Approximate Unification with Fuzzy Relations: A Survey

Bachelor Thesis

Paul-Gabriel Turcuman

May, 28 2024

Introduction

- Process of solving satisfiability problems:

  Given: A set of identities E and two terms s and t

  Find: A substitution σ with $\sigma(s) \approx_E \sigma(t)$

- In syntactic unification: $E = \emptyset$

  Given: Two terms s and t

  Find: A substitution σ with $\sigma(s) = \sigma(t)$

- Basis of logic programming
- Used in program transformation
- etc.

Introduction

Classic Unification

- Introduced by Robinson in 1965
- In the thesis the focus is on the rule-based algorithm
- (Pseudocode included only for reference)
- $\mathcal{V}ar(t)$ denotes the set of variables present in term t

Introduction

Algorithm

- **Classic-Trivial**: (C-Tri)
  $\{s =^? s\} \uplus P'; \sigma \Rightarrow P'; \sigma$, where s can be a variable or a constant.
- **Classic-Decomposition**: (C-Dec)
  $\{f(s_1, ..., s_n) =^? f(t_1, ..., t_n)\} \uplus P'; \sigma \Rightarrow \{s_1 =^? t_1, ..., s_n =^? t_n\} \cup P'; \sigma$, where $n \geq 0$.
- **Classic-Symbol Clash**: (C-SC)
  $\{f(s_1, ..., s_n) =^? g(t_1, ..., t_n)\} \uplus P'; \sigma \Rightarrow \bot$, if $f \neq g$.

Introduction

Algorithm

- **Classic-Orient**: (C-Or)
  $\{t =^? x\} \uplus P'; \sigma \Rightarrow \{x =^? t\} \cup P'; \sigma$, if $t \notin \mathcal{V}$.

- **Classic-Occurs Check**: (C-OC)
  $\{x =^? t\} \cup P'; \sigma \Rightarrow \bot$, if $x \in \mathcal{V}ar(t)$ but $x \neq t$.

- **Classic-Variable Elimination**: (C-VE)
  $\{x =^? t\} \cup P'; \sigma \Rightarrow P'\{x \mapsto t\}; \sigma\{x \mapsto t\} \cup \{x \mapsto t\}$, if $x \notin \mathcal{V}ar(t)$.

Example

- We want to unify $p(f(x), y)$ and $p(f(a), g(a))$.
- Applying the algorithm gives:
  $\{p(f(x), y) =^? p(f(a), g(a))\}; \emptyset \Rightarrow_{\text{C-Dec}}$
  $\{f(x) =^? f(a), y =^? g(a)\}; \emptyset \Rightarrow_{\text{C-VE}}$
  $\{f(x) =^? f(a)\}; \{y \mapsto g(a)\} \Rightarrow_{\text{C-Dec}}$
  $\{x =^? a\}; \{y \mapsto g(a)\} \Rightarrow_{\text{C-VE}}$
  $\emptyset; \{x \mapsto a, y \mapsto g(a)\}$.
- Substituion $\sigma = \{x \mapsto a, y \mapsto g(a)\}$ is a solution

Motivation

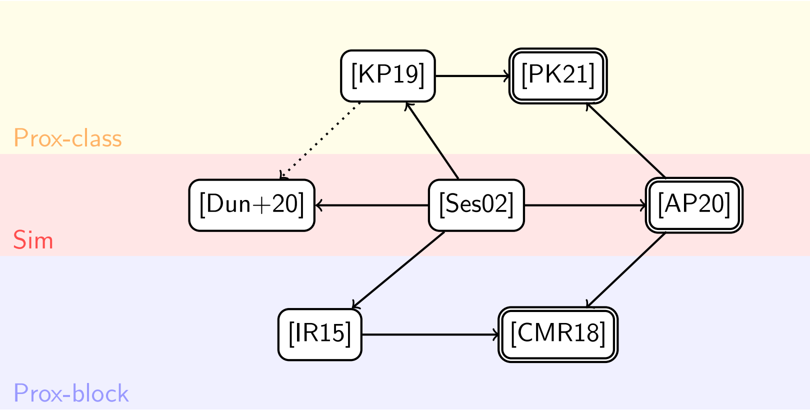
- With different symbols classical algorithm fails (eg. $p(x)$ with $q(a)$)
- Or different arities
- But we want to be able to unify such terms as well
- What to do?
- Introduce a fuzzy relation between the two symbols
- We need to define the notions of proximity and similarity for them

Motivation

Two kinds of signatures, depending how fuzzy relations are defined on the set of function symbols:

- More special: basic fuzzy signatures. Proximal/similar function symbols can have different names, but not different arities.
- More general: fully fuzzy signatures. Proximal/similar function symbols can have different names and different arities.

Approximate Unification



General Preliminaries

- We need to know what proximity and similarity relations are:

Definition

A proximity relation $\mathfrak{P} : \mathcal{U} \times \mathcal{U} \to [0, 1]$ is a fuzzy subset of $\mathcal{U} \times \mathcal{U}$, where $\mathcal{U}$ is a domain, that satisfies the following:

i. $\mathfrak{P}$ is reflexive (i.e. $\mathfrak{P}(x, x) = 1$ for all $x \in \mathcal{U}$);
ii. $\mathfrak{P}$ is symmetric (i.e. $\mathfrak{P}(x, y) = P(y, x)$ for all $x, y \in \mathcal{U}$).

General Preliminaries

Definition

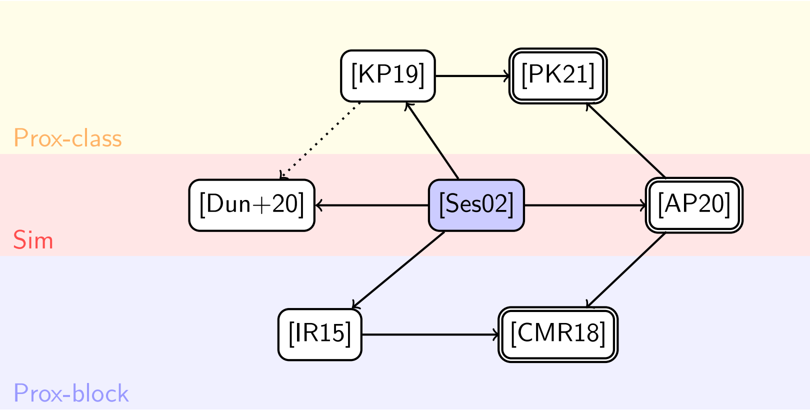
A similarity relation $\mathfrak{S} : \mathcal{U} \times \mathcal{U} \to [0,1]$ is a proximity relation that also has the transitivity regarding a t-norm (in our case the minimum t-norm) as one of its properties (i.e. $\mathfrak{S}(x,z) \geq \mathfrak{S}(x,y) \wedge \mathfrak{S}(y,z)$ for all $x, y, z \in \mathcal{U}$).

- We also need the notion of the λ-cut:

Definition

Let $\mathcal{U}$ be a domain and $\mathfrak{F}$ be a relation on $\mathcal{U}$ (can be either a proximity or a similarity relation). We define the λ-cut of $\mathfrak{F}$, for any $\lambda \in [0,1]$, as the relation $\simeq_{\mathfrak{F},\lambda}$ with: $x \simeq_{\mathfrak{F},\lambda} y \Leftrightarrow \mathfrak{F}(x,y) \geq \lambda$ for all $x, y \in [0,1]$.

Similarity Unification



Similarity Unification - Basic Case

- We now use the system $P; \alpha; \sigma$, where α denotes the similarity degree
- Trivial, Orient, Occurs Check, Variable Elimination remain the same
- Only changes occur in Decomposition and Symbol Clash:

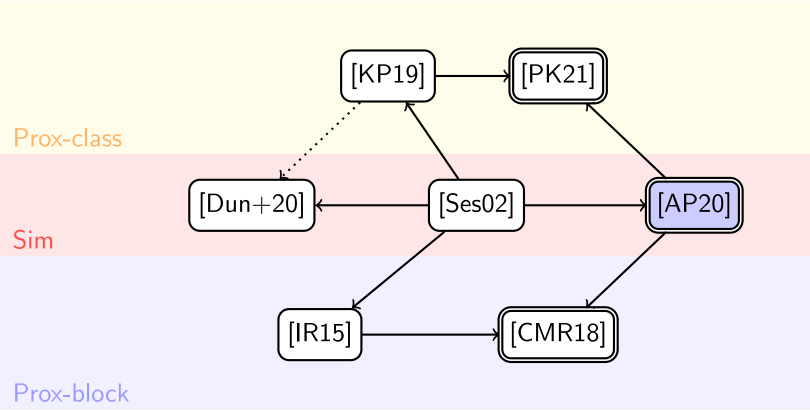
- **Similarity-Symbol Clash**: (S-SC)
  $\{f(s_1, ..., s_n) \simeq^?_{\mathfrak{S},\lambda} g(t_1, ..., t_m)\} \uplus P'; \alpha; \sigma \Rightarrow \bot$, if $\mathfrak{S}(f, g) < \lambda$.

- **Similarity-Decomposition**: (S-Dec)
  $\{f(s_1, ..., s_n) \simeq^?_{\mathfrak{S},\lambda} g(t_1, ..., t_n)\} \uplus P'; \alpha; \sigma \Rightarrow$
  $\{s_1 \simeq^?_{\mathfrak{S},\lambda} t_1, ..., s_n \simeq^?_{\mathfrak{S},\lambda} t_n\} \cup P'; \alpha \wedge \mathfrak{S}(f, g); \sigma$, if $\mathfrak{S}(f, g) \geq \lambda$,
  where $n \geq 0$.

Example

- The following similarity relation is given: $\mathfrak{S}(f,g) = 0.6, \mathfrak{S}(p,q) = 0.7, \mathfrak{S}(a,b) = \mathfrak{S}(b,c) = 0.4$ and $\mathfrak{S}(a,c) = 0.8$, the cut value $\lambda = 0.2$ and the terms that need to be unified are $f(x, p(y), b)$ and $g(a, q(c), y)$.
- Applying the algorithm gives:
  $\{f(x,p(y),b) \simeq_{\mathfrak{S},0.2} g(a,q(c),y)\}; 1; id \Rightarrow_{\text{S-Dec}}$
  $\{x \simeq_{\mathfrak{S},0.2} a, p(y) \simeq_{\mathfrak{S},0.2} q(c), b \simeq_{\mathfrak{S},0.2} y\}; 0.6; id \Rightarrow_{\text{S-VE}}$
  $\{p(y) \simeq_{\mathfrak{S},0.2} q(c), b \simeq_{\mathfrak{S},0.2} y\}; 0.6; \{x \mapsto a\} \Rightarrow_{\text{S-Or}}$
  $\{p(y) \simeq_{\mathfrak{S},0.2} q(c), y \simeq_{\mathfrak{S},0.2} b\}; 0.6; \{x \mapsto a\} \Rightarrow_{\text{S-VE}}$
  $\{p(b) \simeq_{\mathfrak{S},0.2} q(c)\}; 0.6; \{x \mapsto a, y \mapsto b\} \Rightarrow_{\text{S-Dec}}$
  $\{b \simeq_{\mathfrak{S},0.2} c\}; 0.6; \{x \mapsto a, y \mapsto b\} \Rightarrow_{\text{S-Dec}}$
  $\emptyset; 0.5; \{x \mapsto a, y \mapsto b\}$
- Substituion $\sigma = \{x \mapsto a, y \mapsto b\}$ is a solution with degree $\alpha = 0.5$.

Similarity Unification



Similarity Unification - Fully Fuzzy Case

- Take into consideration arity mismatch
- For each pair of functors f, g with arities m, respectively n, where $0 \leq m \leq n$ there exists an injective mapping $\rho_{fg} : \{1, 2, ..., m\} \to \{1, 2, ..., n\}$
- The mapping associates each of the m argument positions of f with a unique position among the n arguments of g
- It should also hold:
  1. ρ_{ff} = the identity function (i.e $1 \mapsto 1, 2 \mapsto 2, etc.$);
  2. $\rho_{fg} \circ \rho_{gf}$ = the identity function, if f and g have the same arity;
  3. for three terms f, g and h with arities m, n and respectively l, with $0 \leq m \leq n \leq l$: $\rho_{fh} = \rho_{gh} \circ \rho_{fg}$.
- Meaning it should be consistent
- (Functor meaning either function or predicate symbols)

Similarity Unification - Fully Fuzzy Case

- We use again the system $P; \alpha; \sigma$
- Trivial, Symbol Clash, Orient, Occurs Check, Variable Elimination remain the same
- Only changes occur in Decomposition and Symbol Clash
- Introduce a new rule: Equation Orient

Algorithm

- **Fully Fuzzy Similarity-Symbol Clash**: (FFS-SC)
$\{f(s_1, ..., s_m) \simeq^?_{\mathfrak{S},\lambda} g(t_1, ..., t_n)\} \uplus P'; \alpha; \sigma \Rightarrow \bot$, if $\mathfrak{S}(f, g) < \lambda$.

- **Fully Fuzzy Similarity-Decomposition**: (FFS-Dec)
$\{f(s_1, ..., s_m) \simeq^?_{\mathfrak{S},\lambda} g(t_1, ..., t_n)\} \uplus P'; \alpha; \sigma \Rightarrow$
$\{s_1 \simeq^?_{\mathfrak{S},\lambda} t_{\rho_{fg}(1)}, \dots, s_m \simeq^?_{\mathfrak{S},\lambda} t_{\rho_{fg}(m)}\} \cup P'; \alpha \wedge \mathfrak{S}(f, g); \sigma$, if $\mathfrak{S}(f, g) \geq \lambda$, where $n \geq m \geq 0$ with respect to the mapping ρ.

- **Fully Fuzzy Similarity-Equation Orient**: (FFS-EO)
$\{g(t_1, ..., t_n) \simeq^?_{\mathfrak{S},\lambda} f(s_1, ...s_m)\} \uplus P'; \alpha; \sigma \Rightarrow$
$\{f(s_1, ...s_m) \simeq^?_{\mathfrak{S},\lambda} g(t_1, ..., t_n)\} \cup P'; \alpha; \sigma$, if $n > m \geq 0$.

Example

- The following similarity relation is given: $\mathfrak{S}(p,q) = 0.7, \mathfrak{S}(h,g) = 0.3, \mathfrak{S}(c,d) = 0.5$ with cut value $\lambda = 0.2$, the mapping $\rho_{qp} = \{1 \mapsto 1, 2 \mapsto 3\}, \rho_{gh} = \{1 \mapsto 2\}$ and the terms that need to be unified are $p(h(x,y),a,y)$ and $q(g(c),d)$.
- Applying the algorithm gives:
  $\{p(h(x,y),a,y) \simeq_{\mathfrak{S},0.2} q(g(c),d)\}; 1; id \Rightarrow_{\text{FFS-EO}}$
  $\{q(g(c),d) \simeq_{\mathfrak{S},0.2} p(h(x,y),a,y)\}; 1; id \Rightarrow_{\text{FFS-Dec}}$
  $\{g(c) \simeq_{\mathfrak{S},0.2} h(x,y), d \simeq_{\mathfrak{S},0.2} y\}; 0.7; id \Rightarrow_{\text{FFS-Or}}$
  $\{g(c) \simeq_{\mathfrak{S},0.2} h(x,y), y \simeq_{\mathfrak{S},0.2} d\}; 0.7; id \Rightarrow_{\text{FFS-VE}}$
  $\{g(c) \simeq_{\mathfrak{S},0.2} h(x,d)\}; 0.7; \{y \mapsto d\} \Rightarrow_{\text{FFS-Dec}}$
  $\{c \simeq_{\mathfrak{S},0.2} d\}; 0.3; \{y \mapsto d\} \Rightarrow_{\text{FFS-Dec}}$
  $\emptyset; 0.3; \{y \mapsto d\}$
- Substituion $\sigma = \{y \mapsto d\}$ is a solution with degree $\alpha = 0.3$

Block- and Class-based Approach for Proximity

- Looking at proximity relations as undirected graphs, one can talk about cliques and neighborhoods in them.
- One distinguishes between block- and class-based approaches towards solving symbolic constraints for proximity relations.

Block- and Class-based Approach for Proximity

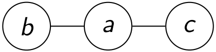
- Looking at proximity relations as undirected graphs, one can talk about cliques and neighborhoods in them.
- One distinguishes between block- and class-based approaches towards solving symbolic constraints for proximity relations.

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

Block- and Class-based Approach for Proximity

- Looking at proximity relations as undirected graphs, one can talk about cliques and neighborhoods in them.
- One distinguishes between block- and class-based approaches towards solving symbolic constraints for proximity relations.

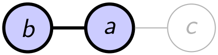
block-based	vs	class-based
block of a:		class of a:
a clique to which a belongs		the neighborhood of a



Block- and Class-based Approach for Proximity

- Looking at proximity relations as undirected graphs, one can talk about cliques and neighborhoods in them.
- One distinguishes between block- and class-based approaches towards solving symbolic constraints for proximity relations.

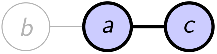
block-based	vs	class-based
block of *a*:		class of *a*:
a clique to which *a* belongs		the neighborhood of *a*



Block- and Class-based Approach for Proximity

- Looking at proximity relations as undirected graphs, one can talk about cliques and neighborhoods in them.
- One distinguishes between block- and class-based approaches towards solving symbolic constraints for proximity relations.

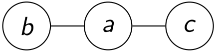
block-based	vs	class-based
block of a:		class of a:
a clique to which a belongs		the neighborhood of a



Block- and Class-based Approach for Proximity

- Looking at proximity relations as undirected graphs, one can talk about cliques and neighborhoods in them.
- One distinguishes between block- and class-based approaches towards solving symbolic constraints for proximity relations.

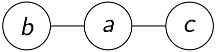
block-based	vs	class-based
block of a:		class of a:
a clique to which a belongs		the neighborhood of a

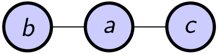


Block- and Class-based Approach for Proximity

- Looking at proximity relations as undirected graphs, one can talk about cliques and neighborhoods in them.
- One distinguishes between block- and class-based approaches towards solving symbolic constraints for proximity relations.

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

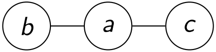




Block- and Class-based Approach for Proximity

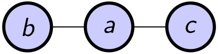
- Looking at proximity relations as undirected graphs, one can talk about cliques and neighborhoods in them.
- One distinguishes between block- and class-based approaches towards solving symbolic constraints for proximity relations.

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



$\{x \simeq_{\mathfrak{P},\lambda} b,\ x \simeq_{\mathfrak{P},\lambda} c\}$

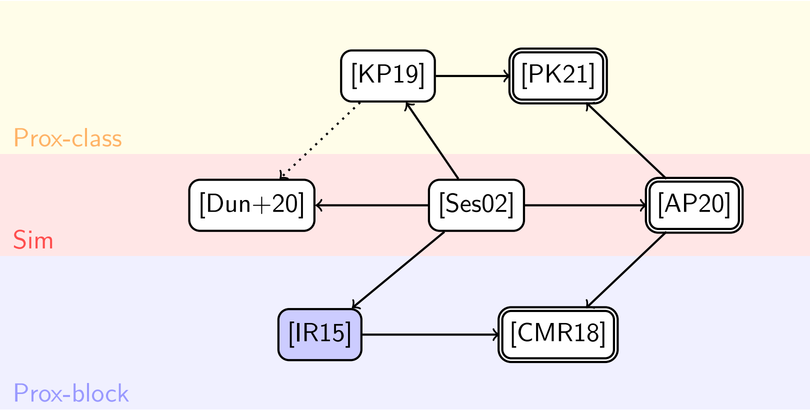
not solvable



$\{x \simeq_{\mathfrak{P},\lambda} b,\ x \simeq_{\mathfrak{P},\lambda} c\}$

solved by $\{x \mapsto a\}$

Proximity Unification



Proximity Unification - Block-Based Basic Case

- Before the rule-based algorithm some notions need to be introduced

Definition

Given a proximity relation $\mathfrak{P}$ on a domain $\mathcal{U}$, a proximity block of level λ (λ-block), denoted as $\mathcal{B}_i^\lambda$ (where i is the index of the block), is a subset of $\mathcal{U}$ such that $\simeq_{\mathfrak{P},\lambda} |_{\mathcal{B}_i^\lambda}$ is total and maximal.

- Maximal in this case means that the elements of the proximity block are not contained in another set that restricts $\simeq_{\mathfrak{P},\lambda}$ to form a total relation.

Proximity Unification - Block-Based Basic Case

Definition

Let $\mathcal{S} := \mathcal{C} \uplus \mathcal{F} \uplus \mathcal{P}$ be the union set of constants, function symbols and predicate symbols of $\mathcal{L}$. Then we define a proximity constraint $a \approx b$ as an unordered pair of elements $a, b \in S$.

- The following definition will also be needed

Definition

Given a proximity relation $\mathfrak{P}$, a cut value $\lambda \in [0, 1]$ and a set C of proximity constraints, the function Sat looks at all the constraints $a \approx b$ in this set C, and takes the value fail if and only if it finds $a \approx b$ in C with $\mathfrak{P}(a, b) < \lambda$. Otherwise Sat$(C)$ returns success.

Algorithm

- We use the system $P; C; \alpha; \sigma$, where C is the set of proximity constraints
- Trivial, Orient, Variable Elimination and Occurs Check remain the same
- Decomposition changes into 2 rules
- Symbol Clash changes

Algorithm

- **Block-Based Proximity-Decomposition1**: (BBP-Dec1)
  $\{f(s_1, ..., s_n) \simeq^?_{\mathfrak{P},\lambda} f(t_1, ..., t_n)\} \uplus P'; C; \alpha; \sigma \Rightarrow$
  $\{s_1 \simeq^?_{\mathfrak{P},\lambda} t_1, ..., s_n \simeq^?_{\mathfrak{P},\lambda} t_n\} \cup P'; C; \alpha; \sigma$, where $n \geq 0$.

- **Block-Based Proximity-Decomposition2**: (BBP-Dec2)
  $\{f(s_1, ..., s_n) \simeq^?_{\mathfrak{P},\lambda} g(t_1, ..., t_n)\} \uplus P'; C; \alpha; \sigma \Rightarrow$
  $\{s_1 \simeq^?_{\mathfrak{P},\lambda} t_1, ..., s_n \simeq^?_{\mathfrak{P},\lambda} t_n\} \cup P'; C \cup \{f \approx g\}; \alpha \wedge \mathfrak{P}(f, g); \sigma$,
  if $\mathfrak{P}(f, g) \geq \lambda$, $\text{Sat}(C \cup \{f \approx g\}) \neq \text{fail}$, where $n \geq 0$.

- **Block-Based Proximity-Symbol Clash**: (BBP-SC)
  $\{f(s_1, ..., s_n) \simeq^?_{\mathfrak{P},\lambda} g(t_1, ..., t_m)\} \uplus P'; C; \alpha; \sigma \Rightarrow \perp$, if
  $n \neq m, \mathfrak{P}(f, g) < \lambda$ or $Sat(C \cup \{f \approx g\}) = fail$.

Example

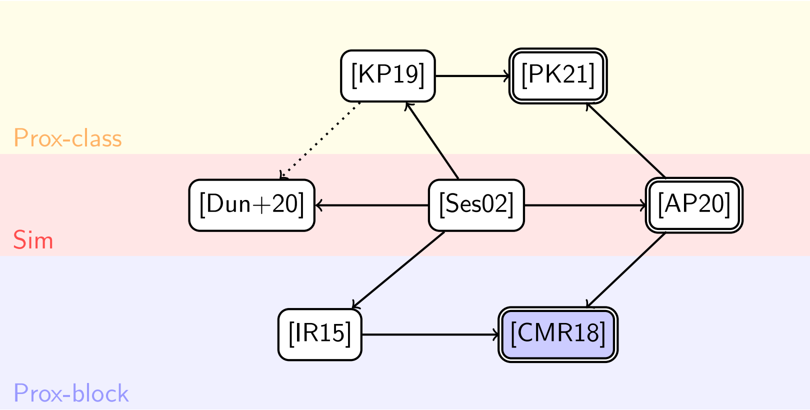
- The following proximity relation is given: $\mathfrak{P}(f,g) = 0.5, \mathfrak{P}(a,c) = 0.2, \mathfrak{P}(b,d) = 0.3, \mathfrak{P}(p,q) = 0.6$, the cut value $\lambda = 0.4$ and the terms that need to be unified $f(x, p(x), b, y)$ respectively $g(a, q(c), d, p(d))$.
- Applying the algorithm gives:

$$\{f(x, p(x), b, y) \simeq_{\mathfrak{P},0.4} g(a, q(c), d, p(d))\}; \emptyset; 1; id \Rightarrow_{\text{BBP-Dec}}$$
$$\{x \simeq_{\mathfrak{P},0.4} a, p(x) \simeq_{\mathfrak{P},0.4} q(c), b \simeq_{\mathfrak{P},0.4} d, y \simeq_{\mathfrak{P},0.4} p(d)\}; 0.5; id \Rightarrow_{\text{BBP-VE}}$$
$$\{p(a) \simeq_{\mathfrak{P},0.4} q(c), b \simeq_{\mathfrak{P},0.4} d, y \simeq_{\mathfrak{P},0.4} p(d)\}; \{f \approx g\}; 0.5; \{x \mapsto a\} \Rightarrow_{\text{BBP-Dec}}$$
$$\{a \simeq_{\mathfrak{P},0.4} c, b \simeq_{\mathfrak{P},0.4} d, y \simeq_{\mathfrak{P},0.4} p(d)\}; \{f \approx g, p \approx q\}; 0.5; \{x \mapsto a\} \Rightarrow_{\text{BBP-Dec}}$$
$$\{b \simeq_{\mathfrak{P},0.4} d, y \simeq_{\mathfrak{P},0.4} p(d)\}; \{f \approx g, p \approx q, a \approx c\}; 0.5; \{x \mapsto a\}.$$

Example

- $\{b \simeq_{\mathfrak{P},0.4} d, y \simeq_{\mathfrak{P},0.4} p(d)\}; \{f \approx g, p \approx q, a \approx c\}; 0.5; \{x \mapsto a\}$ $\Rightarrow_{\text{BBP-Dec}}$ $\{y \simeq_{\mathfrak{P},0.4} p(d)\}; \{f \approx g, p \approx q, a \approx c, b \approx d\}; 0.5; \{x \mapsto a\} \Rightarrow_{\text{BBP-VE}}$ $\emptyset; 0.5; \{f \approx g, p \approx q, a \approx c, b \approx d\}; \{x \mapsto a, y \mapsto p(d)\}$.
- Substituion $\sigma = \{x \mapsto a, y \mapsto p(d)\}$ is a solution with degree $\alpha = 0.5$ and constraints set $C = \{f \approx g, p \approx q, a \approx c, b \approx d\}$.

Proximity Unification



Proximity Unification - Block-Based Fully Fuzzy Case

- Algorithm only takes into consideration the case where the terms to be unified contain only constants
- Solution after applying the algorithm would then remain *id*
- Only the degree changes

Definition

1. Let $\mathscr{S}$ be a set containing $S_1, ..., S_k$ (which are named "sorts").
2. We denote X_{S_i}, Y_{S_i} the sets containing all constants of sort S_i from the terms that we want to unify (named "sort sets").
3. The degree between X_{S_i} and Y_{S_i} is the maximal degree between the constants from X_{S_i} and Y_{S_i} via a proximity relation $\mathfrak{P}$.

Algorithm

- We use the system $P; \alpha; \sigma$
- Trivial, Occurs Check and Orient remain the same
- We don't have the Variable Elimination rule anymore
- Decomposition and Symbol Clash split into 2 rules
- New rule is added: Equation Elimination

Algorithm

- **Fully Fuzzy Block-Based Proximity-Decomposition1**: (FFBBP-Dec1)
  $\{f(s_1, \ldots, s_n) \simeq^?_{\mathfrak{P},\lambda} g(t_1, \ldots, t_n)\} \uplus P'; \alpha; \sigma \Rightarrow$
  $\{X_{S_1} \simeq^?_{\mathfrak{P},\lambda} Y_{S_1}, \ldots, X_{S_k} \simeq^?_{\mathfrak{P},\lambda} Y_{S_k}\} \cup P'; \alpha \wedge \mathfrak{P}(f, g); \sigma$, where $n \geq 0$, $k \geq 0$ and $\mathfrak{P}(f, g) \geq \lambda$.
  Also each X_s and Y_s contain the arguments of f respectively g that belong to their respective sort s.

- **Fully Fuzzy Block-Based Proximity-Decomposition2**: (FFBBP-Dec2)
  $\{X_{S_i} \simeq^?_{\mathfrak{P},\lambda} Y_{S_i}\} \uplus P'; \alpha; \sigma \Rightarrow P'; \alpha \wedge \mathfrak{P}(X_{S_i}, Y_{S_i}); \sigma$ if $\mathfrak{P}(X_{S_i}, Y_{S_i}) \geq \lambda$, where $n \geq 0, i \geq 0$.

Algorithm

- **Fully Fuzzy Block-Based Proximity-Symbol Clash1**: (FFBBP-SC1)
  $\{f(s_1, ..., s_n) \simeq^?_{\mathfrak{P},\lambda} g(t_1, ..., t_m)\} \uplus P; \alpha; \sigma \Rightarrow \bot$, if $\mathfrak{P}(f, g) < \lambda$.

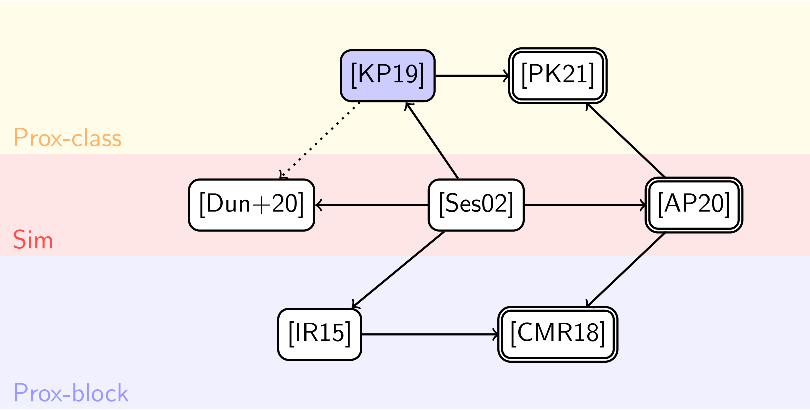
- **Fully Fuzzy Block-Based Proximity-Symbol Clash2**: (FFBBP-SC2)
  $\{X_{S_i} \simeq^?_{\mathfrak{P},\lambda} Y_{S_i}\} \uplus P; \alpha; \sigma \Rightarrow \bot$, if $\mathfrak{P}(X_{S_i}, Y_{S_i}) < \lambda$.

- **Fully Fuzzy Block-Based Proximity-Equation Elimination**: (FFBBP-EE)
  $\{X_{S_i} \simeq^?_{\mathfrak{P},\lambda} Y_{S_i}\} \cup P'; \alpha; \sigma \Rightarrow P'; \alpha; \sigma$, if $X_{S_i} = \emptyset$ and/or $Y_{S_i} = \emptyset$.

Example

- The following proximity relation is given: $\mathfrak{P}(f, g) = 0.6, \mathfrak{P}(a, b) = 0.4, \mathfrak{P}(b, c) = 0.3, \mathfrak{P}(a, c) = 0.3$, the cut value $\lambda = 0.2$, the set of sorts $\mathscr{S} = \{S_1, S_2, S_3\}$, with $S_1 = \{a, b\}, S_2 = \{c\}, S_3 = \{d\}$ and the terms that need to be unified $f(a, c, b)$ respectively $g(d, a)$.
- Applying the algorithm gives:
  $\{f(a, c, b) \simeq_{\mathfrak{P},0.2} g(d, a)\}; 1; id \Rightarrow_{\text{FFBBP-Dec1}}$
  $\{X_{S_1} \simeq^?_{\mathfrak{P},0.2} Y_{S_1}, X_{S_2} \simeq^?_{\mathfrak{P},0.2} Y_{S_2}, X_{S_3} \simeq^?_{\mathfrak{P},0.2} Y_{S_3}\}; 0.6; id$, where $X_{S_1} = \{a, b\}, X_{S_2} = \{c\}, X_{S_3} = \emptyset$, $Y_{S_1} = \{a\}$, $Y_{S_2} = \emptyset$ and $Y_{S_3} = \{d\} \Rightarrow_{\text{FFBBP-EE}}$
  $\{X_{S_1} \simeq^?_{\mathfrak{P},0.2} Y_{S_1}, X_{S_2} \simeq^?_{\mathfrak{P},0.2} Y_{S_2}\}; 0.6; id \Rightarrow_{\text{FFBBP-EE}}$
  $\{X_{S_1} \simeq^?_{\mathfrak{P},0.2} Y_{S_1}\}; 0.6; id \Rightarrow_{\text{FFBBP-Dec2}}$
  $\emptyset; 0.6; id$
- Substituion $\sigma = id$ is a solution with degree $\alpha = 0.6$.

Proximity Unification



Proximity Unification - Class-Based Basic Case

We use now proximity classes:

Definition

We have a proximity relation $\mathfrak{P}$ on a set S and a cut-value $\lambda \in (0, 1]$. Then we define the proximity class of level λ of $s \in S$ (denoted as $\mathbf{pc}(s, \mathfrak{P})$), as the set: $\mathbf{pc}(s, \mathfrak{P}) := \{t \in S \mid \mathfrak{P}(s, t) \geq \lambda\}$.

We also need the notion of extended terms:

Definition

An extended term is a term that includes, besides variables and function symbols, finite sets of function symbols, whose elements have the same arity. We denote them in bold: $\mathbf{t}$ for eg.

Proximity Unification - Class-Based Basic Case

a) We consider now the countable set $\mathcal{N}$ the set of names. Names are symbols with associated arity (like function symbols). We assume that $\mathcal{N} \cap \mathcal{F} = \emptyset, \mathcal{N} \cap \mathcal{V} = \emptyset$. They are denoted as N, M, K.

b) Now a neighborhood is either a name or a finite subset of $\mathcal{F}$, where all elements have the same arity. We denote it as **Nb**.

c) We denote Φ as a name-neighborhood mapping, which is a finite mapping from names to non-name neighborhoods.

d) A neighborhood equation is a pair of neighborhoods that needs to be solved, i.e. $\mathbf{F} =^? \mathbf{G}$.

e) A neighborhood constraint is a finite set of neighborhood equations.

f) We say that $\{x \simeq_{\mathfrak{P},\lambda} \mathbf{t}\} \uplus P$ contains an occurence cycle for the variable x, if $\mathbf{t} \notin \mathcal{V}$ and there exist $(x_0, \mathbf{t}_0), (x_1, \mathbf{t}_1), ..., (x_n, \mathbf{t}_n)$ such that $x_0 = x, \mathbf{t}_0 = \mathbf{t}$, for each $0 \leq i \leq n$ P contains an equation $x_i \simeq_{\mathfrak{P},\lambda} \mathbf{t}_i$ or $\mathbf{t}_i \simeq_{\mathfrak{P},\lambda} x_i$, and $x_{i+1} \in \mathcal{V}(\mathbf{t}_i)$, where $x_{n+1} = x_0$.

Pre-Unification Algorithm

- Use the system $P; C; \alpha; \sigma$, where C is the set of proximity constraints that need to be solved
- First apply the pre-unification algorithm to get σ
- Then apply constraint solving algorithm to computed C to get Φ
- The solution will be then $\Phi(\sigma)$
- Trivial, Orient and Occurs Check remain the same
- Decomposition and Variable Elimination change
- Symbol Clash transforms to Clash

Pre-Unification Algorithm

- **Class-Based Proximity-Decomposition**: (CBP-Dec)
$\{\mathbf{F}(\mathbf{s}_1, ..., \mathbf{s}_n) \simeq^?_{\mathfrak{P},\lambda} \mathbf{G}(\mathbf{t}_1, ..., \mathbf{t}_n)\} \uplus P'; C; \alpha; \sigma \Rightarrow$
$\{\mathbf{s}_1 \simeq^?_{\mathfrak{P},\lambda} \mathbf{t}_1, ..., \mathbf{s}_n \simeq^?_{\mathfrak{P},\lambda} \mathbf{t}_n\} \cup P'; \{\mathbf{F} \approx^? \mathbf{G}\} \cup C; \alpha \wedge \mathfrak{P}(\mathbf{F}, \mathbf{G}); \sigma$,
where $n \geq 0$ and $\mathfrak{P}(\mathbf{F}, \mathbf{G}) \geq \lambda$.

- **Class-Based Proximity-Clash**: (CBP-C)
$\{\mathbf{F}(\mathbf{s}_1, ..., \mathbf{s}_n) \simeq^?_{\mathfrak{P},\lambda} \mathbf{G}(\mathbf{t}_1, ..., \mathbf{t}_m)\} \uplus P'; C; \alpha; \sigma \Rightarrow \bot$, if $n \neq m$.

- **Class-Based Proximity-Variable Elimination**: (CBP-VE)
$\{x \simeq^?_{\mathfrak{P},\lambda} \mathbf{t}\} \cup P'; C; \alpha; \sigma \Rightarrow$
$\{\mathbf{t}' \simeq^?_{\mathfrak{P},\lambda} \mathbf{t}\} \cup P'\{x \mapsto \mathbf{t}'\}; C; \alpha; \sigma\{x \mapsto \mathbf{t}'\} \cup \{x \mapsto \mathbf{t}'\}$, where $\mathbf{t} \notin \mathcal{V}$, there is no occurrence cycle for x in $\{x \simeq_{\mathfrak{P},\lambda} \mathbf{t}\}$, and $\mathbf{t}'$ is a a fresh copy of $\mathbf{t}$.

- There is also such an algorithm that solves the constraints obtained from this one.

Example

- We want to unify $p(x,y,x)$ and $q(f(a),g(d),y)$, with the proximity relation: $\mathfrak{P}(f,g) = 0.3, \mathfrak{P}(a,b) = 0.2, \mathfrak{P}(p,q) = 0.7, \mathfrak{P}(c,d) = 0.75, \mathfrak{P}(b,c) = 0.35$ and the cut value $\lambda = 0.2$.
- We use then the pre-unification algorithm first:
$\{p(x,y,x) \simeq^?_{\mathfrak{P},0.2} q(f(a),g(d),y)\}; \emptyset; 1; id \Rightarrow_{\text{CBP-Dec}}$
$\{x \simeq^?_{\mathfrak{P},0.2} f(a), y \simeq^?_{\mathfrak{P},0.2} g(d), x \simeq^?_{\mathfrak{P},0.2} y\}; \{p \approx q\}; 0.7; id \Rightarrow_{\text{CBP-VE}}$
$\{N_2 \simeq^?_{\mathfrak{P},0.2} a, y \simeq^?_{\mathfrak{P},0.2} g(d), t' \simeq^?_{\mathfrak{P},0.2} y\}; \{p \approx q, N_1 \approx f\};$
$0.7; \{x \mapsto t'\}$, where $t' = N_1(N_2) \Rightarrow_{\text{CBP-VE}}$
$\{y \simeq^?_{\mathfrak{P},0.2} g(d), t' \simeq^?_{\mathfrak{P},0.2} y\}; \{p \approx q, N_1 \approx f, N_2 \approx a\};$
$0.7; \{x \mapsto t'\} \Rightarrow_{\text{CBP-VE}}$
$\{N_4 \simeq^?_{\mathfrak{P},0.2} d, t' \simeq^?_{\mathfrak{P},0.2} s'\};$
$\{p \approx q, N_1 \approx f, N_2 \approx a, N_3 \approx g\}; 0.7; \{x \mapsto t', y \mapsto s'\}$, where $s' = N_3(N_4)$.

Example

$$\{N_4 \simeq^?_{\mathfrak{P},0.2} d, t' \simeq^?_{\mathfrak{P},0.2} s'\}; \{p \approx q, N_1 \approx f, N_2 \approx a, N_3 \approx g\};$$
$$0.7; \{x \mapsto t', y \mapsto s'\}, \text{ where } s' = N_3(N_4)$$

$\Rightarrow_{\text{CBP-VE}}$

$\{N_1(N_2) \simeq^?_{\mathfrak{P},0.2} N_3(N_4)\};$

$\{p \approx q, N_1 \approx f, N_2 \approx a, N_3 \approx g, N_4 \approx d\}; 0.7; \{x \mapsto t', y \mapsto s'\}$

$\Rightarrow_{\text{CBP-Dec}}$

$\{N_2 =^? N_4\};$

$\{p \approx q, N_1 \approx f, N_2 \approx a, N_3 \approx g, N_4 \approx d, N_1 \approx N_3\}; 0.3; \{x \mapsto t', y \mapsto s'\}$

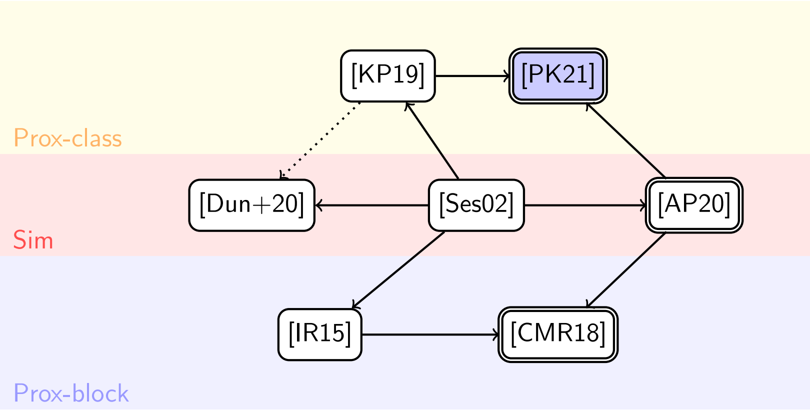
$\Rightarrow_{\text{CBP-Dec}}$

$$\emptyset; \{p \approx q, N_1 \approx f, N_2 \approx a, N_3 \approx g, N_4 \approx d, N_1 \approx N_3, N_2 \approx N_4\};$$
$$0.3; \{x \mapsto N_1(N_2), y \mapsto N_3(N_4)\}.$$

Example

- Then by applying the constraint solving algorithm on $\{p \approx q, N_1 \approx f, N_2 \approx a, N_3 \approx g, N_4 \approx d, N_1 \approx N_3, N_2 \approx N_4\}$, we get the substitution $\Phi = \{N_1 \mapsto \{f, g\}, N_2 \mapsto \{b\}, N_3 \mapsto \{f, g\}, N_4 \mapsto \{c\}\}$.
- One of the solutions is then $\Phi(\sigma) = \{x \mapsto f(b), y \mapsto g(c)\}$, with degree $\alpha = 0.3$.

Proximity Unification



Proximity Unification - Class-Based Fully Fuzzy Case

- Again take into consideration arity mismatch
- Introduce argument relation ρ
- We use the system $P; \alpha; \sigma$
- Trivial, Orient and Occurence Check stay the same
- Decomposition, Symbol Clash and Variable Elimination change

Algorithm

- **Fully Fuzzy Class-Based Proximity-Decomposition**: (FFCBP-Dec)
$\{f(s_1, \ldots, s_m) \simeq^?_{\mathfrak{P},\lambda} g(t_1, \ldots, t_n)\} \uplus P'; \alpha; \sigma \Rightarrow$
$\{s_i \simeq^?_{\mathfrak{P},\lambda} t_j \mid (i,j) \in \rho\} \cup P'; \alpha \wedge \mathfrak{P}(f,g); \sigma$ if $\mathfrak{P}(f,g) \geq \lambda$, where $n, m \geq 0$ with respect to the relation ρ.

- **Fully Fuzzy Class-Based Proximity-Symbol Clash**: (FFCBP-SC)
$\{f(s_1, \ldots, s_n) \simeq^?_{\mathfrak{P},\lambda} g(t_1, \ldots, t_m)\} \uplus P'; \alpha; \sigma \Rightarrow \bot$ if $\mathfrak{P}(f,g) < \lambda$.

- **Fully Fuzzy Class-Based Proximity-Variable Elimination**: (FFSCBP-VE)
$\{x \simeq^?_{\mathfrak{P},\lambda} g(s_1, \ldots, s_n)\} \cup P'; \alpha; \sigma \Rightarrow$
$P'\theta \cup \{v_i \simeq^?_{\mathfrak{P},\lambda} s_j \mid (i,j) \in \rho\}; \alpha \wedge \mathfrak{P}(f,g); \sigma\theta \cup \{x \mapsto t\}$, where $\{x \simeq^?_{\mathfrak{P},\lambda} g(s_1, \ldots, s_n)\}$ does not contain an occurrence cycle for x, $\theta = \{x \mapsto f(v_1, \ldots, v_m)\}$, with fresh variables $v_1, \ldots, v_m$, $\mathfrak{P}(f,g) \geq \lambda$, with respect to ρ and $n, m \geq 0$.

Example

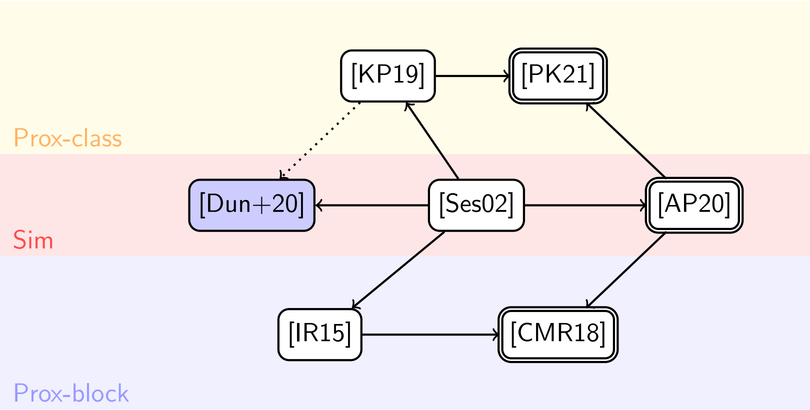
- The following proximity relation is given: $\mathfrak{P}(f,g) = 0.6, \mathfrak{P}(f,h) = 0.7, \mathfrak{P}(a,b) = 0.4, \mathfrak{P}(b,c) = 0.3$, the cut value $\lambda = 0.2$, the relations $\rho_{fg} = \{(1,1),(2,1)\}$, $\rho_{fh} = \{(1,1),(2,2)\}$ and the terms that need to be unified are $f(x,x)$ and $f(g(a),h(a,c))$.
- Applying the algorithm gives:

$$\{f(x,x) \simeq_{\mathfrak{P},0.2} f(g(a),h(a,c))\}; 1; id \Rightarrow_{\text{FFCBP-Dec}}$$
$$\{x \simeq_{\mathfrak{P},0.2} g(a), x \simeq_{\mathfrak{P},0.2} h(a,c)\}; 1; id \Rightarrow_{\text{FFCBP-VE}}$$
$$\{v_1 \simeq^?_{\mathfrak{P},0.2} a, v_2 \simeq^?_{\mathfrak{P},0.2} a, f(v_1,v_2) \simeq^?_{\mathfrak{P},0.2} h(a,c)\}; 0.6; \{x \mapsto f(v_1,v_2)\} \Rightarrow_{\text{FFCBP-Dec}}$$
$$\{v_1 \simeq^?_{\mathfrak{P},0.2} a, v_2 \simeq^?_{\mathfrak{P},0.2} a, v_1 \simeq^?_{\mathfrak{P},0.2} a, v_2 \simeq^?_{\mathfrak{P},0.2} c\}; 0.6; \{x \mapsto f(v_1,v_2)\} \Rightarrow_{\text{FFCBP-VE}}$$
$$\{v_2 \simeq^?_{\mathfrak{P},0.2} a, a \simeq^?_{\mathfrak{P},0.2} a, v_2 \simeq^?_{\mathfrak{P},0.2} c\}; 0.6; \{x \mapsto f(a,v_2), v_1 \mapsto a\} \Rightarrow_{\text{FFCBP-Tri}}$$
$$\{v_2 \simeq^?_{\mathfrak{P},0.2} a, v_2 \simeq^?_{\mathfrak{P},0.2} c\}; 0.6; \{x \mapsto f(a,v_2), v_1 \mapsto a\}$$

Example

- $\{v_2 \simeq^?_{\mathfrak{P},0.2} a, v_2 \simeq^?_{\mathfrak{P},0.2} c\}; 0.6; \{x \mapsto f(a, v_2), v_1 \mapsto a\} \Rightarrow_{\text{FFCBP-VE}}$
  $\{b \simeq^?_{\mathfrak{P},0.2} c\}; 0.6; \{x \mapsto f(a, b), v_1 \mapsto a, v_2 \mapsto b\} \Rightarrow_{\text{FFCBP-Dec}}$
  $\emptyset; 0.3; \{x \mapsto f(a, b), v_1 \mapsto a, v_2 \mapsto b\}$
- Substituion $\sigma = \{x \mapsto f(a, b)\}$ is a solution with degree $\alpha = 0.3$

Multiple Similarities Unification



Multiple Similarities

- Take into consideration the case when there are more similarity relations between objects
- The "relation" between those similarities become a proximity relation
- New algorithm for multiple similarities

Conclusion

- We saw the respective algorithms on how to deal with different symbols and not fail, using fuzzy relations
- And on how to deal with different arities

– I implemented Sessa's algorithm in Prolog

◇ This work showed that it would be interesting to extend fully fuzzy block-based proximity unification by taking variables into consideration

◇ It is a potential future work, using CI-unification algorithm