Combining rules and ontologies via parametrized logic programming

Ricardo Gonçalves

NOVA LINCS
Universidade NOVA de Lisboa

ONTOLP 2015
Outline

1. Motivation

2. Parametrized logic programming

3. Combining rules and ontologies

4. Conclusions and future work
Motivation - Parametrized logic programs

- Extension of Answer Set Programming
  - Increase the expressivity of logic programs by allowing complex formulas in the body and head of rules
  - Modular combination of logic programming connectives with other connectives

R. Gonçalves (NOVA LINCS)

Combining rules and ontologies

ONTOLP 2015
Motivation - Parametrized logic programs

- Extension of Answer Set Programming
  - Increase the expressivity of logic programs by allowing complex formulas in the body and head of rules
  - Modular combination of logic programming connectives with other connectives
- Decoupling between the metalevel language and the parametrized logical language
Extension of Answer Set Programming

- Increase the expressivity of logic programs by allowing complex formulas in the body and head of rules
- Modular combination of logic programming connectives with other connectives

Decoupling between the metalevel language and the parametrized logical language

- **metalevel**: usual constructors of rules (←, “,” , not , “or”)
Motivation - Parametrized logic programs

- Extension of Answer Set Programming
  - Increase the expressivity of logic programs by allowing complex formulas in the body and head of rules
  - Modular combination of logic programming connectives with other connectives
- Decoupling between the metalevel language and the parametrized logical language
  - metalevel: usual constructors of rules (←, “,”, not, “or”)
  - parametrized logical level: formulas of the parameter logic
Motivation - Parametrized logic programs

- Extension of Answer Set Programming
  - Increase the expressivity of logic programs by allowing complex formulas in the body and head of rules
  - Modular combination of logic programming connectives with other connectives
- Decoupling between the metalevel language and the parametrized logical language
  - **metalevel:** usual constructors of rules (←, “,”, not, “or”)
  - **parametrized logical level:** formulas of the parameter logic

\[ p \leftarrow p_1, \ldots, p_n, \text{not } q_1, \ldots, \text{not } q_n \]
Motivation - Parametrized logic programs

- Extension of Answer Set Programming
  - Increase the expressivity of logic programs by allowing complex formulas in the body and head of rules
  - Modular combination of logic programming connectives with other connectives
- Decoupling between the metalevel language and the parametrized logical language
  - metalevel: usual constructors of rules (←, “,” , not , “or”)
  - parametrized logical level: formulas of the parameter logic

\[ \varphi \leftarrow \varphi_1, \ldots, \varphi_n, \text{not } \psi_1, \ldots, \text{not } \psi_n \]
Motivation

- Combine non-monotonic rules and ontologies
Motivation

- Combine non-monotonic rules and ontologies
- Use a *description logic* as *parameter logic*
A (monotonic) logic is a pair $\mathcal{L} = \langle L, \vdash_\mathcal{L} \rangle$:

- $L$ is the set of formulas
- $\vdash_\mathcal{L}$ is a Tarskian consequence relation over $L$, i.e., it satisfies:
  - **Reflexivity**: if $\varphi \in T$ then $T \vdash_\mathcal{L} \varphi$;
  - **Cut**: if $T \vdash_\mathcal{L} \varphi$ for all $\varphi \in \Phi$, and $\Phi \vdash_\mathcal{L} \psi$ then $T \vdash_\mathcal{L} \psi$;
  - **Weakening**: if $T \vdash_\mathcal{L} \varphi$ and $T \subseteq \Phi$ then $\Phi \vdash_\mathcal{L} \varphi$. 

Examples

- Classical logic, Intuitionistic logic, Paraconsistent logics, First-order logic,
- Modal logic, Description logics, . . .
A (monotonic) logic is a pair $\mathcal{L} = \langle L, \vdash_\mathcal{L} \rangle$:

- $L$ is the set of formulas
- $\vdash_\mathcal{L}$ is a Tarskian consequence relation over $L$, i.e., it satisfies:
  - **Reflexivity:** if $\varphi \in T$ then $T \vdash_\mathcal{L} \varphi$;
  - **Cut:** if $T \vdash_\mathcal{L} \varphi$ for all $\varphi \in \Phi$, and $\Phi \vdash_\mathcal{L} \psi$ then $T \vdash_\mathcal{L} \psi$;
  - **Weakening:** if $T \vdash_\mathcal{L} \varphi$ and $T \subseteq \Phi$ then $\Phi \vdash_\mathcal{L} \varphi$.

Examples

Classical logic, Intuitionistic logic, Paraconsistent logics, First-order logic, Modal logic, Description logics, ...
A logical theory of $\mathcal{L} = \langle L, \vdash_{\mathcal{L}} \rangle$ is a set $\Phi \subseteq L$ of formulas closed under logical consequence, i.e., if $\Phi \vdash_{\mathcal{L}} \varphi$ then $\varphi \in \Phi$. 
A logical theory of $\mathcal{L} = \langle L, \vdash_{\mathcal{L}} \rangle$ is a set $\Phi \subseteq L$ of formulas closed under logical consequence, i.e., if $\Phi \vdash_{\mathcal{L}} \varphi$ then $\varphi \in \Phi$.

For every monotonic logic $\mathcal{L}$, the pair $\langle Th(\mathcal{L}), \subseteq \rangle$ is a complete lattice with

- smallest element: the set of theorems of $\mathcal{L}$
- greatest element: the set $L$ of all formulas of $\mathcal{L}$;
parameter logic = a fixed monotonic logic $\mathcal{L} = \langle L, \vdash_{\mathcal{L}} \rangle$;

(parametrized) atoms = the formulas of $\mathcal{L}$;
Parameter logic = a fixed monotonic logic $\mathcal{L} = \langle L, \vdash_\mathcal{L} \rangle$; (parametrized) atoms = the formulas of $\mathcal{L}$;

**Definition**

A $\mathcal{L}$-parametrized logic program is a set of rules

$$\varphi \leftarrow \psi_1, \ldots, \psi_n, \text{not } \delta_1, \ldots, \text{not } \delta_m$$

where $\varphi, \psi_1, \ldots, \psi_n, \delta_1, \ldots, \delta_m \in L$. 

R. Gonçalves (NOVA LINCS) Combining rules and ontologies ONTOLP 2015 7 / 16
Parameter logic = a fixed monotonic logic $\mathcal{L} = \langle L, \vdash_\mathcal{L} \rangle$;

(parametrized) atoms = the formulas of $\mathcal{L}$;

**Definition**

A $\mathcal{L}$-parametrized logic program is a set of rules

$$\varphi \leftarrow \psi_1, \ldots, \psi_n, \text{not } \delta_1, \ldots, \text{not } \delta_m$$

where $\varphi, \psi_1, \ldots, \psi_n, \delta_1, \ldots, \delta_m \in L$.

**Definition**

A definite $\mathcal{L}$-parametrized logic program is a set of rules without negations as failure, i.e. of the form $\varphi \leftarrow \psi_1, \ldots, \psi_n$ where $\varphi, \psi_1, \ldots, \psi_n \in L$. 
A (parametrized) interpretation is a logical theory of $\mathcal{L}$. 
Stable model semantics

Definition

A (parametrized) interpretation is a logical theory of $\mathcal{L}$.

Definition

An interpretation $I$ satisfies a rule

$$\varphi \leftarrow \psi_1, \ldots, \psi_n, \text{not } \delta_1, \ldots, \text{not } \delta_m$$

if $\varphi \in I$ whenever $\{\psi_1, \ldots, \psi_n\} \subseteq I$ and $\{\delta_1, \ldots, \delta_m\} \cap I = \emptyset$. 
Stable model semantics

**Definition**

A (parametrized) interpretation is a logical theory of $\mathcal{L}$.

**Definition**

An interpretation $I$ satisfies a rule

$$\varphi \leftarrow \psi_1, \ldots, \psi_n, \text{not } \delta_1, \ldots, \text{not } \delta_m$$

if $\varphi \in I$ whenever $\{\psi_1, \ldots, \psi_n\} \subseteq I$ and $\{\delta_1, \ldots, \delta_m\} \cap I = \emptyset$.

**Definition**

An interpretation is a model of $\mathcal{P}$ if it satisfies every rule of $\mathcal{P}$.
Stable model semantics

**Theorem**

*Every definite $\mathcal{L}$-parametrized logic program has a least model.*
Stable model semantics

Theorem

Every definite $\mathcal{L}$-parametrized logic program has a least model.

Gelfond-Lifschitz like operator

Let $\mathcal{P}$ be a $\mathcal{L}$-parametrized logic program and $I$ an interpretation. The GL-transformation of $\mathcal{P}$ modulo $I$ is the program $\mathcal{P}_I$ obtained by:

- removing from $\mathcal{P}$ all rules containing a literal not $\varphi$ such that $I \models_{\mathcal{L}} \varphi$;
- removing from the remaining rules of $\mathcal{P}$ all default negated literals.

Define $\Gamma_{\mathcal{P}}(I) = J$, where $J$ is the unique least model of $\mathcal{P}_I$. 
Stable model semantics

**Theorem**

*Every definite $\mathcal{L}$-parametrized logic program has a least model.*

**Gelfond-Lifschitz like operator**

Let $\mathcal{P}$ be a $\mathcal{L}$-parametrized logic program and $I$ an interpretation. The GL-transformation of $\mathcal{P}$ modulo $I$ is the program $\mathcal{P}_I$ obtained by:

- removing from $\mathcal{P}$ all rules containing a literal not $\varphi$ such that $I \models_\mathcal{L} \varphi$;
- removing from the remaining rules of $\mathcal{P}$ all default negated literals.

Define $\Gamma_{\mathcal{P}}(I) = J$, where $J$ is the unique least model of $\mathcal{P}_I$.

**Definition**

A parametrized interpretation $I$ is a **stable model** of a $\mathcal{L}$-parametrized logic program $\mathcal{P}$ iff $\Gamma_{\mathcal{P}}(I) = I$. 
Example

Classical propositional logic

\[ r \leftarrow \neg t \]
\[ t \leftarrow \neg r \]
\[ \neg p \leftarrow \]
\[ (p \lor q) \leftarrow r \]
\[ s \leftarrow q \]
Example

Classical propositional logic

\[ \begin{align*}
  r & \leftarrow \text{not } t \\
  t & \leftarrow \text{not } r \\
  P : \quad \neg p & \leftarrow \\
  (p \lor q) & \leftarrow r \\
  s & \leftarrow q \\
  \end{align*} \]

\[ \{\neg p, (p \lor q)\} \vdash_{\text{CPL}} q \]
Example

Classical propositional logic

\[ r \leftarrow \neg t \]
\[ t \leftarrow \neg r \]

\[ \mathcal{P} : \quad \neg p \leftarrow \quad \{\neg p, (p \lor q)\} \vdash_{\text{CPL}} q \]

\[ (p \lor q) \leftarrow r \]
\[ s \leftarrow q \]

Two stable models:
Example

Classical propositional logic

\[
\begin{align*}
    r & \leftarrow \neg t \\
    t & \leftarrow \neg r \\
    \mathcal{P} : & \quad \neg p \leftarrow \\
    & (p \lor q) \leftarrow r \\
    s & \leftarrow q
\end{align*}
\]

Two stable models:

\[
\begin{align*}
    \{r, \neg p, (p \lor q), q, s\} \vdash_{\text{CPL}} & \quad \text{and} \quad \{t, \neg p\} \vdash_{\text{CPL}}
\end{align*}
\]
### Description logic $ALC$

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic concept</td>
<td>$A \in N_C$</td>
</tr>
<tr>
<td></td>
<td>$A^\mathcal{I} \subseteq \Delta^\mathcal{I}$</td>
</tr>
<tr>
<td>atomic role</td>
<td>$R \in N_R$</td>
</tr>
<tr>
<td></td>
<td>$R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$</td>
</tr>
<tr>
<td>individual</td>
<td>$a \in N_I$</td>
</tr>
<tr>
<td></td>
<td>$a^\mathcal{I} \in \Delta^\mathcal{I}$</td>
</tr>
</tbody>
</table>

Interpretation: $\mathcal{I} = (\Delta^\mathcal{I}, \mathcal{I})$
Description logic $\mathcal{ALC}$

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>$\top$</td>
</tr>
<tr>
<td>bottom</td>
<td>$\bot$</td>
</tr>
<tr>
<td>conjunction</td>
<td>$C \sqcap D$</td>
</tr>
<tr>
<td>disjunction</td>
<td>$C \sqcup D$</td>
</tr>
<tr>
<td>complement</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>existential restriction</td>
<td>$\exists R.C$</td>
</tr>
<tr>
<td>universal restriction</td>
<td>$\forall R.C$</td>
</tr>
</tbody>
</table>
Description logic \( \mathcal{ALC} \)

### Axioms

<table>
<thead>
<tr>
<th>Concept Type</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>concept inclusion</td>
<td>( C \sqsubseteq D )</td>
<td>( C^\mathcal{I} \sqsubseteq D^\mathcal{I} )</td>
</tr>
<tr>
<td>concept assertion</td>
<td>( C(a) )</td>
<td>( a^\mathcal{I} \in C^\mathcal{I} )</td>
</tr>
<tr>
<td>role assertion</td>
<td>( R(a, b) )</td>
<td>( (a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I} )</td>
</tr>
</tbody>
</table>

### Consequence relation

\( \mathcal{O} \models \alpha \) if every model of \( \mathcal{O} \) satisfies \( \alpha \).
\[
\text{NotMarried} \equiv \neg \text{Married} \leftarrow \\
\text{NotMarried} \sqsubseteq \text{HighRisk} \leftarrow \\
\exists \text{Spouse.} \top \sqsubseteq \text{Married} \leftarrow \\
\text{NotMarried}(x) \leftarrow p(x), \neg \text{Married}(x) \\
\text{Discount}(x) \leftarrow \text{Spouse}(x, y), p(x), p(y) \\
p(\text{John}) \leftarrow
\]
\[
\text{NotMarried} \equiv \neg \text{Married} \leftarrow \\
not \text{Married} \sqsubseteq \text{HighRisk} \leftarrow \\
\exists \text{Spouse.} \top \sqsubseteq \text{Married} \leftarrow \\
\text{NotMarried}(x) \leftarrow p(x), \text{not Married}(x) \\
\text{Discount}(x) \leftarrow \text{Spouse}(x, y), p(x), p(y) \\
p(\text{John}) \leftarrow
\]
\[ \text{NotMarried} \equiv \neg \text{Married} \leftarrow \]

\[ \text{NotMarried} \equiv \text{HighRisk} \leftarrow \quad \exists \text{Spouse}. \top \equiv \text{Married} \leftarrow \]

\[ \text{NotMarried}(x) \leftarrow p(x), \neg \text{Married}(x) \]

\[ \text{Discount}(x) \leftarrow \text{Spouse}(x, y), p(x), p(y) \]

\[ p(\text{John}) \leftarrow \]
Example

\[\neg \text{Married} \sqsubseteq \text{HighRisk} \leftarrow \text{not exceptionalPeriod} \]

\[\exists \text{Spouse}. \top \sqsubseteq \text{Married} \leftarrow \]

\[\neg \text{Married}(x) \leftarrow p(x), \text{not Married}(x) \]

\[\text{Discount}(x) \leftarrow \text{Spouse}(x, y), p(x), p(y) \]

\[p(\text{John}) \leftarrow \]
\neg Married \sqsubseteq HighRisk \leftarrow not \ exceptional Period

\exists \text{Spouse}. \top \sqsubseteq \text{Married} \leftarrow

\neg \text{Married}(x) \leftarrow p(x), not \text{Married}(x)

\text{Discount}(x) \leftarrow \text{Spouse}(x,y), p(x), p(y)

p(John) \leftarrow
\neg Married \sqsubseteq HighRisk \leftarrow \text{not exceptionalPeriod}

\exists Spouse. \top \sqsubseteq Married \leftarrow

\neg Married(x) \leftarrow p(x), \text{not Married}(x)

\text{Discount}(x) \leftarrow Spouse(x, y), p(x), p(y)

p(\text{John}) \leftarrow

\neg Married(\text{John}) \text{ and } HighRisk(\text{John}) \text{ follow from this program.}
\neg Married \sqsubseteq HighRisk \leftarrow not \ exceptionalPeriod

\exists Spouse. \top \sqsubseteq Married \leftarrow

\neg Married(x) \leftarrow p(x), \neg Married(x)

Discount(x) \leftarrow Spouse(x, y), p(x), p(y)

p(John) \leftarrow

p(Bill) \leftarrow

\exists Spouse. \top (Bill) \leftarrow

exceptionalPeriod \leftarrow
\neg Married \sqsubseteq HighRisk \leftarrow \text{not exceptionalPeriod}

\exists Spouse. \top \sqsubseteq Married \leftarrow

\neg Married(x) \leftarrow p(x), \text{not Married}(x)

Discount(x) \leftarrow Spouse(x, y), p(x), p(y)

p(John) \leftarrow

p(Bill) \leftarrow

\exists Spouse. \top (Bill) \leftarrow

\text{exceptionalPeriod} \leftarrow

\neg Married(John) \text{ follows from } P \text{ but } HighRisk(John) \text{ does not.}
\neg Married \sqsubseteq HighRisk \leftarrow \neg exceptionalPeriod

\exists Spouse. \top \sqsubseteq Married \leftarrow

\neg Married(x) \leftarrow p(x), \neg Married(x)

Discount(x) \leftarrow Spouse(x, y), p(x), p(y)

p(John) \leftarrow

p(Bill) \leftarrow

\exists Spouse. \top (Bill) \leftarrow

does not.
Decidability

For a finite parametrized logic program over a decidable parameter logic, entailment over stable model semantics is decidable.
Decidability

For a finite parametrized logic program over a decidable parameter logic, entailment over stable model semantics is decidable.

Implementation

Modular implementation combining a reasoner for the parameter logic and an answer set solver.
Conclusions and future work

Conclusions

- Parametrized logic programming as a framework for combining rules and ontologies
- Expressive: allows complex DL formulas in the head and body of rules
- Decidable and can be implemented in a modular way

Future work

- Explore the well-founded semantics of parametrized logic programs
- Extensive comparison with related work
Conclusions and future work

Conclusions

- Parametrized logic programming as a framework for combining rules and ontologies
- Expressive: allows complex DL formulas in the head and body of rules
- Decidable and can be implemented in a modular way

Future work

- Explore the well-founded semantics of parametrized logic programs
- Extensive comparison with related work