Semantics for proximity-based logic programming

Maximilian Donnermair

2024-12-03

Background: unification

Task: find unifiers of terms t to s, i.e., substitutions σ such that $t\sigma = s\sigma$.

Let $a \sim b$ and $b \sim c$. Then

- $p(b, b)$ not close to $p(a, a)$
- $p(b, b)$ not close to $p(c, c)$
- $p(b, b)$ not close to $p(a, c)$

Background: unification

Task: find $(\mathcal{R}, \lambda)$-unifiers of terms t to s, i.e., substitutions σ such that $\mathcal{R}(t\sigma, s\sigma) \geq \lambda$ with cut value λ and similarity relation $\mathcal{R}$.

1. from crisp case to similarity relation

Let $a \sim b$ and $b \sim c$. Then

- $p(b, b)$ close to $p(a, a)$
- $p(b, b)$ close to $p(c, c)$
- $p(b, b)$ not close to $p(a, c)$

Background: unification

Task: find $(\mathcal{R}, \lambda)$-unifiers of terms t to s, i.e., substitutions σ such that $\mathcal{R}(t\sigma, s\sigma) \geq \lambda$ with cut value λ and proximity relation $\mathcal{R}$.

1. from crisp case to similarity relation
2. from similarity relation to block-based proximity relation

Let $a \sim b$ and $b \sim c$. Then

- $p(b, b)$ close to $p(a, a)$
- $p(b, b)$ close to $p(c, c)$
- $p(b, b)$ not close to $p(a, c)$

Background: unification

Task: find $(\mathcal{R}, \lambda)$-unifiers of terms t to s, i.e., substitutions σ such that $\mathcal{R}(t\sigma, s\sigma) \geq \lambda$ with cut value λ and proximity relation $\mathcal{R}$.

1. from crisp case to similarity relation
2. from similarity relation to block-based proximity relation
3. from block-based proximity to class-based proximity

Let $a \sim b$ and $b \sim c$. Then

- $p(b, b)$ close to $p(a, a)$
- $p(b, b)$ close to $p(c, c)$
- $p(b, b)$ close to $p(a, c)$

Background: unification

Task: find $(\mathcal{R}, \lambda)$-unifiers of terms t to s, i.e., substitutions σ such that $\mathcal{R}(t\sigma, s\sigma) \geq \lambda$ with cut value λ and proximity relation $\mathcal{R}$.

1. from crisp case to similarity relation
2. from similarity relation to block-based proximity relation
3. from block-based proximity to class-based proximity
4. from minimum T-norm to arbitrary T-norms

Let $a \sim b$ and $b \sim c$. Then

- $p(b, b)$ close to $p(a, a)$
- $p(b, b)$ close to $p(c, c)$
- $p(b, b)$ close to $p(a, c)$
- all depending on T-norm and cut value

From unification to logic programming

Literature

- 2002, Maria Sessa: *"Approximate reasoning by similarity-based SLD-resolution"*

From unification to logic programming

Literature

- 2002, Maria Sessa: *"Approximate reasoning by similarity-based SLD-resolution"*
- 2017, Julián-Iranzo and Rubio-Manzano: *"A sound and complete semantics for a similarity-based logic programming language"*
  - includes declarative and operational semantics

From unification to logic programming

Literature

- 2002, Maria Sessa: *“Approximate reasoning by similarity-based SLD-resolution”*
- 2017, Julián-Iranzo and Rubio-Manzano: *“A sound and complete semantics for a similarity-based logic programming language”*
  - includes declarative and operational semantics
- 2023, Julián-Iranzo and Sáenz-Pérez: *“Bousi Prolog - Design and implementation of a proximity-based fuzzy logic programming language”*
  - only block-based approach
  - no declarative semantics

Declarative semantics: Interpretation

Theoretical foundation on definite (Horn) clauses $A \leftarrow B_1, \ldots, B_n$.
Interpretation function: $\mathcal{I} = \langle \mathcal{D}, \mathcal{V} \rangle$, where

- $\mathcal{D}$ subset of Herbrand base
- $\mathcal{V} : \mathcal{D}^n \rightarrow [0, 1]$

Declarative semantics: Interpretation

Theoretical foundation on definite (Horn) clauses $A \leftarrow B_1, \ldots, B_n$.
Interpretation function: $\mathcal{I} = \langle \mathcal{D}, \mathcal{V} \rangle$, where

- $\mathcal{D}$ subset of Herbrand base
- $\mathcal{V} : \mathcal{D}^n \rightarrow [0, 1]$

Recursively:

- $\mathcal{I}(A_1, \ldots, A_n) = \bigwedge_{i=1}^{n} \mathcal{I}(A_i)$
- $\mathcal{I}(A \leftarrow Q) = \begin{cases} \mathcal{I}(A) & \text{if } \mathcal{I}(A) < \mathcal{I}(Q) \\ 1 & \text{else} \end{cases}$

Declarative semantics: Annotation

- equip elements of program with truth value 1
- instantiate with terms from Herbrand universe
- subsequently add atoms in proximity classes

Declarative semantics: Annotation

- equip elements of program with truth value 1
- instantiate with terms from Herbrand universe
- subsequently add atoms in proximity classes
- problem: non-linear atoms (e.g., $p(X, X)$)

Declarative semantics: Annotation

- equip elements of program with truth value 1
- instantiate with terms from Herbrand universe
- subsequently add atoms in proximity classes
- problem: non-linear atoms (e.g., $p(X,X)$)

Linearization

Needed for defining notions of *model* and *logical consequence*.

Example:
$lin(\{p(X,X)\}) = \{p(X,Y) \leftarrow X \sim Y\} = \{p(X,Y) \leftarrow \mathcal{R}(X,Y)\}$

In proximity case: $lin(\{p(X,X)\}) =$
$\{p(Y,Z) \leftarrow Y \sim X, X \sim Z\} = \{p(Y,Z) \leftarrow \mathcal{R}(Y,X) \otimes \mathcal{R}(X,Z)\}$

Declarative semantics: Models

Models

- $\mathcal{I}$ is λ-model for formula $\mathcal{F}$ iff $\mathcal{I}(\mathcal{F}) \geq \lambda$.

Declarative semantics: Models

Models

- $\mathcal{I}$ is λ-model for formula $\mathcal{F}$ iff $\mathcal{I}(\mathcal{F}) \geq \lambda$.
- Looking for suitable notion of least Herbrand λ-model

Declarative semantics: Models

Models

- $\mathcal{I}$ is λ-model for formula $\mathcal{F}$ iff $\mathcal{I}(\mathcal{F}) \geq \lambda$.
- Looking for suitable notion of least Herbrand λ-model

Declarative semantics: Models

Models

- $\mathcal{I}$ is λ-model for formula $\mathcal{F}$ iff $\mathcal{I}(\mathcal{F}) \geq \lambda$.
- Looking for suitable notion of least Herbrand λ-model

Tasks

- describe fixpoint/immediate consequence operator
- check *model intersection property*

Operational semantics

Weak (fuzzy) unification

Has to be checked for correctness.

Operational semantics

Weak (fuzzy) unification

Has to be checked for correctness.

Weak SLD-resolution

- decide between computing family of answers or constraints/best answer
- adapt for linearized programs
- check for correctness

Further outlook

Rewriting

Builds on *fuzzy matching*:
Find all $(\mathcal{R}, \lambda)$-matchers of t to s, i.e., substitutions σ such that $\mathcal{R}(t\sigma, s) \geq \lambda$.

Further outlook

Rewriting

Builds on *fuzzy matching*:
Find all $(\mathcal{R}, \lambda)$-matchers of t to s, i.e., substitutions σ such that $\mathcal{R}(t\sigma, s) \geq \lambda$.

- *Task:* Extend Meseguer's rewriting logic (foundation of the Maude programming language)

Further outlook

Rewriting

Builds on *fuzzy matching*:
Find all $(\mathcal{R}, \lambda)$-matchers of t to s, i.e., substitutions σ such that $\mathcal{R}(t\sigma, s) \geq \lambda$.

- *Task:* Extend Meseguer's rewriting logic (foundation of the Maude programming language)

Quantales

Generalization of "arbitrary T-norms".