Combining rules and ontologies via parametrized logic programming

Ricardo Gonçalves

NOVA LINCS
Universidade NOVA de Lisboa

ONTOLP 2015

Outline

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

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")

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

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

$$p \leftarrow p_1, \ldots, p_n, not\ q_1, \ldots, not\ q_n$$

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

$$\varphi \leftarrow \varphi_1, \ldots, \varphi_n, not\ \psi_1, \ldots, not\ \psi_n$$

Motivation

- Combine non-monotonic rules and ontologies

Motivation

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

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

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

Examples

Classical logic, Intuitionistic logic, Paraconsistent logics, First-order logic, Modal logic, Description logics, ...

Logical theory

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 $\varphi \in \Phi$.

Logical theory

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 $\mathcal{L}$;

Language

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

Language

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

$$\varphi \leftarrow \psi_1, \ldots, \psi_n, not\ \delta_1, \ldots, not\ \delta_m$$

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

Language

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

$$\varphi \leftarrow \psi_1, \ldots, \psi_n, not\ \delta_1, \ldots, not\ \delta_m$$

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 $\varphi, \psi_1, \ldots, \psi_n \in L$.

Stable model semantics

Definition

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

$$\varphi \leftarrow \psi_1, \ldots, \psi_n, not\ \delta_1, \ldots, not\ \delta_m$$

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

$$\varphi \leftarrow \psi_1, \ldots, \psi_n, not\ \delta_1, \ldots, not\ \delta_m$$

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 P modulo I is the program $\frac{\mathcal{P}}{I}$ obtained by:

- removing from $\mathcal{P}$ all rules containing a literal $not\ \varphi$ 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 $\frac{\mathcal{P}}{I}$.

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 P modulo I is the program $\frac{\mathcal{P}}{I}$ obtained by:

- removing from $\mathcal{P}$ all rules containing a literal $not\ \varphi$ 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 $\Gamma_{\mathcal{P}}(I) = I$.

Example

Classical propositional logic

$$\mathcal{P}: \quad \begin{aligned} r &\leftarrow not\ t \\ t &\leftarrow not\ r \\ \neg p &\leftarrow \\ (p \vee q) &\leftarrow r \\ s &\leftarrow q \end{aligned}$$

Example

Classical propositional logic

$$\mathcal{P}: \quad \begin{array}{rl} r & \leftarrow not\ t \\ t & \leftarrow not\ r \\ \neg p & \leftarrow \\ (p \vee q) & \leftarrow r \\ s & \leftarrow q \end{array} \qquad \{\neg p,\ (p \vee q)\} \vdash_{\mathsf{CPL}} q$$

Example

Classical propositional logic

$$\mathcal{P}: \quad \begin{aligned} r &\leftarrow not\ t \\ t &\leftarrow not\ r \\ \neg p &\leftarrow \\ (p \vee q) &\leftarrow r \\ s &\leftarrow q \end{aligned} \qquad \{\neg p,\ (p \vee q)\} \vdash_{\mathsf{CPL}} q$$

Two stable models:

Example

Classical propositional logic

$$\mathcal{P}: \quad \begin{array}{rl} r & \leftarrow not\ t \\ t & \leftarrow not\ r \\ \neg p & \leftarrow \\ (p \vee q) & \leftarrow r \\ s & \leftarrow q \end{array} \qquad \{\neg p,\ (p \vee q)\} \vdash_{\mathsf{CPL}} q$$

Two stable models:

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

Description logic $\mathcal{ALC}$

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

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

Description logic $\mathcal{ALC}$

	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
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	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$

Description logic $\mathcal{ALC}$

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]

$$NotMarried \equiv \neg Married \leftarrow$$

$$NotMarried \sqsubseteq HighRisk \leftarrow$$

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

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

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

$$p(John) \leftarrow$$

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

$$\neg Married \sqsubseteq HighRisk \leftarrow not\ exceptionalPeriod$$

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

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

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

$$p(John) \leftarrow$$

Example

$$\neg Married \sqsubseteq HighRisk \leftarrow not\ exceptionalPeriod$$

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

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

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

$$p(John) \leftarrow$$

Example

$$\neg Married \sqsubseteq HighRisk \leftarrow not\ exceptionalPeriod$$

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

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

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

$$p(John) \leftarrow$$

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

Example

$$\neg Married \sqsubseteq HighRisk \leftarrow not\ exceptionalPeriod$$

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

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

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

$$p(John) \leftarrow$$

$$p(Bill) \leftarrow$$

$$\exists Spouse.\top(Bill) \leftarrow$$

$$exceptionalPeriod \leftarrow$$

Example

$$\neg Married \sqsubseteq HighRisk \leftarrow not\ exceptionalPeriod$$

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

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

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

$$p(John) \leftarrow$$

$$p(Bill) \leftarrow$$

$$\exists Spouse.\top(Bill) \leftarrow$$

$$exceptionalPeriod \leftarrow$$

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

Example

$$\neg Married \sqsubseteq HighRisk \leftarrow not\ exceptionalPeriod$$

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

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

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

$$p(John) \leftarrow$$

$$p(Bill) \leftarrow$$

$$\exists Spouse.\top(Bill) \leftarrow$$

$$exceptionalPeriod \leftarrow$$

$Married(Bill)$ follows from P **but** $Discount(Bill)$ **does not**.

On decidability and implementation

Decidability

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

On decidability and implementation

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

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