(Similarity-based) Qualified Logic Programming
A General Overview

Carlos A. Romero-Díaz
Grupo de Programación Declarativa
DSIC - Universidad Complutense de Madrid, Spain
cromdia@fdi.ucm.es

Università di Salerno (Italy)
December 2008
Introduction

**Motivation:**
- Work out an expressive framework for *Constraint Functional-Logic Programming* with uncertainty.
- In particular, extend the multiparadigm language TOY with support for uncertainty and maybe more.
Motivation:

- Work out an expressive framework for Constraint Funcional-Logic Programming with uncertainty.
- In particular, extend the multiparadigm language TOY with support for uncertainty and maybe more.

Planning:

- Two main phases to develop:
  1. Framework for (Constraint) Logic Programming.
  2. Extend the previous framework with non-deterministic lazy functions and constraints (that is, Constraint Functional-Logic Programming).

This talk is focused on the actual results of the first phase.
Introduction (II): Starting the work

1. Search for fundamental works in the field to start our investigation: taken as base a seminal paper by *van Emden* based in a previous one by *Shapiro*.
Introduction (II): Starting the work

1. Search for fundamental works in the field to start our investigation: taken as base a seminal paper by van Emden based in a previous one by Shapiro.

2. Revise the early proposal by van Emden (1986): Quantitative Logic Programming and improve it in two ways:
   ▶ generalizing van Emden’s QLP to a generic scheme parametrized by a qualification domain \( D \).
   ▶ generalizing van Emden’s results by providing stronger ones, both in declarative semantics and goal solving.

Obtaining as result a basic QLP(\( D \)).
Introduction (II): Starting the work

1. Search for fundamental works in the field to start our investigation: taken as base a seminal paper by van Emden based in a previous one by Shapiro.

2. Revise the early proposal by van Emden (1986): *Quantitative Logic Programming* and improve it in two ways:
   - generalizing van Emden’s QLP to a generic scheme parametrized by a qualification domain $D$.
   - generalizing van Emden’s results by providing stronger ones, both in declarative semantics and goal solving.

Obtaining as result a basic QLP($D$).

3. Extending QLP($D$) with characteristics of other related approaches as similarity-based reasoning, and improving its expresiveness with threshold constraints (annotations in the body of clauses).
Van Emden considered Quantitative Logic Programs as sets of clauses of the form:

\[ A \leftarrow f \times B_1, \ldots, B_n \]

for which the certainty being propagated to a clause head was \( f \times b \) where:

- \( f \) - attenuation factor with \( f \in (0, 1] \)
- \( b \) - minimum of the certainty factors known for the body atoms
Van Emden considered Quantitative Logic Programs as sets of clauses of the form:

$$A \leftarrow f - B_1, \ldots, B_n$$

for which the certainty being propagated to a clause head was $f \times b$ where:

- $f$ - attenuation factor with $f \in (0, 1]$
- $b$ - minimum of the certainty factors known for the body atoms

This approach led to:

- general results on model theoretic and fixpoint semantics (similar to those for classical LP)
- procedure for computing the certainty factors in the least Herbrand model of a given program by an alpha-beta hueristic (only for ground atoms with finite search trees)
Contents
Qualification Domains
Qualification Domains: Axioms

- **Structure** $\mathcal{D} = \langle D, \sqsubseteq, \bot, \top, \circ \rangle$ such that:
  - $\langle D, \sqsubseteq, \bot, \top \rangle$ is a lattice with extreme points $\bot$ and $\top$ w.r.t. $\sqsubseteq$.
  - $\circ : D \times D \rightarrow D$, called *attenuation operation*, verifies:
    (a) $\circ$ is associative, commutative and monotonic w.r.t. $\sqsubseteq$.
    (b) $\forall d \in D : d \circ \top = d$.
    (c) $\forall d \in D : d \circ \bot = \bot$.
    (d) $\forall d, e \in D \setminus \{ \bot, \top \} : d \circ e \sqsubseteq e$.
    (e) $\forall d, e_1, e_2 \in D : d \circ (e_1 \sqcap e_2) = d \circ e_1 \sqcap d \circ e_2$.

We generalize van Emden’s QLP to QLP$(\mathcal{D})$ where:
- qualification values $d \in D \setminus \{ \bot \}$ instead of $d \in (0, 1]$.
- the $\sqcap$ operator instead of $\min$.
- $\circ$ instead of $\times$ (product).
Qualification Domains: Axioms

- **Structure** $\mathcal{D} = \langle D, \sqsubseteq, \bot, \top, \circ \rangle$ such that:
  - $\langle D, \sqsubseteq, \bot, \top \rangle$ is a lattice with extreme points $\bot$ and $\top$ w.r.t. $\sqsubseteq$.
  - $\circ : D \times D \rightarrow D$, called *attenuation operation*, verifies:
    1. $\circ$ is associative, commutative and monotonic w.r.t. $\sqsubseteq$.
    2. $\forall d \in D : d \circ \top = d$.
    3. $\forall d \in D : d \circ \bot = \bot$.
    4. $\forall d, e \in D \setminus \{ \bot, \top \} : d \circ e \sqsubseteq e$.
    5. $\forall d, e_1, e_2 \in D : d \circ (e_1 \sqcap e_2) = d \circ e_1 \sqcap d \circ e_2$.

- We generalize *van Emden’s* QLP to QLP($\mathcal{D}$) where:
  - qualification values $d \in D \setminus \{ \bot \}$ instead of $d \in (0, 1]$.
  - the *glb* operator $\sqcap$ instead of $\min$.
  - $\circ$ instead of $\times$ (product).
Qualification Domains (and II): Instances

\[ B = (\{0, 1\}, \leq, 0, 1, \land) \] (Classical Boolean Values).
\[ U = ([0, 1], \leq, 0, 1, \times) \] (van Emden’s Certainty Degrees).
\[ W = ([0, \infty], \geq, \infty, 0, +) \] (Proof trees’ depths).

The cartesian product \[ D = D_1 \times D_2 \] of two given qualification domains is always another qualification domain.

Note: \[ U \times W \] qualifies both certainties and proof trees’ depths.
Qualified Logic Programming
Syntax: Programs

Program: set of qualified definite Horn clauses of the form

\[ A \leftarrow \alpha - B_1, \ldots, B_k \]

with \( \alpha \in D \setminus \{ \bot \} \).

Example 1

a) Instance \( \mathcal{U} \):
   cruel(X) <-0.9- human(X), eats(X,Y), animal(Y)

b) Instance \( \mathcal{W} \):
   cruel(X) <-2.0- human(X), eats(X,Y), animal(Y)

c) Instance \( \mathcal{U} \times \mathcal{W} \):
   cruel(X) <-\((0.9, 2.0)\)- human(X), eats(X,Y), animal(Y)
Syntax (II): Program Example

Example 2

domestic(cat) <-0.8-
intelligent(A) <-0.9- domestic(A)
Example 2

domestic(cat) <-0.8-

intelligent(A) <-0.9- domestic(A)

- Classic logic atoms.
Syntax (II): Program Example

Example 2

\[
\text{domestic(cat)} \leftarrow 0.8 - \\
\text{intelligent(A)} \leftarrow 0.9 - \text{domestic(A)}
\]

- Classic logic atoms.
- Attenuation factors (elements in \((0, 1]\)).
Syntax (II): Program Example

Example 2

domestic(cat) <-0.8-
intelligent(A) <-0.9- domestic(A)

- Classic logic atoms.
- Attenuation factors (elements in (0, 1]).
- Proving: intelligent(cat)#0.72.
Example 2

domestic(cat)#0.8 <-0.8-
intelligent(A) <-0.9- domestic(A)

- Classic logic atoms.
- Attenuation factors (elements in (0, 1]).
- Proving: intelligent(cat)#0.72.
- \[ 0.8 = 0.8 \times \min\{\} = 0.8 \times 1.0. \]
Example 2

domestic(cat)#0.8 <-0.8-
intelligent(cat) <-0.9- domestic(cat)#0.8

- Classic logic atoms.
- Attenuation factors (elements in \((0, 1]\)).
- Proving: intelligent(cat)#0.72.
- \(0.8 = 0.8 \times \min \{\} = 0.8 \times 1.0.\)
Syntax (II): Program Example

Example 2

domestic(cat)#0.8 <-0.8-
intelligent(cat)#0.72 <-0.9- domestic(cat)#0.8

- Classic logic atoms.
- Attenuation factors (elements in $(0,1]$).
- Proving: intelligent(cat)#0.72.
  - $0.8 = 0.8 \times \min \{\} = 0.8 \times 1.0$.
  - $0.72 = 0.9 \times \min \{0.8\}$. 
Syntax (and III): Interpretations and Models

- **D-qualified atom**: $A \sharp d$.

- **Open D-qualified atom**: $A \sharp W$ ($W$ is intended to take values over $D \setminus \{\bot\}$).

**D-entailment relation**: is defined as $A \sharp d \sqsupseteq_D A' \sharp d'$ iff there is some substitution $\theta$ such that $A' = A\theta$ and $d' \sqsubseteq d$.

**Interpretation over D**: is defined as any set $\mathcal{I}$ of D-qualified atoms which is closed under D-entailment. The set of all interpretations over $D$ is a complete lattice under $\subseteq$.

**Model**: $\mathcal{I}$ models $(A \leftarrow \alpha \leftarrow B_1, \ldots, B_k) \in \mathcal{P}$ iff:

for any $\theta$ and any $d_1, \ldots, d_k \in D \setminus \{\bot\}$ such that $B_i\theta \sharp d_i \in \mathcal{I}$, one has $A\theta \sharp (\alpha \circ \bigcap\{d_1, \ldots, d_k\}) \in \mathcal{I}$.

And $\mathcal{I}$ models $\mathcal{P}$ iff $\mathcal{I}$ is a model of each clause in $\mathcal{P}$.
Qualified Horn Logic over $\mathcal{D}$: is defined as a deductive system consisting just of one inference rule $\text{QMP}(\mathcal{D})$. If there is some instance rule $(A \leftarrow \alpha \leftarrow B_1, \ldots, B_k) \in [C]$ for some $C \in \mathcal{P}$ and $d \sqsubseteq \alpha \circ \prod\{d_1, \ldots, d_k\}$, the following inference step is allowed:

\[
\begin{array}{c}
B_1 \# d_1 & \cdots & B_k \# d_k \\
\hline
A \# d
\end{array}
\]
Declarative Semantics: Qualified Horn Logic

- **Qualified Horn Logic over** $\mathcal{D}$: is defined as a deductive system consisting just of one inference rule $\text{QMP}(\mathcal{D})$. If there is some instance rule $(A \leftarrow \alpha \neg B_1, \ldots, B_k) \in [C]$ for some $C \in \mathcal{P}$ and $d \sqsubseteq \alpha \circ \prod \{d_1, \ldots, d_k\}$, the following inference step is allowed:

\[
\begin{array}{c}
B_1 \not\! \not\! \not\! d_1 & \cdots & B_k \not\! \not\! \not\! d_k \\
\hline
A \not\! \not\! \not\! d
\end{array}
\]

- **Least Herbrand model of** $\mathcal{P}$:

\[
\mathcal{M}_\mathcal{P} = \{ A \not\! \not\! \not\! d \mid \mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} A \not\! \not\! \not\! d \}
\]
Example 3

cruel(X) <-0.90- human(X), eats(X,Y), animal(Y)
animal(bird) <-1.0-
human(eve) <-1.0-
eats(eve,X) <-0.30- animal(X)
eats(mother(X),Y) <-0.70- eats(X,Y)

\[\begin{align*}
\text{human(eve)} & \text{#1.0} \\
\text{human(mother(eve))} & \text{#0.90} \\
\text{eats(eve,bird)} & \text{#0.30} \\
\text{eats(mother(eve),bird)} & \text{#0.21} \\
\text{animal(bird)} & \text{#1.0} \\
\text{cruel(mother(eve))} & \text{#0.15}
\end{align*}\]

\[0.15 \leq 0.90 \times \min \{0.90, 0.21, 1.0\} = 0.189.\]
Goals

- Initial goals:

\[ \cdots, A_i \not\models W_i, \cdots \models \varepsilon \cdots, W_i \models \beta_i, \cdots \]
Goals

- Initial goals:
  \[ \cdots, A_i \not\models W_i, \cdots, \sigma \not\models \cdots, W_i \models \beta_i, \cdots \]

- General goals:
  \[ \cdots, A_i \not\models W_i, \cdots, W' = d \circ \bigcap \{ \overline{W'_m} \}, \cdots, d \circ W_i \models \beta_i, \cdots \]
Goals

- Initial goals:

\[ \cdots, A_i \not\equiv W_i, \cdots \quad \square \in \cdots, W_i \sqsubseteq \beta_i, \cdots \]

- General goals:

\[ \cdots, A_i \not\equiv W_i, \cdots \quad \square \sigma \quad \cdots, W' = d \circ \bigcap \{ \overline{W'_m} \}, \cdots, d \circ W_i \sqsubseteq \beta_i, \cdots \]

- Solved goals:

\[ \quad \square \sigma \quad \cdots, W' = d \circ \bigcap \{ \overline{W'_m} \}, \cdots \]
QSLD Resolution

- Resolution step:

\[
\cdots, A \# W, \cdots \mathrel{|} \sigma \mathrel{|} \cdots, d \circ W \supseteq \beta, \cdots \vdash_{C_1, \sigma_1}
\]

\[
(\cdots, B_1 \# W_1, \cdots, B_k \# W_k, \cdots)\sigma_1 \mathrel{|} \sigma_1 \mathrel{|} \cdots,
\]

\[
\alpha \circ d \circ W_1 \supseteq \beta, \cdots, \alpha \circ d \circ W_k \supseteq \beta, \cdots,
\]

\[
W = \alpha \circ \bigcap \{W_1, \ldots, W_k\}, \cdots
\]

where \( C_1 \equiv (H \leftarrow \alpha - B_1, \ldots, B_k) \in \mathcal{P} \) and \( \sigma_1 \) the m.g.u. \((A, H)\).

- Resolution computations:

\[
G_0 \vdash_{\sigma_1} G_1 \vdash_{\sigma_2} \cdots \vdash_{\sigma_n} G_n \equiv \sigma_1 \sigma_2 \cdots \sigma_n \mathrel{|} \Delta_n
\]

with \( G_0 \) initial and \( G_n \) solved, and the computed answer \((\sigma, \mu)\) where \( \sigma = \sigma_1 \sigma_2 \cdots \sigma_n \) and \( \mu(W) \) computed according to \( \Delta_n \).
Example 4

\[
eats(X,Y) \# \text{W} \mid \{\} \mid \text{W} \geq 0.20 \quad \vdash \eats.2,\{X \mapsto \text{mother}(X')\}
\]
QSLD Resolution (and II)

Example 4

\[
eats(X,Y) \# W \mid \{\} \mid W \geq 0.20
\]
\[
eats(X',Y) \# W_1 \mid \{X \leftarrow \text{mother}(X')\} \mid W = 0.70 \times \text{min } \{W_1\},
\]
\[
0.70 \times W_1 \geq 0.20
\]

\[\models \eats.2, \{X \leftarrow \text{mother}(X')\}\]

\[
eats(X',Y) \# W_1 \mid \{X \leftarrow \text{mother}(X')\} \mid W = 0.70 \times \text{min } \{W_1\},
\]
\[
0.70 \times W_1 \geq 0.20
\]

\[\models \eats.1, \{X' \leftarrow \text{eve}\}\]
Example 4

\[
eats(X, Y) \# W \mid \{\} \mid W \geq 0.20
\]
\[
eats(X', Y) \# W_1 \mid \{X \leftarrow \text{mother}(X')\} \mid W = 0.70 \times \min\{W_1\},
\]
\[
0.70 \times W_1 \geq 0.20
\]
\[
animal(Y) \# W_2 \mid \{X \leftarrow \text{mother(eve)}\} \mid W = 0.70 \times \min\{W_1\},
\]
\[
W_1 = 0.30
\]
\[
0.30 \times 0.70 \times W_2 \geq 0.20
\]
QSLD Resolution (and II)

Example 4

eats(X,Y)#W | {} | W >= 0.20
\[\vdash \text{eats.2,} \{X \mapsto \text{mother}(X')\}\]
eats(X',Y)#W1 | \{X \mapsto \text{mother}(X')\} |
W = 0.70 \times \text{min}\{W1\},
0.70 \times W1 >= 0.20
\[\vdash \text{eats.1,} \{X' \mapsto \text{eve}\}\]
animal(Y)#W2 | \{X \mapsto \text{mother}(\text{eve})\} |
W = 0.70 \times \text{min}\{W1\},
W1 = 0.30
0.30 \times 0.70 \times W2 >= 0.20
\[\vdash \text{animal,} \{Y \mapsto \text{bird}\}\]
\{X \mapsto \text{mother}(\text{eve}), Y \mapsto \text{bird}\} |
W = 0.70 \times \text{min}\{W1\},
W1 = 0.30
W2 = 1.0

Carlos A. Romero-Díaz (GPD / UCM)  Qualified Logic Programming  December, 2008  17 / 1
Example 4

\[
eats(X,Y)#W \mid \{\} \mid W \geq 0.20
\]
\[
eats(X',Y)#W_1 \mid \{X \mapsto \text{mother}(X')\} \mid
\]
\[
W = 0.70 \times \min\{W_1\},
\]
\[
0.70 \times W_1 \geq 0.20
\]
\[
\vdash \eats.2,\{X \mapsto \text{mother}(X')\}
\]
\[
animal(Y)#W_2 \mid \{X \mapsto \text{mother}(\text{eve})\} \mid
\]
\[
W = 0.70 \times \min\{W_1\},
\]
\[
W_1 = 0.30
\]
\[
0.30 \times 0.70 \times W_2 \geq 0.20
\]
\[
\vdash \eats.1,\{X' \mapsto \text{eve}\}
\]
\[
\vdash \animal,\{Y \mapsto \text{bird}\}
\]

Computed answer: \(\{X \mapsto \text{mother}(\text{eve}), Y \mapsto \text{bird}\} \land \{W \mapsto 0.21\}\)
Properties of QSLD Resolution

Definition of Solution: A pair of substitutions \((\theta, \rho)\) is called solution of a goal \(G \equiv \bar{A} \setminus \sigma \setminus \Delta\) iff:

(i) \(\theta = \sigma \theta\).

(ii) \(\rho \in \text{Sol}(\Delta)\).

(iii) \(\mathcal{P} \vdash_{Q\text{HL}(\mathcal{P})} A\theta \not\models W\rho\) for every annotated atom in \(\bar{A}\).
Properties of QSLD Resolution

- **Definition of Solution:** A pair of substitutions \((\theta, \rho)\) is called solution of a goal \(G \equiv \bar{A} \uparrow \sigma \uparrow \Delta\) iff:
  
  (i) \(\theta = \sigma \theta\).
  
  (ii) \(\rho \in Sol(\Delta)\).
  
  (iii) \(\mathcal{P} \vdash_{QHL(\mathcal{D})} A\theta \# W\rho\) for every annotated atom in \(\bar{A}\).

**Soundness:** Assume \(G_0 \vdash^* G\) and \(G = \sigma \uparrow \Delta\) solved. Let \((\sigma, \mu)\) be the computed answer to \(G\). Then \((\sigma, \mu)\) is a solution of \(G_0\).

**Strong Completeness:** Assume a given solution \((\theta, \rho)\) for \(G_0\) and any fixed strategy for choosing the selected atom at each resolution step. Then there is some computed answer \((\sigma, \mu)\) for \(G_0\) which is a more general solution than \((\theta, \rho)\).
Implementation: translating to \( \mathcal{T}\mathcal{O}\mathcal{Y} \)

Requirement \( CLP \) or \( CFLP \) system with support for \( C_D \) constraints.

\[
C \equiv a(t) \leftarrow \alpha - b_1(s_1), \ldots, b_k(s_k)
\]

\[
C^t \equiv a(t, D, W, Beta) \leftarrow \alpha \circ D \sqsupseteq Beta,
\]

\[
b_1(s_1, \alpha \circ D, W_1, Beta),
\]

\[
\vdots
\]

\[
b_k(s_k, \alpha \circ D, W_k, Beta),
\]

\[
W = \alpha \circ \cap\{W_1, \ldots, W_k\}
\]
Implementation: translating to \( \text{T O Y} \)

**Requirement** \( \text{CLP} \) or \( \text{CFLP} \) system with support for \( C_D \) constraints.

\[
C \equiv a(t) \leftarrow \alpha - b_1(s_1), \ldots, b_k(s_k)
\]

\[
C^t \equiv a(t, D, W, Beta) \leftarrow \alpha \circ D \sqsupset Beta,
\]

\[
b_1(s_1, \alpha \circ D, W_1, Beta),
\]

\[
\vdots
\]

\[
b_k(s_k, \alpha \circ D, W_k, Beta),
\]

\[
W = \alpha \circ \bigcap \{W_1, \ldots, W_k\}
\]

\[
G \equiv a_1(t_1) \# W_1, \ldots, a_m(t_m) \# W_m \sqsupset W_1 \sqsupseteq \beta_1, \ldots, W_m \sqsupseteq \beta_m
\]

\[
G^t \equiv a_1(t_1, \top, W_1, \beta_1), \ldots, a_m(t_m, \top, W_m, \beta_m)
\]
Similarity-based
Qualified Logic Programming
Similarity-based Reasoning

**Motivation:**
- Incorporating similarity-based reasoning in QLP.
- Defining a global scheme for programming with qualifications and similarities.
- Showing the possibility of reducing similarities to qualifications.
Similarity-based Reasoning

**Motivation:**
- Incorporating similarity-based reasoning in QLP.
- Defining a global scheme for programming with qualifications and similarities.
- Showing the possibility of reducing similarities to qualifications.

**Results:**
1. Declarative semantics for a Similarity-based Qualified Logic Programming (SQLP) scheme.
2. A SQLP program transformation to an equivalent QLP program ⇒ Goal Solving Procedure for QLP can be reused.
Similarity Relations

- Traditional approach in SLP: $\mathcal{R} : S \times S \rightarrow [0, 1]$ defined as a generalization of classical equivalence relations.
- $\mathcal{R}(x, y)$: similarity degree between $x$ and $y$.
- For instance: $\mathcal{R}(\text{farm}, \text{domestic}) = 0.3$. 
Similarity Relations

- Traditional approach in SLP: $\mathcal{R} : S \times S \rightarrow [0, 1]$ defined as a generalization of classical equivalence relations.
- $\mathcal{R}(x, y)$: similarity degree between $x$ and $y$.
- For instance: $\mathcal{R}($farm, domestic$) = 0.3$.

\textbf{$\mathcal{D}$-valued similarity relation over $S$:}

$$\mathcal{R} : S \times S \rightarrow \mathcal{D}$$

s.t. the following hold:

(a) Reflexivity: $\mathcal{R}(x, y) = \top$.
(b) Symmetry: $\mathcal{R}(x, y) = \mathcal{R}(y, x)$.
(c) Transitivity: $\mathcal{R}(x, z) \sqsupseteq \mathcal{R}(x, y) \sqcap \mathcal{R}(y, z)$.
Admissible relation $\mathcal{R}$ over a set of symbols $S$

- If $S = \mathcal{V}ar \cup CS \cup PS$, and:

$$
\begin{align*}
R(x, y) \neq \perp & \Rightarrow (a) x \text{ and } y \text{ are the same variable, or} \\
 & \text{(b) } x, y \in CS^n \text{ (for arity } n \geq 0), \text{ or} \\
 & \text{(c) } x, y \in PS^n \text{ (for arity } n \geq 0). 
\end{align*}
$$
Admissible relation $\mathcal{R}$ over a set of symbols $S$

If $S = \mathcal{V}ar \cup CS \cup PS$, and:

- $\mathcal{R}$ restricted to $\mathcal{V}ar$: $\mathcal{R}(X, X) = \top$ and $\mathcal{R}(X, Y) = \bot$ if $X \neq Y$. 
Admissible relation $\mathcal{R}$ over a set of symbols $S$

- If $S = \mathcal{V} ar \cup CS \cup PS$, and:
  - $\mathcal{R}$ restricted to $\mathcal{V} ar$: $\mathcal{R}(X,X) = \top$ and $\mathcal{R}(X,Y) = \bot$ if $X \neq Y$.
  - $\mathcal{R}(x,y) \neq \bot \Rightarrow$
    - (a) $x$ and $y$ are the same variable, or
    - (b) $x, y \in CS^n$ (for arity $n \geq 0$), or
    - (c) $x, y \in PS^n$ (for arity $n \geq 0$).
Admissible relation $\mathcal{R}$ over a set of symbols $S$

- If $S = \mathcal{V}ar \cup CS \cup PS$, and:
  - $\mathcal{R}$ restricted to $\mathcal{V}ar$: $\mathcal{R}(X, X) = \top$ and $\mathcal{R}(X, Y) = \bot$ if $X \neq Y$.
  - $\mathcal{R}(x, y) \neq \bot \Rightarrow$
    - (a) $x$ and $y$ are the same variable, or
    - (b) $x, y \in CS^n$ (for arity $n \geq 0$), or
    - (c) $x, y \in PS^n$ (for arity $n \geq 0$).

**Definition: Extension of $\mathcal{R}$ to act over terms.**

- $\mathcal{R}(X, X) = \top$ and $\mathcal{R}(X, t) = \mathcal{R}(t, X) = \bot$ if $X \neq t$. 
Admissible relation $\mathcal{R}$ over a set of symbols $S$

- If $S = \mathcal{V}ar \cup CS \cup PS$, and:
  - $\mathcal{R}$ restricted to $\mathcal{V}ar$: $\mathcal{R}(X, X) = \top$ and $\mathcal{R}(X, Y) = \bot$ if $X \neq Y$.
  - $\mathcal{R}(x, y) \neq \bot \Rightarrow$
    - (a) $x$ and $y$ are the same variable, or
    - (b) $x, y \in CS^n$ (for arity $n \geq 0$), or
    - (c) $x, y \in PS^n$ (for arity $n \geq 0$).

Definition: Extension of $\mathcal{R}$ to act over terms.

- $\mathcal{R}(X, X) = \top$ and $\mathcal{R}(X, t) = \mathcal{R}(t, X) = \bot$ if $X \neq t$.
- $\mathcal{R}(c(\overline{t_n}), c'(\overline{t_m})) = \bot$ if $n \neq m$. 
Admissible relation $\mathcal{R}$ over a set of symbols $S$

- If $S = \text{Var} \cup CS \cup PS$, and:
  
  - $\mathcal{R}$ restricted to $\text{Var}$: $\mathcal{R}(X, X) = \top$ and $\mathcal{R}(X, Y) = \bot$ if $X \neq Y$.
  
  - $\mathcal{R}(x, y) \neq \bot \Rightarrow$
    
    (a) $x$ and $y$ are the same variable, or
    
    (b) $x, y \in CS^n$ (for arity $n \geq 0$), or
    
    (c) $x, y \in PS^n$ (for arity $n \geq 0$).

**Definition: Extension of $\mathcal{R}$ to act over terms.**

- $\mathcal{R}(X, X) = \top$ and $\mathcal{R}(X, t) = \mathcal{R}(t, X) = \bot$ if $X \neq t$.
- $\mathcal{R}(c(t^n_n), c'(t'_m^m)) = \bot$ if $n \neq m$.
- $\mathcal{R}(c(t^n_n), c'(t'_n^n)) = \mathcal{R}(c, c') \sqcap \mathcal{R}(t_1, t'_1) \sqcap \ldots \sqcap \mathcal{R}(t_n, t'_n)$.

(Analogously for atoms and clauses).
Example 5

farm(cow) <-1.0-
pacific(A) <-0.9- domestic(A)
\( R(farm, domestic) = 0.3 \)
Example 5

farm(cow) <-1.0-
pacific(A) <-0.9- domestic(A)

$\mathcal{R}(farm, domestic) = 0.3$

- A QLP($\mathcal{U}$) program.
Example 5

\[
farm(cow) \leftarrow 1.0 - \\
pacific(A) \leftarrow 0.9 - \text{domestic}(A)
\]

\[\mathcal{R}(\text{farm,domestic}) = 0.3\]

- A QLP(\mathcal{U}) program and an \textit{admissible} similarity relation \(\mathcal{R}\).
Example 5

\[ \text{farm(cow)} \leftarrow 1.0 - \]
\[ \text{pacific(A)} \leftarrow 0.9 - \text{domestic(A)} \]
\[ \mathcal{R}(\text{farm}, \text{domestic}) = 0.3 \]

- A QLP(\(\mathcal{U}\)) program and an \textit{admissible} similarity relation \(\mathcal{R}\).
- Proving: \text{pacific(cow)} \# 0.27.
Similarity-based Qualified Logic Programming

Example 5

\[
\begin{align*}
\text{farm(cow)} & \leftarrow 1.0 - \\
\text{pacific(A)} & \leftarrow 0.9 - \text{domestic(A)} \\
R(\text{farm}, \text{domestic}) & = 0.3
\end{align*}
\]

- A QLP(\mathcal{U}) program and an admissible similarity relation \( R \).
- Proving: \( \text{pacific(cow)} \# 0.27 \).
- \( (\text{domestic(cow)} \leftarrow 1.0 - , 0.3) \in [\text{farm(cow)} \leftarrow 1.0 - ]_R \).
Similarity-based Qualified Logic Programming

Example 5

\[
\text{farm(cow) } \leftarrow 1.0 - \\
\text{pacific(cow) } \leftarrow 0.9 - \text{ domestic(cow) } \leq 0.3
\]

\[\mathcal{R}(\text{farm,domestic}) = 0.3\]

- A QLP(\(U\)) program and an admissible similarity relation \(\mathcal{R}\).
- Proving: \text{pacific(cow)} \leq 0.27.
- \((\text{domestic(cow) } \leftarrow 1.0 - , 0.3) \in [\text{farm(cow) } \leftarrow 1.0 - ]_{\mathcal{R}}\).
- \(0.3 = 1.0 \times \text{min}\{0.3\} \).
Example 5

farm(cow) <-1.0-
pacific(cow)#0.27 <-0.9- domestic(cow) #0.3

$\mathcal{R}(farm, domestic) = 0.3$

- A QLP($\mathcal{U}$) program and an admissible similarity relation $\mathcal{R}$.
- Proving: pacific(cow)#0.27.
- $(domestic(cow) <-1.0-, 0.3) \in [farm(cow) <-1.0-]_{\mathcal{R}}$.
- $0.3 = 1.0 \times \min \{0.3\}$.
- $0.27 = 0.9 \times \min \{1.0, 0.3\}$. 
Similarity-based Qualified Horn Logic over \((\mathcal{R}, \mathcal{D})\): If there is some \(((A \leftarrow \alpha \leftarrow B_1, \ldots, B_k), \delta) \in [C]_\mathcal{R}\) for some \(C \in \mathcal{P}\). Then, for any \(d \in D \setminus \{\bot\}\) s.t. \(d \sqsubseteq \alpha \circ \bigcap\{\delta, d_1, \ldots, d_k\}\):

\[
\begin{align*}
B_1 &\nmid d_1 & \cdots & B_k &\nmid d_k \\
\hline
A &\nmid d
\end{align*}
\]
**Similarity-based Qualified Horn Logic over \((\mathcal{R}, \mathcal{D})\):** If there is some \(((A \leftarrow \alpha - B_1, \ldots, B_k), \delta) \in [C]_\mathcal{R}\) for some \(C \in \mathcal{P}\). Then, for any \(d \in D \setminus \{\bot\}\) s.t. \(d \sqsubseteq \alpha \circ \bigcap \{\delta, d_1, \ldots, d_k\}\):

\[
\begin{array}{cccc}
B_1 \not\sqsupseteq d_1 & \cdots & B_k \not\sqsupseteq d_k \\
\hline
A \not\sqsupseteq d
\end{array}
\]

**Least Herbrand model of \(\mathcal{P}\):**

\[
\{ A \not\sqsupseteq d \mid \mathcal{P} \models_{SQHL(\mathcal{R}, \mathcal{D})} A \not\sqsupseteq d \}
\]
Goal Solving in SQLP

- Two possibilities:
  1. Change unification to similarity-based unification (done in some SLP works).
  2. Reduce a SQLP program to an equivalent QLP one and solve there (some transformations from SLP to LP proposed by other authors).
Goal Solving in SQLP

- Two possibilities:
  1. Change unification to similarity-based unification (done in some SLP works).
  2. Reduce a SQLP program to an equivalent QLP one and solve there (some transformations from SLP to LP proposed by other authors).

- We take possibility (2), so

  \[
  \text{SQLP}(\mathcal{R}, \mathcal{D}) \text{ program } \quad \mathcal{P} \quad \Rightarrow \quad \text{QLP}(\mathcal{D}) \text{ program } \quad S_{\mathcal{R}}(\mathcal{P})
  \]
Goal Solving in SQLP

- Two possibilities:
  1. Change unification to similarity-based unification (done in some SLP works).
  2. Reduce a SQLP program to an equivalent QLP one and solve there (some transformations from SLP to LP proposed by other authors).

- We take possibility (2), so

\[
\text{QLP}(\mathcal{D}) \text{ program } \quad \mathcal{P} \quad \Rightarrow \quad S_{\mathcal{R}}(\mathcal{P})
\]

Proving as main result that

\[
\mathcal{P} \vdash_{\text{SQHL}(\mathcal{R}, \mathcal{D})} A \# d \iff S_{\mathcal{R}}(\mathcal{P}) \vdash_{\text{QHL}(\mathcal{D})} A \# d.
\]
Reducing Similarities to Qualifications: A Program Transformation

\[ S_R(P) = P_S \cup P_\sim \cup P_{\text{pay}} \]
Reducing Similarities to Qualifications: A Program Transformation

\[ S_R(\mathcal{P}) = \mathcal{P}_S \cup \mathcal{P}_\sim \cup \mathcal{P}_{pay} \]

**\( \mathcal{P}_S \)** For every \((H \leftarrow \alpha - \overline{B}) \in \mathcal{P}\) and for each \(H'\) such that \(R(H_l, H') \neq \bot\):

\[(H' \leftarrow \alpha - \text{pay}_{R(H_l, H')}, S_l, \overline{B}) \in \mathcal{P}_S\]

where \((H_l, S_l) = \text{lin}(H)\).

Linearization of clause heads is a novelty in our approach.
Why to linearize?

- To let equal variables in the head atom take similar (and not only the same) values:
Why to linearize?

- To let equal variables in the head atom take similar (and not only the same) values:

Example 6

\[ \mathcal{P} \equiv p(x, x) \leftarrow 1.0-, \quad \mathcal{R}(c, d) = 0.7 \]
Why to linearize?

- To let equal variables in the head atom take similar (and not only the same) values:

Example 6

\[ \mathcal{P} \equiv p(X, X) \leftarrow 1.0, \quad \mathcal{R}(c, d) = 0.7 \]

We want to prove: \( p(c, c) \# 1.0 \) and \( p(d, d) \# 1.0 \),
Why to linearize?

- To let equal variables in the head atom take similar (and not only the same) values:

Example 6

\[ \mathcal{P} \equiv p(X, X) \leftarrow 1.0-, \quad \mathcal{R}(c, d) = 0.7 \]

We want to prove: \( p(c, c) \neq 1.0 \) and \( p(d, d) \neq 1.0 \), and also \( p(c, d) \neq 0.7 \) and \( p(d, c) \neq 0.7 \).
Why to linearize?

- To let equal variables in the head atom take similar (and not only the same) values:

Example 6

\[ P \equiv p(X, X) \leftarrow 1.0-, \ R(c, d) = 0.7 \]

We want to prove: \( p(c, c) \neq 1.0 \) and \( p(d, d) \neq 1.0 \), and also \( p(c, d) \neq 0.7 \) and \( p(d, c) \neq 0.7 \).

Definition: \( lin(H) = (H_\ell, S_\ell) \)
Why to linearize?

- To let equal variables in the head atom take similar (and not only the same) values:

Example 6

\[
\mathcal{P} \equiv p(X, X) \leftarrow 1.0 - , \quad \mathcal{R}(c, d) = 0.7
\]

We want to prove: \( p(c, c) \neq 1.0 \) and \( p(d, d) \neq 1.0 \), and also \( p(c, d) \neq 0.7 \) and \( p(d, c) \neq 0.7 \).

Definition: \( \text{lin}(H) = (H_\ell, S_\ell) \)

- \( H_\ell \): linear version of \( H \) (now without repeated variables).
Why to linearize?

- To let equal variables in the head atom take similar (and not only the same) values:

### Example 6

\[ P \equiv p(X, X) \leftarrow 1.0-, \quad \mathcal{R}(c, d) = 0.7 \]

We want to prove: \( p(c, c) \neq 1.0 \) and \( p(d, d) \neq 1.0 \), and also \( p(c, d) \neq 0.7 \) and \( p(d, c) \neq 0.7 \).

### Definition: \( \text{lin}(H) = (H_\ell, S_\ell) \)

- \( H_\ell \): linear version of \( H \) (now without repeated variables).
- \( S_\ell \): set of *similarity conditions* \( X \sim X_i \) \((1 \leq i \leq n)\) between the variables that replaced the \( n \) additional occurrences of the same variable in \( H \).
Why to linearize?

- To let equal variables in the head atom take similar (and not only the same) values:

**Example 6**

\[ P \equiv p(X, X) \leftarrow 1.0, \quad R(c, d) = 0.7 \]

We want to prove: \( p(c, c) \neq 1.0 \) and \( p(d, d) \neq 1.0 \), and also \( p(c, d) \neq 0.7 \) and \( p(d, c) \neq 0.7 \).

**Definition:** \( lin(H) = (H_\ell, S_\ell) \)

- \( H_\ell \): linear version of \( H \) (now without repeated variables).
- \( S_\ell \): set of *similarity conditions* \( X \sim X_i \ (1 \leq i \leq n) \) between the variables that replaced the \( n \) additional occurrences of the same variable in \( H \).

Example: \( lin(p(X, X)) = (p(X, X_1), \ \{X \sim X_1\}) \).
Reducing Similarities to Qualifications:
A Program Transformation

- $S_{\mathcal{R}}(\mathcal{P}) = \mathcal{P}_S \cup \mathcal{P}_{\sim} \cup \mathcal{P}_{\text{pay}}$

$\mathcal{P}_S$ For every $(H \leftarrow \alpha - B) \in \mathcal{P}$ and for each $H'$ such that $\mathcal{R}(H_\ell, H') \neq \perp$:

$$(H' \leftarrow \alpha - \text{pay}_{\mathcal{R}(H_\ell, H')}, S_\ell, B) \in \mathcal{P}_S$$

where $(H_\ell, S_\ell) = \text{lin}(H)$.

Linearization of clause heads is a novelty in our approach.
Reducing Similarities to Qualifications:
A Program Transformation

- \( S_R(\mathcal{P}) = \mathcal{P}_S \cup \mathcal{P}_\sim \cup \mathcal{P}_{\text{pay}} \)

\( \mathcal{P}_S \) For every \((H \leftarrow \alpha - \overline{B}) \in \mathcal{P} \) and for each \(H'\) such that \(\mathcal{R}(H_\ell, H') \neq \bot\):

\[ (H' \leftarrow \alpha - \text{pay}_{\mathcal{R}(H_\ell, H')}, S_\ell, \overline{B}) \in \mathcal{P}_S \]

where \((H_\ell, S_\ell) = \text{lin}(H)\).

Linearization of clause heads is a novelty in our approach.

\( \mathcal{P}_\sim = \left\{ X \sim X \leftarrow \top \right\} \)

\[ \cup \left\{ (c(X_n) \sim c' (Y_n) \leftarrow \top - \text{pay}_{\mathcal{R}(c,c')}, X_n \sim Y_n) \mid c, c' \in CS^n, \mathcal{R}(c, c') \neq \bot \right\} \].
Reducing Similarities to Qualifications: A Program Transformation

- \( S_{R}(\mathcal{P}) = \mathcal{P}_S \cup \mathcal{P}_{\sim} \cup \mathcal{P}_{\text{pay}} \)

**\( \mathcal{P}_S \)**
For every \((H \leftarrow \alpha - \overline{B}) \in \mathcal{P}\) and for each \(H'\) such that \(R(H, H') \neq \bot\):

\[
(H' \leftarrow \alpha - \text{pay}_{R(H, H')}, S, \overline{B}) \in \mathcal{P}_S
\]

where \((H, S) = \text{lin}(H)\).

Linearization of clause heads is a novelty in our approach.

**\( \mathcal{P}_{\sim} \)**

\[
\mathcal{P}_{\sim} = \{X \sim X \leftarrow \top-\}
\]

\[
\cup \{(c(X_n) \sim c'(Y_n) \leftarrow \top - \text{pay}_{R(c, c')}, \overline{X_n} \sim \overline{Y_n}) | c, c' \in CS^n, R(c, c') \neq \bot\}.
\]

**\( \mathcal{P}_{\text{pay}} \)**

\[
\mathcal{P}_{\text{pay}} = \{\text{pay}_w \leftarrow w- | \text{for each atom pay}_w \text{ occurring in } \mathcal{P}_S \cup \mathcal{P}_{\sim}\}.
\]
Program transformation in action

Example 7: SQLP(\mathcal{R},\mathcal{U}) program

\text{farm(cow) <-1.0-}
\text{domestic(cat) <-0.8-}
\text{intelligent(A) <-0.9- domestic(A)}

\mathcal{R}(farm,\text{domestic}) = 0.3
\mathcal{R}(\text{cat,lynx}) = 0.8
Example 7

* farm(cow) <-1.0-
  domestic(cat) <-0.8-
  intelligent(A) <-0.9- domestic(A)

\[ R(farm,\text{domestic}) = 0.3 \]
\[ R(cat,\text{lynx}) = 0.8 \]
Example 7

\[ \text{farm(cow)} \leftarrow 1.0 \cdot \text{pay}_{1.0} \]
\[ \ast \text{farm(cow)} \leftarrow 1.0 \]
\[ \text{domestic(cat)} \leftarrow 0.8 \]
\[ \text{intelligent(A)} \leftarrow 0.9 \cdot \text{domestic(A)} \]

\[ R(\text{farm,domestic}) = 0.3 \]
\[ R(\text{cat,lynx}) = 0.8 \]
Example 7

farm(cow) <-1.0- pay1.0
* farm(cow) <-1.0-
domestic(cat) <-0.8-
in intelligent(A) <-0.9- domestic(A)

R(farm,domestic) = 0.3
R(cat,lynx) = 0.8
Example 7

farm(cow) <-1.0- pay₁₀
domestic(cow) <-1.0- pay₀₃
* farm(cow) <-1.0-
domestic(cat) <-0.8-
intelligent(A) <-0.9- domestic(A)

\( R(farm, domestic) = 0.3 \)
\( R(cat, lynx) = 0.8 \)
Program transformation in action

Example 7

farm(cow) <-1.0- pay_{1.0}
domestic(cow) <-1.0- pay_{0.3}
* domestic(cat) <-0.8-
intelligent(A) <-0.9- domestic(A)

\[ R(\text{farm,domestic}) = 0.3 \]
\[ R(\text{cat,lynx}) = 0.8 \]
Example 7

farm(cow) <- 1.0 - pay_{1.0}
domestic(cow) <- 1.0 - pay_{0.3}
* domestic(cat) <- 0.8 -
intelligent(A) <- 0.9 - domestic(A)

R(farm, domestic) = 0.3
R(cat, lynx) = 0.8
Program transformation in action

Example 7

farm(cow) <- 1.0 - pay_{1.0}
domestic(cow) <- 1.0 - pay_{0.3}
domestic(cat) <- 0.8 - pay_{1.0}
* domestic(cat) <- 0.8 -
intelligent(A) <- 0.9 - domestic(A)

R(farm,domestic) = 0.3
R(cat,lynx) = 0.8
Program transformation in action

Example 7

farm(cow) <- 1.0 - pay_{1.0}
domestic(cow) <- 1.0 - pay_{0.3}
domestic(cat) <- 0.8 - pay_{1.0}
* domestic(cat) <- 0.8 -
intelligent(A) <- 0.9 - domestic(A)

\( R(farm, domestic) = 0.3 \)
\( R(cat, lynx) = 0.8 \)
Program transformation in action

Example 7

farm(cow) <- 1.0 - pay
farm(cat) <- 0.8 - pay

domestic(cow) <- 1.0 - pay
domestic(cat) <- 0.8 - pay

* domestic(cat) <- 0.8 -
intelligent(A) <- 0.9 - domestic(A)

R(farm, domestic) = 0.3
R(cat, lynx) = 0.8
Example 7

farm(cow) <- 1.0 - pay1.0
farm(cat) <- 0.8 - pay0.3
domestic(cow) <- 1.0 - pay0.3
domestic(cat) <- 0.8 - pay1.0
* domestic(cat) <- 0.8 -
intelligent(A) <- 0.9 - domestic(A)

\[ R(\text{farm}, \text{domestic}) = 0.3 \]
\[ R(\text{cat}, \text{lynx}) = 0.8 \]
Program transformation in action

### Example 7

\[
\begin{align*}
\text{farm(cow)} & \leftarrow 1.0 - \text{pay}_{1.0} \\
\text{farm(cat)} & \leftarrow 0.8 - \text{pay}_{0.3} \\
\text{domestic(cow)} & \leftarrow 1.0 - \text{pay}_{0.3} \\
\text{domestic(cat)} & \leftarrow 0.8 - \text{pay}_{1.0} \\
\text{domestic(lynx)} & \leftarrow 0.8 - \text{pay}_{0.8} \\
\ast \text{ domestic(cat)} & \leftarrow 0.8 - \\
\text{intelligent(A)} & \leftarrow 0.9 - \text{domestic(A)} \\
\end{align*}
\]

\[\mathcal{R}(\text{farm,domestic}) = 0.3\]
\[\mathcal{R}(\text{cat,lynx}) = 0.8\]
Example 7

farm(cow) <- 1.0 - pay_{1.0}
farm(cat) <- 0.8 - pay_{0.3}
domestic(cow) <- 1.0 - pay_{0.3}
domestic(cat) <- 0.8 - pay_{1.0}
domestic(lynx) <- 0.8 - pay_{0.8}
* domestic(cat) <- 0.8 - pay_{0.8}
intelligent(A) <- 0.9 - domestic(A)

\[ R(farm, domestic) = 0.3 \]
\[ R(cat, lynx) = 0.8 \]
Example 7

farm(cow) <-1.0- pay1.0
farm(cat) <-0.8- pay0.3
farm(lynx) <-0.8- pay0.3
domestic(cow) <-1.0- pay0.3
domestic(cat) <-0.8- pay1.0
domestic(lynx) <-0.8- pay0.8
* domestic(cat) <-0.8-
intelligent(A) <-0.9- domestic(A)

$R(farm,domestic) = 0.3$
$R(cat,lynx) = 0.8$
Program transformation in action

Example 7

\[
\begin{align*}
\text{farm}(\text{cow}) & \leftarrow 1.0 - \text{pay}_{1.0} \\
\text{farm}(\text{cat}) & \leftarrow 0.8 - \text{pay}_{0.3} \\
\text{farm}(\text{lynx}) & \leftarrow 0.8 - \text{pay}_{0.3} \\
\text{domestic}(\text{cow}) & \leftarrow 1.0 - \text{pay}_{0.3} \\
\text{domestic}(\text{cat}) & \leftarrow 0.8 - \text{pay}_{1.0} \\
\text{domestic}(\text{lynx}) & \leftarrow 0.8 - \text{pay}_{0.8} \\
\ast \text{intelligent}(A) & \leftarrow 0.9 - \text{domestic}(A) \\
R(\text{farm}, \text{domestic}) & = 0.3 \\
R(\text{cat}, \text{lynx}) & = 0.8
\end{align*}
\]
Program transformation in action

Example 7

\[
\begin{align*}
\text{farm}(\text{cow}) & \leftarrow 1.0 - \text{pay}_{1.0} \\
\text{farm}(\text{cat}) & \leftarrow 0.8 - \text{pay}_{0.3} \\
\text{farm}(\text{lynx}) & \leftarrow 0.8 - \text{pay}_{0.3} \\
\text{domestic}(\text{cow}) & \leftarrow 1.0 - \text{pay}_{0.3} \\
\text{domestic}(\text{cat}) & \leftarrow 0.8 - \text{pay}_{1.0} \\
\text{domestic}(\text{lynx}) & \leftarrow 0.8 - \text{pay}_{0.8} \\
\text{* intelligent}(A) & \leftarrow 0.9 - \text{domestic}(A) \\
\end{align*}
\]

\[\mathcal{R}(\text{farm,domestic}) = 0.3\]
\[\mathcal{R}(\text{cat,lynx}) = 0.8\]
Program transformation in action

Example 7

farm(cow) <-1.0- pay1.0
farm(cat) <-0.8- pay0.3
farm(lynx) <-0.8- pay0.3
domestic(cow) <-1.0- pay0.3
domestic(cat) <-0.8- pay1.0
domestic(lynx) <-0.8- pay0.8
intelligent(A) <-0.9- pay1.0,domestic(A)
* intelligent(A) <-0.9- domestic(A)

R(farm,domestic) = 0.3
R(cat,lynx) = 0.8
Program transformation in action

Example 7

farm(cow) <-1.0- pay1.0
farm(cat) <-0.8- pay0.3
farm(lynx) <-0.8- pay0.3
domestic(cow) <-1.0- pay0.3
domestic(cat) <-0.8- pay1.0
domestic(lynx) <-0.8- pay0.8
intelligent(A) <-0.9- pay1.0,domestic(A)
Example 7

farm(cow) <-1.0- pay1.0
farm(cat) <-0.8- pay0.3
farm(lynx) <-0.8- pay0.3
domestic(cow) <-1.0- pay0.3
domestic(cat) <-0.8- pay1.0
domestic(lynx) <-0.8- pay0.8
intelligent(A) <-0.9- pay1.0,domestic(A)

X ~ X <-1.0-
cat ~ lynx <-1.0- pay0.8
Example 7

farm(cow) <- 1.0 - pay
farm(cat) <- 0.8 - pay
farm(lynx) <- 0.8 - pay
domestic(cow) <- 1.0 - pay
domestic(cat) <- 0.8 - pay
domestic(lynx) <- 0.8 - pay
intelligent(A) <- 0.9 - pay, domestic(A)

X ~ X <- 1.0 -
cat ~ lynx <- 1.0 - pay

pay <- 1.0 -
pay <- 0.8 -
pay <- 0.3 -
Program transformation in action

Example 7: QLP(\(\mathcal{U}\)) program

\[
\begin{align*}
\text{farm}(\text{cow}) & \leftarrow -1.0 - \text{pay}_{1.0} \\
\text{farm}(\text{cat}) & \leftarrow -0.8 - \text{pay}_{0.3} \\
\text{farm}(\text{lynx}) & \leftarrow -0.8 - \text{pay}_{0.3} \\
\text{domestic}(\text{cow}) & \leftarrow -1.0 - \text{pay}_{0.3} \\
\text{domestic}(\text{cat}) & \leftarrow -0.8 - \text{pay}_{1.0} \\
\text{domestic}(\text{lynx}) & \leftarrow -0.8 - \text{pay}_{0.8} \\
\text{intelligent}(A) & \leftarrow -0.9 - \text{pay}_{1.0}, \text{domestic}(A) \\
X \sim X & \leftarrow -1.0 - \\
\text{cat} \sim \text{lynx} & \leftarrow -1.0 - \text{pay}_{0.8} \\
\text{pay}_{1.0} & \leftarrow -1.0 - \\
\text{pay}_{0.8} & \leftarrow -0.8 - \\
\text{pay}_{0.3} & \leftarrow -0.3 - 
\end{align*}
\]
Improving the semantics
Improveing the semantics

Motivation:

- Incorporating to QLP a simple treatment of negation.
- Adding further expressivity to QLP with threshold constraints in clause bodies.
Improving the semantics

**Motivation:**
- Incorporating to QLP a simple treatment of negation.
- Adding further expressivity to QLP with threshold constraints in clause bodies.

**Results:**
1. An adapted declarative semantics for a Qualified Logic Programming with Bivalued Predicates (BQLP) scheme.
2. Improved expressivity (from QLP) with a simple kind of negation and threshold constraints in clause bodies.
3. Sound and complete goal solving procedure (via QSLD($\mathcal{D}$) resolution).
4. Easy implementation based on the existent QLP implementation.
Example 8

animal(bird)#tt <-1.0-
eats(eve,X)#ff <-0.7- animal(X)#(tt,?)
eats(ft(X),Y)#ff <-0.8- eats(X,Y)#(ff,0.4)
BQLP Program Example

Example 8

- Marked atoms, \( m \in \{tt, ff\} \).

\[
\begin{align*}
\text{animal(bird)} & \text{#tt } \leftarrow 1.0 \\
\text{eats(eve,X)} & \text{#ff } \leftarrow 0.7 & \text{animal(X)} & \text{#(tt,?)} \\
\text{eats(ft(X),Y)} & \text{#ff } \leftarrow 0.8 & \text{eats(X,Y)} & \text{#(ff,0.4)}
\end{align*}
\]
BQLP Program Example

Example 8

animal(bird)#tt <-1.0-
eats(eve,X)#ff <-0.7- animal(X)#(tt,?)
eats(ft(X),Y)#ff <-0.8- eats(X,Y)#(ff,0.4)

- Marked atoms, \( m \in \{ tt, ff \} \).
- Annotated atoms, \((m, w)\) such that \( m \in \{ tt, ff \}\) and \( w \in (D \setminus \{ \bot \}) \cup \{?\}\).
BQLP Program Example

Example 8

animal(bird)#tt <-1.0-
eats(eve,X)#ff <-0.7- animal(X)#(tt,?)
eats(ft(X),Y)#ff <-0.8- eats(X,Y)#(ff,0.4)

- Marked atoms, \( m \in \{\text{tt, ff}\} \).
- Annotated atoms, \((m, w)\) such that \( m \in \{\text{tt, ff}\} \) and \( w \in (D \setminus \{\bot\}) \cup \{?\} \).
- Proving: \( \text{eats(ft(eve), bird)}#(ff,0.5) \) (qualified atom, \((m, d)\) such that \( m \in \{\text{tt, ff}\} \) and \( d \in D \setminus \{\bot\} \)).
Marked atoms, \( m \in \{tt, ff\} \).

Annotated atoms, \((m, w)\) such that \( m \in \{tt, ff\} \) and \( w \in (D \setminus \{\bot\}) \cup \{?\} \).

Proving: \( \text{eats}(ft(eve), \text{bird})#(ff, 0.5) \) (qualified atom, \((m, d)\) such that \( m \in \{tt, ff\} \) and \( d \in D \setminus \{\bot\} \)).

\[ 1.0 = 1.0 \times \min \{\} = 1.0 \times 1.0. \]
Example 8

animal(bird)#(tt,1.0) <-1.0-

\[ \text{eats(eve,bird)#ff <-0.7- animal(bird)#(tt,1.0)} \]

\[ \text{eats(ft(eve),bird)#ff <-0.8- eats(eve,bird)#(ff,0.4)} \]

Marked atoms, \( m \in \{ tt, ff \} \).

Annotated atoms, \((m, w)\) such that \( m \in \{ tt, ff \} \) and \( w \in (D \setminus \{ \bot \}) \cup \{ ? \} \).

Proving: eats(ft(eve),bird)#(ff,0.5) (qualified atom, \((m, d)\) such that \( m \in \{ tt, ff \} \) and \( d \in D \setminus \{ \bot \} \)).

\[ 1.0 = 1.0 \times \min \{ \} = 1.0 \times 1.0. \]
Example 8

animal(bird)#(tt,1.0) <-1.0-
eats(eve,bird)#(ff,0.7) <-0.7- animal(bird)#(tt,1.0)
eats(ft(eve),bird)#ff <-0.8- eats(eve,bird)#(ff,0.4)

- Marked atoms, \( m \in \{ tt, ff \} \).
- Annotated atoms, \((m, w)\) such that \( m \in \{ tt, ff \} \) and \( w \in (D \setminus \{ \bot \}) \cup \{ ? \} \).
- Proving: \( eats(ft(eve),bird)#(ff,0.5) \) (qualified atom, \((m, d)\) such that \( m \in \{ tt, ff \} \) and \( d \in D \setminus \{ \bot \} \)).
  - \( 1.0 = 1.0 \times \min \{ \} = 1.0 \times 1.0 \).
  - \( 0.7 = 0.7 \times \min \{ 1.0 \} \).
Example 8

\begin{align*}
\text{animal}(\text{bird})#(tt,1.0) & \leftarrow -1.0- \\
\text{eats}(\text{eve},\text{bird})#(ff,0.7) & \leftarrow -0.7- \text{animal}(\text{bird})#(tt,1.0) \\
\text{eats}(\text{ft}(\text{eve}),\text{bird})#ff & \leftarrow -0.8- \text{eats}(\text{eve},\text{bird})#(ff,0.7)
\end{align*}

- Marked atoms, $m \in \{tt, ff\}$.
- Annotated atoms, $(m, w)$ such that $m \in \{tt, ff\}$ and $w \in (D \setminus \{\bot\}) \cup \{?\}$.
- Proving: $\text{eats}(\text{ft}(\text{eve}),\text{bird})#(ff,0.5)$ (qualified atom, $(m, d)$ such that $m \in \{tt, ff\}$ and $d \in D \setminus \{\bot\}$).
- $1.0 = 1.0 \times \min \{\} = 1.0 \times 1.0$.
- $0.7 = 0.7 \times \min \{1.0\}$.  

BQLP Program Example

Example 8

\[
\begin{align*}
\text{animal(bird)#(tt,1.0)} & \leftarrow 1.0 - \\
\text{eats(eve,bird)#(ff,0.7)} & \leftarrow 0.7 - \text{animal(bird)#(tt,1.0)} \\
\text{eats(ft(eve),bird)#(ff,0.5)} & \leftarrow 0.8 - \text{eats(eve,bird)#(ff,0.7)}
\end{align*}
\]

- Marked atoms, \( m \in \{tt, ff\} \).
- Annotated atoms, \( (m, w) \) such that \( m \in \{tt, ff\} \) and \( w \in (D \setminus \{\bot\}) \cup \{\?\} \).
- Proving: \( \text{eats(ft(eve),bird)#(ff,0.5)} \) (qualified atom, \( (m, d) \) such that \( m \in \{tt, ff\} \) and \( d \in D \setminus \{\bot\} \)).
  - \( 1.0 = 1.0 \times \min \{\} = 1.0 \times 1.0 \).
  - \( 0.7 = 0.7 \times \min \{1.0\} \).
  - \( 0.5 \leq 0.56 = 0.8 \times \min \{0.7\} \) and \( 0.7 \geq 0.4 \).
**Qualified Horn Logic over** $D$: If there is some instance rule $(A \# v \leftarrow \alpha \leftarrow B_1 \# (v_1, w_1), \ldots, B_k \# (v_k, w_k)) \in [C]$ with $C \in \mathcal{P}$, and some $d_1, \ldots, d_k \in D \setminus \{\bot\}$ such that $d_i \supseteq ? w_i$ for all $1 \leq i \leq k$, then the following inference step is allowed for any $d \sqsubseteq \alpha \circ \bigcap\{d_1, \ldots, d_k\}$:

$$
\frac{B_1 \# (v_1, d_1) \quad \cdots \quad B_k \# (v_k, d_k)}{A \# (v, d)}
$$

- $d \supseteq w \iff_{\text{def}} w = ?$ or $w \neq ?$ and $d \supseteq w$.

Least Herbrand model of $P$:

$$
\{A \# (v, d) | P \vdash_{\QHL} (D) A \# (v, d)\}$$
BQLP Declarative Semantics

- **Qualified Horn Logic over** $\mathcal{D}$: If there is some instance rule \((A \# v \leftarrow \alpha \leftarrow B_1 \# (v_1, w_1), \ldots, B_k \# (v_k, w_k)) \in [C]\) with \(C \in \mathcal{P}\), and some \(d_1, \ldots, d_k \in D \setminus \{\bot\}\) such that \(d_i \supseteq ? w_i\) for all \(1 \leq i \leq k\), then the following inference step is allowed for any \(d \sqsubseteq \alpha \circ \bigcap \{d_1, \ldots, d_k\}\):

\[
\begin{array}{c}
B_1 \# (v_1, d_1) \quad \ldots \quad B_k \# (v_k, d_k) \\
\hline
A \# (v, d)
\end{array}
\]

- \(d \sqsubseteq ? w \iff_{\text{def}} \text{w =? or w} \neq ? \text{ and } d \sqsubseteq w\).

- **Least Herbrand model of** $\mathcal{P}$:

\[
\{A \# (v, d) \mid \mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} A \# (v, d)\}\]
Example 9

cruel(X)#ff <-0.90-
   human(X)#(tt,?), eats(X,Y)#(ff,?), animal(Y)#(tt,?)
animal(cat)#tt <-1.0-
human(eve)#tt <-1.0-
human(mother(X))#tt <-0.90- human(X)#(tt,0.50)
eats(eve,X)#ff <-0.70- animal(X)#(tt,?)
eats(mother(X),Y)#ff <-0.70- eats(X,Y)#(ff,0.40)

\[
\begin{array}{c}
\text{human(eve)#(tt,1.0)} \\
\text{human(mother(eve))#(tt,0.90)} \\
\text{animal(cat)#(tt,1.0)} \\
\text{eats(eve,cat)#(ff,0.70)} \\
\text{eats(mother(eve),cat)#(ff,0.49)} \\
\text{animal(cat)#(tt,1.0)} \\
\text{cruel(mother(eve))#(ff,0.40)}
\end{array}
\]

\[ (*) \ 0.40 \leq 0.90 \ast \min \{0.90, 0.49, 1.0\} = 0.441. \]
Ongoing & Future Work
Ongoing & Future Work

**Ongoing Work**

1. Improvement of the actual prototype supporting threshold constraints and making an extensive use of constraint solvers for qualification constraints managing.
2. Extension of QLP to a first-order Qualified Constraint Functional-Logic Programming Language.

**Future Work**

- Goal Solving by Similarity-based QSLD and its implementation both via a program transformation and similarity-based primitive unification.
- Development of methods and tools for defining complex similarity relations.
- Search for new instances and interesting examples.

To think about:

- Similarity between terms as if they were constants?
- Even more general Qualification Domains without losing usefulness?
Ongoing & Future Work

- **Ongoing Work**
  1. Improvement of the actual prototype supporting threshold constraints and making an extensive use of constraint solvers for qualification constraints managing.
  2. Extension of QLP to a first-order Qualified Constraint Functional-Logic Programming Language.

- **Future Work**
  - Goal Solving by Similarity-based QSLD and its implementation both via a program transformation and similarity-based primitive unification.
  - Develop of methods and tools for defining complex similarity relations.
  - Search for new instances and interesting examples.
Ongoing & Future Work

**Ongoing Work**

1. Improvement of the actual prototype supporting threshold constraints and making an extensive use of constraint solvers for qualification constraints managing.
2. Extension of QLP to a first-order Qualified Constraint Functional-Logic Programming Language.

**Future Work**

- Goal Solving by Similarity-based QSLD and its implementation both via a program transformation and similarity-based primitive unification.
- Develop of methods and tools for defining complex similarity relations.
- Search for new instances and interesting examples.

**To think about:**

- Similarity between terms as if they were constants?
- Even more general Qualification Domains without losing usefulness?
Thank you!