

Combining rules and ontologies via parametrized logic programming

Ricardo Gonçalves

NOVA LINCS
Universidade NOVA de Lisboa

ONTOLP 2015

- 1 Motivation
- 2 Parametrized logic programming
- 3 Combining rules and ontologies
- 4 Conclusions and future work

- 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

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**

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 (\leftarrow , “,” , *not* , “or”)

- 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 (\leftarrow , “,” , *not* , “or”)
 - **parametrized logical level**: formulas of the parameter logic

- 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 (\leftarrow , “,” , *not* , “or”)
 - **parametrized logical level**: formulas of the parameter logic

$$p \leftarrow p_1, \dots, p_n, \textit{not } q_1, \dots, \textit{not } q_n$$

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 (\leftarrow , “,” , *not* , “or”)
 - **parametrized logical level**: formulas of the parameter logic

$$\varphi \leftarrow \varphi_1, \dots, \varphi_n, \textit{not } \psi_1, \dots, \textit{not } \psi_n$$

- Combine **non-monotonic rules** and **ontologies**

- Combine non-monotonic rules and ontologies
- Use a **description logic** as **parameter logic**

Definition

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 $\Phi \vdash_{\mathcal{L}} \varphi$.

Definition

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, ...

Definition

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 $\varphi \in \Phi$.

Definition

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 $\varphi \in \Phi$.

Property

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

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

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

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

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

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

Definition

A **definite** \mathcal{L} -parametrized logic program is a set of rules without negations as failure, i.e. of the form $\varphi \leftarrow \psi_1, \dots, \psi_n$ where $\varphi, \psi_1, \dots, \psi_n \in L$.

Definition

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

Definition

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

Definition

An interpretation I **satisfies** a rule

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

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

Definition

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

Definition

An interpretation I satisfies a rule

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

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

Definition

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

Theorem

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

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 $\frac{\mathcal{P}}{I}$ obtained by:

- removing from \mathcal{P} all rules containing a literal *not* φ such that $I \vdash_{\mathcal{L}} \varphi$;
- 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 $\frac{\mathcal{P}}{I}$.

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 $\frac{\mathcal{P}}{I}$ obtained by:

- removing from \mathcal{P} all rules containing a literal *not* φ such that $I \vdash_{\mathcal{L}} \varphi$;
- 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 $\frac{\mathcal{P}}{I}$.

Definition

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

Classical propositional logic

$$r \leftarrow \text{not } t$$

$$t \leftarrow \text{not } r$$

$$\mathcal{P} : \quad \neg p \leftarrow$$

$$(p \vee q) \leftarrow r$$

$$s \leftarrow q$$

Classical propositional logic

$$r \leftarrow \text{not } t$$

$$t \leftarrow \text{not } r$$

$$\mathcal{P} : \quad \neg p \leftarrow \quad \{\neg p, (p \vee q)\} \vdash_{\text{CPL}} q$$

$$(p \vee q) \leftarrow r$$

$$s \leftarrow q$$

Classical propositional logic

$$r \leftarrow \text{not } t$$

$$t \leftarrow \text{not } r$$

$$\mathcal{P} : \quad \begin{array}{l} \neg p \leftarrow \\ (p \vee q) \leftarrow r \\ s \leftarrow q \end{array} \quad \{\neg p, (p \vee q)\} \vdash_{\text{CPL}} q$$

Two stable models:

Classical propositional logic

$$r \leftarrow \text{not } t$$

$$t \leftarrow \text{not } r$$

$$\mathcal{P} : \quad \begin{array}{l} \neg p \leftarrow \\ (p \vee q) \leftarrow r \\ s \leftarrow q \end{array} \quad \{\neg p, (p \vee q)\} \vdash_{\text{CPL}} q$$

Two stable models:

$$\{r, \neg p, (p \vee q), q, s\}^{\vdash_{\text{CPL}}} \quad \text{and} \quad \{t, \neg p\}^{\vdash_{\text{CPL}}}$$

	Syntax	Semantics
atomic concept	$A \in N_C$	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
atomic role	$R \in N_R$	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
individual	$a \in N_I$	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

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

	Syntax	Semantics
top	\top	$\Delta^{\mathcal{I}}$
bottom	\perp	\emptyset
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$

Axioms

	Syntax	Semantics
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$R(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Consequence relation

$\mathcal{O} \models \alpha$ if every model of \mathcal{O} satisfies α .

Example [Motik and Rosati 2007]

$NotMarried \equiv \neg Married \leftarrow$

$NotMarried \sqsubseteq HighRisk \leftarrow$

$\exists Spouse.\top \sqsubseteq Married \leftarrow$

$NotMarried(x) \leftarrow p(x), not\ Married(x)$

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

$p(John) \leftarrow$

Example [Motik and Rosati 2007]

$NotMarried \equiv \neg Married \leftarrow$

$NotMarried \sqsubseteq HighRisk \leftarrow$

$\exists Spouse.\top \sqsubseteq Married \leftarrow$

$NotMarried(x) \leftarrow p(x), not\ Married(x)$

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

$p(John) \leftarrow$

$NotMarried \equiv \neg Married \leftarrow$

$NotMarried \sqsubseteq HighRisk \leftarrow$

$\exists Spouse.\top \sqsubseteq Married \leftarrow$

$NotMarried(x) \leftarrow p(x), not\ Married(x)$

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

$p(John) \leftarrow$

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

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

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

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

$p(\text{John}) \leftarrow$

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

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

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

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

$p(\text{John}) \leftarrow$

Example

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

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

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

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

$p(\text{John}) \leftarrow$

$\neg \text{Married}(\text{John})$ and $\text{HighRisk}(\text{John})$ follow from this program.

Example

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

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

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

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

$p(\text{John}) \leftarrow$

$p(\text{Bill}) \leftarrow$

$\exists \text{Spouse}.\top(\text{Bill}) \leftarrow$

$\text{exceptionalPeriod} \leftarrow$

Example

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

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

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

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

$p(\text{John}) \leftarrow$

$p(\text{Bill}) \leftarrow$

$\exists \text{Spouse}.\top(\text{Bill}) \leftarrow$

$\text{exceptionalPeriod} \leftarrow$

$\neg \text{Married}(\text{John})$ follows from P **but** $\text{HighRisk}(\text{John})$ **does not.**

Example

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

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

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

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

$p(\text{John}) \leftarrow$

$p(\text{Bill}) \leftarrow$

$\exists \text{Spouse}.\top(\text{Bill}) \leftarrow$

$\text{exceptionalPeriod} \leftarrow$

Married(Bill) follows from *P* **but** *Discount(Bill)* **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

- 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

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