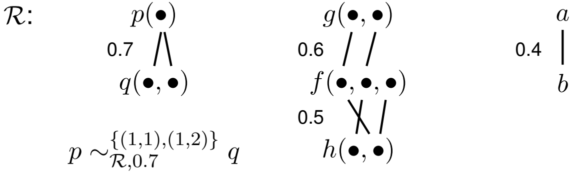
Fuzzy proximities

Example of a proximity relation on a fully fuzzy signature.



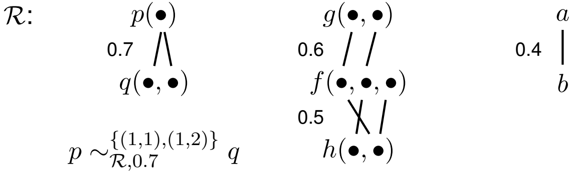
Fuzzy proximities

Example of a proximity relation on a fully fuzzy signature.



Fuzzy proximities

Example of a proximity relation on a fully fuzzy signature.



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

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

Fuzzy proximities over terms

Extending $\mathcal{R}$ from the signature to terms:

- $\mathcal{R}(x, x) = 1$ for all variables x.
- $\mathcal{R}(f(t_1, \ldots, t_n), g(s_1, \ldots, s_m)) = \alpha \wedge \bigwedge_{(i,j)\in\rho} \mathcal{R}(t_i, s_j)$, where $f \sim^{\rho}_{\mathcal{R},\alpha} g$.
- $\mathcal{R}(t, s) = 0$ in all other cases.

Fuzzy proximities over terms

Extending $\mathcal{R}$ from the signature to terms:

- $\mathcal{R}(x, x) = 1$ for all variables x.
- $\mathcal{R}(f(t_1, \ldots, t_n), g(s_1, \ldots, s_m)) = \alpha \wedge \bigwedge_{(i,j)\in\rho} \mathcal{R}(t_i, s_j)$, where $f \sim^{\rho}_{\mathcal{R},\alpha} g$.
- $\mathcal{R}(t, s) = 0$ in all other cases.

Such an extension is a proximity relation on terms.

$$\mathcal{R}(t, t) = 1 \qquad \mathcal{R}(s, t) = \mathcal{R}(t, s)$$

$$\mathcal{R}(t, s) \leq \mathcal{R}(t\sigma, s\sigma) \qquad \mathcal{R}(C[t], C[s]) = \mathcal{R}(t, s)$$

Proximity-based unification

Given: A proximity relation $\mathcal{R}$, a cut value λ, and term pairs (t_i, s_i), $1 \leq i \leq n$.

Find: A substitution σ such that $t_i\sigma \simeq_{\mathcal{R},\lambda} s_i\sigma$ for all $1 \leq i \leq n$.

Proximity-based unification

Given: A proximity relation $\mathcal{R}$, a cut value λ, and term pairs (t_i, s_i), $1 \leq i \leq n$.

Find: A substitution σ such that $t_i\sigma \simeq_{\mathcal{R},\lambda} s_i\sigma$ for all $1 \leq i \leq n$.

$(\mathcal{R}, \lambda)$-unification problem: $P = \{t_1 \simeq^?_{\mathcal{R},\lambda} s_1, \ldots, t_n \simeq^?_{\mathcal{R},\lambda} s_n\}$.

σ: $(\mathcal{R}, \lambda)$-unifier of P.

Interesting unifiers are most general ones.

Proximity-based matching

Given: A proximity relation $\mathcal{R}$, a cut value λ, and term pairs (t_i, s_i), $1 \leq i \leq n$.

Find: A substitution σ such that $t_i\sigma \simeq_{\mathcal{R},\lambda} s_i$ for all $1 \leq i \leq n$.

Proximity-based matching

Given: A proximity relation $\mathcal{R}$, a cut value λ, and term pairs (t_i, s_i), $1 \leq i \leq n$.

Find: A substitution σ such that $t_i\sigma \simeq_{\mathcal{R},\lambda} s_i$ for all $1 \leq i \leq n$.

$(\mathcal{R}, \lambda)$-matching problem: $P = \{t_1 \precsim^?_{\mathcal{R},\lambda} s_1, \ldots, t_n \precsim^?_{\mathcal{R},\lambda} s_n\}$.

σ: $(\mathcal{R}, \lambda)$-matcher of P.

Can be treated as a special case of unification.

Better: use a simpler dedicated algorithm.

Proximity-based generalization

Given: A proximity relation $\mathcal{R}$, a cut value λ, and two terms t and s.

Find: A term r such that $r \precsim_{\mathcal{R},\lambda} t$ and $r \precsim_{\mathcal{R},\lambda} s$.

Proximity-based generalization

Given: A proximity relation $\mathcal{R}$, a cut value λ, and two terms t and s.

Find: A term r such that $r \precsim_{\mathcal{R},\lambda} t$ and $r \precsim_{\mathcal{R},\lambda} s$.

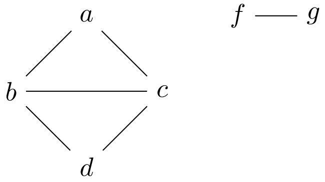
$t \triangleq_{\mathcal{R},\lambda} s$: the notation for t and s to be generalized.

r: $(\mathcal{R}, \lambda)$-generalization of s and t.

Interesting generalizations are the least general ones.

Proximity classes

$\mathcal{R}_\lambda$:



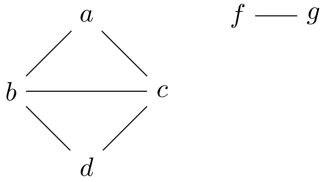
In the class-based approach, the terms $f(x, x)$ and $g(a, d)$ are unifiable.

Reason: a and d have common neighbors, b and c.

It is natural to have $\{x \mapsto b\}$ and $\{x \mapsto c\}$ as unifiers of $f(x, x)$ and $g(a, d)$.

Proximity classes

$\mathcal{R}_\lambda$:



In the class-based approach, the terms $f(x, x)$ and $g(a, d)$ are unifiable.

Reason: a and d have common neighbors, b and c.

It is natural to have $\{x \mapsto b\}$ and $\{x \mapsto c\}$ as unifiers of $f(x, x)$ and $g(a, d)$.

Proximity class of a symbol: its neighborhood in the graph.

class$(a, \mathcal{R}, \lambda) = \{a, b, c\}$. class$(d, \mathcal{R}, \lambda) = \{d, b, c\}$.

Proximity-based unification using classes

One of the peculiarities:

Syntactic unification problems

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

have the same set of unifiers.

In proximity-based unification with classes this is not the case.

Proximity-based unification using classes

One of the peculiarities:

Syntactic unification problems

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

have the same set of unifiers.

In proximity-based unification with classes this is not the case.

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

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

Proximity-based unification using classes

One of the peculiarities:

Syntactic unification problems

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

have the same set of unifiers.

In proximity-based unification with classes this is not the case.

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

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

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

Proximity-based unification using classes

One of the peculiarities:

Syntactic unification problems

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

have the same set of unifiers.

In proximity-based unification with classes this is not the case.

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

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

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

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

Proximity-based unification using classes

One of the peculiarities:

Syntactic unification problems

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

have the same set of unifiers.

In proximity-based unification with classes this is not the case.

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

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

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: $f(d,c) \not\simeq_{\mathcal{R},\lambda} f(b,b)$.

Proximity-based unification using classes

One of the peculiarities:

Syntactic unification problems

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

have the same set of unifiers.

In proximity-based unification with classes this is not the case.

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

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

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

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

Proximity-based unification using classes

One of the peculiarities:

Syntactic unification problems

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

have the same set of unifiers.

In proximity-based unification with classes this is not the case.

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

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

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

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: $f(a,c) \simeq_{\mathcal{R},\lambda} f(b,b)$.

Unification using classes, fully fuzzy

The algorithm works for argument relations $\rho \subseteq N \times M$ that are correspondence relations, i.e. they are:

- left-total
  for all $i \in N$ there exists $j \in M$ such that $(i, j) \in \rho$;
- right-total
  for all $j \in M$ there exists $i \in N$ such that $(i, j) \in \rho$.

Unification using classes, fully fuzzy

The algorithm works for argument relations $\rho \subseteq N \times M$ that are correspondence relations, i.e. they are:

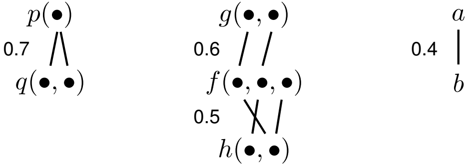
- left-total
  for all $i \in N$ there exists $j \in M$ such that $(i, j) \in \rho$;
- right-total
  for all $j \in M$ there exists $i \in N$ such that $(i, j) \in \rho$.

This is to make sure that failing with occurrence cycles does not lead to losing a solution.

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

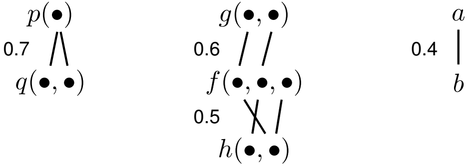
Unification using classes, fully fuzzy

The argument relation in this example is not correspondence:

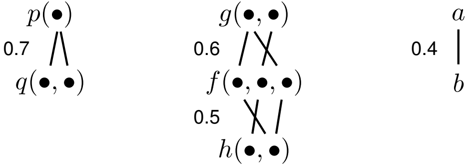


Unification using classes, fully fuzzy

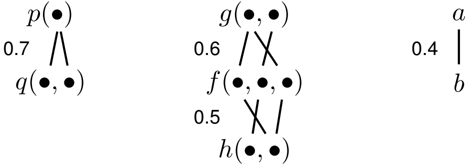
The argument relation in this example is not correspondence:



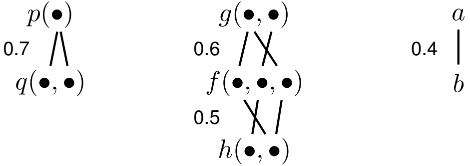
Here it is:



Unification using classes, fully fuzzy



Unification using classes, fully fuzzy



Unification problem: $P = \{p(x) \simeq^?_{\mathcal{R},0.3} q(g(u,a), h(z,u))\}$.

For P, the algorithm produces four final configurations:

$$\{v_1 \simeq^?_{\mathcal{R},0.3} u,\ \ v_3 \simeq^?_{\mathcal{R},0.3} u\}; \quad \{v_1 \simeq^?_{\mathcal{R},0.3} u,\ \ v_3 \simeq^?_{\mathcal{R},0.3} u\};$$

$$\{x \mapsto f(v_1, a, v_3), z \mapsto a\};\ 0.5 \quad \{x \mapsto f(v_1, b, v_3), z \mapsto a\};\ 0.4$$

$$\{v_1 \simeq^?_{\mathcal{R},0.3} u,\ \ v_3 \simeq^?_{\mathcal{R},0.3} u\}; \quad \{v_1 \simeq^?_{\mathcal{R},0.3} u,\ \ v_3 \simeq^?_{\mathcal{R},0.3} u\};$$

$$\{x \mapsto f(v_1, a, v_3), z \mapsto b\};\ 0.4 \quad \{x \mapsto f(v_1, b, v_3), z \mapsto b\};\ 0.5$$

Unifiability

The decision problem of class-based approximate unifiability with in fully fuzzy signatures is NP-hard.

It can be shown by a reduction from positive 1-in-3-SAT problem.

Unifiability

The decision problem of class-based approximate unifiability with in fully fuzzy signatures is NP-hard.

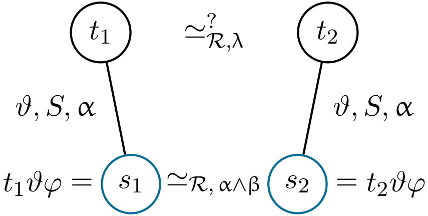
It can be shown by a reduction from positive 1-in-3-SAT problem.

In fact, the reduction shows that already a special case of unifiability (well-moded) is NP-hard.

Unification using classes, fully fuzzy

$$\textcircled{t_1} \quad \simeq^?_{\mathcal{R},\lambda} \quad \textcircled{t_2}$$

Unification using classes, fully fuzzy



If φ solves the variable-only constraint S with degree β then $\vartheta\varphi$ solves the unification problem $t_1 \simeq^?_{\mathcal{R},\lambda} t_2$ with degree $\alpha \wedge \beta$

Matching using classes, fully fuzzy

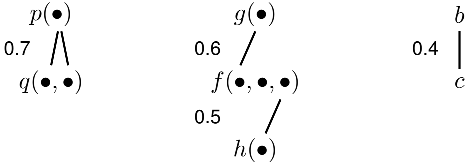
Unlike unification, we do not have to restrict argument relations for matching.

It may cause matchers to contain fresh variables.

Matching using classes, fully fuzzy

Unlike unification, we do not have to restrict argument relations for matching.

It may cause matchers to contain fresh variables.



Consider the matching problem $p(x) \precsim^{?}_{\mathcal{R},0.4} q(g(a), h(c))$.

The matching algorithm returns two solutions:

$$\{x \mapsto f(a, v, c)\}; 0.5 \qquad\qquad \{x \mapsto f(a, v, b)\}; 0.4$$

where v is a fresh variable.

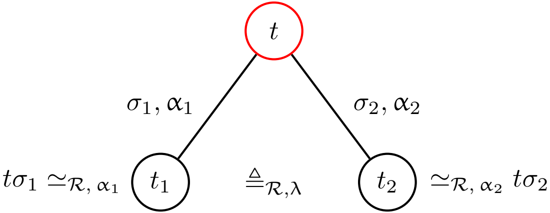
Generalization using classes, fully fuzzy

We compute t, α_1, α_2, and a representation from which σ_1 and σ_2 can be read.

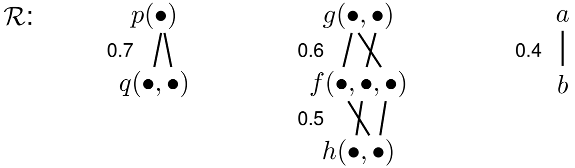
t: a least general generalization

$X = t$ solves the anti-unification problem $X : t_1 \triangleq_{\mathcal{R},\lambda} t_2$

with degrees α_1 and α_2



Generalization using classes, fully fuzzy

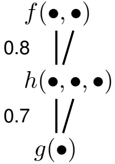


Given $\mathcal{R}$ and $\lambda = 0.3$, anti-unify $g(a, b)$ and $h(c, b)$.

One of the solutions: $f(a, x, a)$, where $x : b \triangleq c$, with the approximation degrees 0.6 for $g(a, b)$ and 0.4 for $h(c, b)$.

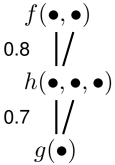
Generalization using classes, fully fuzzy

- $f \sim_{\mathcal{R},0.8}^{\{(1,1),(2,1)\}} h$.
- $h \sim_{\mathcal{R},0.7}^{\{(1,1),(2,1)\}} g$.
- $a \sim_{\mathcal{R},0.6}^{\emptyset} b$, $\quad b \sim_{\mathcal{R},0.5}^{\emptyset} c$



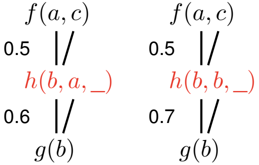
Generalization using classes, fully fuzzy

- $f \sim_{\mathcal{R},0.8}^{\{(1,1),(2,1)\}} h$.
- $h \sim_{\mathcal{R},0.7}^{\{(1,1),(2,1)\}} g$.
- $a \sim_{\mathcal{R},0.6}^{\emptyset} b$, $\quad b \sim_{\mathcal{R},0.5}^{\emptyset} c$



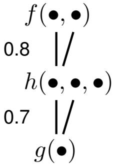
$(\mathcal{R},0.5)$-lggs of $f(a,c)$ and $g(b)$:
$h(b,a,\_)$ and $h(b,b,\_)$.

- lgg's can be comparable wrt $\precsim_{\mathcal{R},\lambda}$ (but not wrt $\preceq$),
- the irrelevant generalization argument is expressed by the anonymous variable _.



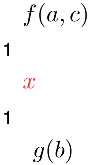
Generalization using classes, fully fuzzy

- $f \sim_{\mathcal{R},0.8}^{\{(1,1),(2,1)\}} h$.
- $h \sim_{\mathcal{R},0.7}^{\{(1,1),(2,1)\}} g$.
- $a \sim_{\mathcal{R},0.6}^{\emptyset} b$, $\;b \sim_{\mathcal{R},0.5}^{\emptyset} c$



$(\mathcal{R}, 0.6)$-lgg of $f(a,c)$ and $g(b)$: x.

- It can not be $h(y, b, \_)$, because y can not be instantiated by a term that is $(\mathcal{R}, 0.6)$-close to both a and c.
- The set $\{a, c\}$ is $(\mathcal{R}, 0.6)$-inconsistent



Family of algorithms

Some features of class-based fully fuzzy anti-unification:

- nonstandard variable merging (also in basic signatures)

- irrelevant position abstraction

- look-ahead consistency check of arguments

Family of algorithms

Some features of class-based fully fuzzy anti-unification:

- nonstandard variable merging (also in basic signatures)
  Not needed for linear generalizations
- irrelevant position abstraction
- look-ahead consistency check of arguments

Family of algorithms

Some features of class-based fully fuzzy anti-unification:

- nonstandard variable merging (also in basic signatures)
  Not needed for linear generalizations
- irrelevant position abstraction
  Not needed if argument relations are left- and right-total
- look-ahead consistency check of arguments

Family of algorithms

Some features of class-based fully fuzzy anti-unification:

- nonstandard variable merging (also in basic signatures)
  Not needed for linear generalizations
- irrelevant position abstraction
  Not needed if argument relations are left- and right-total
- look-ahead consistency check of arguments
  Not needed if argument relations are (partial) injective functions

Family of algorithms

Some features of class-based fully fuzzy anti-unification:

- nonstandard variable merging (also in basic signatures)
  Not needed for linear generalizations
- irrelevant position abstraction
  Not needed if argument relations are left- and right-total
- look-ahead consistency check of arguments
  Not needed if argument relations are (partial) injective functions

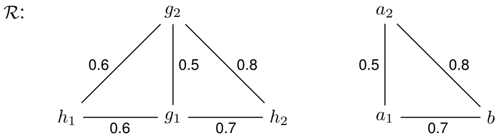
Combinations lead to eight different algorithms, obtained from the general set of rules in a modular way.

They differ from each other by the decomposition rule.

Each of them computes the respective minimal complete sets of generalizations, together with their approximation degree upper bounds.

Things are easier in basic signatures

A set-based compact representation is a convenient notation for formulating a matching algorithm for basic signatures.



$$\{f(x,x) \preceq^?_{\mathcal{R},0.6} f(g_1(a_1), g_2(a_2))\};\ \emptyset \Longrightarrow$$

$$\{x \preceq^?_{\mathcal{R},0.6} g_1(a_1),\ x \preceq^?_{\mathcal{R},0.6} g_2(a_2)\};\ \emptyset \Longrightarrow$$

$$\{x \preceq^?_{\mathcal{R},0.6} g_2(a_2)\};$$

$$\{x \approx \{(g_1,1),(h_1,0.6),(h_2,0.7)\}(\{(a_1,1),(b,0.7)\})\} \Longrightarrow$$

$$\emptyset;\ \{x \approx \{\{(g_1,1),(h_1,0.6),(h_2,0.7)\}(\{(a_1,1),(b,0.7)\}),$$

$$x \approx \{(g_2,1),(h_1,0.6),(h_2,0.8)\}(\{(a_2,1),(b,0.8)\})\} \Longrightarrow$$

$$\emptyset;\ \{x \approx \{(h_1,0.6),(h_2,0.7)\}(\{(b,0.7)\})\}.$$

Representing two solutions: $h_1(b); 0.6$ and $h_2(b); 0.7$.

Outline

Quantitative eq. logic: inference rules

$$\frac{s \approx_\lambda t \in E}{E \vdash s \approx_\lambda t}(\text{Ax.}) \qquad \frac{}{E \vdash s \approx_0 s}(\text{Refl.}) \qquad \frac{E \vdash s \approx_\lambda t}{E \vdash t \approx_\lambda s}(\text{Sym.})$$

$$\frac{E \vdash s \approx_\lambda t \quad E \vdash t \approx_\delta r}{E \vdash s \approx_{\lambda+\delta} r}(\text{Triang.}) \qquad \frac{E \vdash s \approx_\lambda t}{E \vdash s\sigma \approx_\lambda t\sigma}(\text{Inst.})$$

$$\frac{E \vdash s_1 \approx_\lambda t_1 \quad \cdots \quad E \vdash s_n \approx_\lambda t_n}{E \vdash f(s_1, \ldots, s_n) \approx_\lambda f(t_1, \ldots, t_n)}(\text{NonExp.})$$

$$\frac{E \vdash \phi \text{ for all } \phi \in E' \quad E' \vdash \psi}{E \vdash \psi}(\text{Cut.})$$

$$\frac{\delta > 0 \quad E \vdash s \approx_\lambda t}{E \vdash s \approx_{\lambda+\delta} t}(\text{Max.}) \qquad \frac{\{E \vdash s \approx_\delta t \mid \delta > \lambda\}}{E \vdash s \approx_\lambda t}(\text{Arch.})$$

Outline

Quantitative extensions of equalities

Fuzzy proximities

Quantitative equational logic

Future research directions

Directions for future research

- Generic treatment of T-norms.
- In the proximity setting, computing a best solution (by some criterion), instead of all solutions or some arbitrarily chosen ones ($\longrightarrow$ optimization?).
- Proximity-based unification, matching, and anti-unification modulo background theories (similar to crisp equational unification /matching / anti-unification).

Directions for future research

- Generic treatment of T-norms.
- In the proximity setting, computing a best solution (by some criterion), instead of all solutions or some arbitrarily chosen ones ($\longrightarrow$ optimization?).
- Proximity-based unification, matching, and anti-unification modulo background theories (similar to crisp equational unification /matching / anti-unification).
- Unification, matching, and anti-unification for quantitative theories.
- Relating to a recently introduced framework of quantitative and metric rewriting (Gavazzo & del Florio, POPL'23): completion.

Directions for future research

- Generic treatment of T-norms.
- In the proximity setting, computing a best solution (by some criterion), instead of all solutions or some arbitrarily chosen ones ($\longrightarrow$ optimization?).
- Proximity-based unification, matching, and anti-unification modulo background theories (similar to crisp equational unification /matching / anti-unification).
- Unification, matching, and anti-unification for quantitative theories.
- Relating to a recently introduced framework of quantitative and metric rewriting (Gavazzo & del Florio, POPL'23): completion.
- Applications.