Combining rules and ontologies via parametrized logic programs

Ricardo Gonçalves
NOVA LINCS
Faculdade de Ciências e Tecnologia
Universidade NOVA de Lisboa

Abstract

Parametrized logic programs are very expressive logic programs that generalize normal logic programs under the stable model semantics, by allowing complex formulas of a parameter logic to appear in the body and head of rules. In this paper we explore the use of description logics as parameter logics, and show the expressivity of this framework for combining rules and ontologies.

Introduction

Parametrized logic programming (Gonçalves and Alferes 2010) was introduced as an extension of answer set programming (Gelfond and Lifschitz 1988) with the motivation of providing a meaning to theories combining both logic programming connectives with other logical connectives, and allowing complex formulas using these connectives to appear in the head and body of a rule. The main idea is to fix a monotonic logic $\mathcal{L}$, called the parameter logic, and build up logic programs using formulas of $\mathcal{L}$ instead of just atoms. The obtained parametrized logic programs have, therefore, the same structure of normal logic programs, being the only difference the fact that atomic symbols are replaced by formulas of $\mathcal{L}$.

When applying this framework, the choice of the parameter logic depends on the domain of the problem to be modeled. As examples, (Gonçalves and Alferes 2010) shows how to obtain the answer-set semantics of logic programs with explicit negation, a paraconsistent version of it, and also the semantics of MKNF hybrid knowledge bases (Motik and Rosati 2010), using an appropriate choice of the parameter logic. In (Gonçalves and Alferes 2012) deontic logic programs are introduced using standard deontic logic (von Wright 1951) as the parameter logic. Moreover, in (Gonçalves and Alferes 2013) the decidability and implementation of parametrized logic was discussed.

Parametrized logic programming can thus be seen as a framework which allows to add non-monotonic rule based reasoning on top of an existing (monotonic) language. This view is quite interesting, in particular in those cases where we already have a monotonic logic to model a problem, but we are still lacking some conditional or non-monotonic reasoning. In these situations, parametrized logic programming offers a modular framework for adding such conditional and non-monotonic reasoning, without having to give up of the monotonic logic at hand.

In recent years, there has been a considerable amount of effort devoted to combining Description Logics (DLs) with logic programming non-monotonic rules – see, e.g., related work in (Eiter et al. 2008; Motik and Rosati 2010).

In this paper we explore precisely the use of description logics as parameter logics, and show the expressivity of the resulting framework for combining rules and ontologies.

Parametrized logic programs

Parametrized logic programs are very expressive logic programs that generalize normal logic programs under the stable model semantics, by allowing complex formulas of a parameter logic to appear in the body and head of rules. In this section we introduce the syntax and semantics of normal parametrized logic programs (Gonçalves and Alferes 2010).

Language

The syntax of a normal parametrized logic program has the same structure of that of a normal logic program. The only difference is that the atomic symbols of a normal parametrized logic program are replaced by formulas of a parameter logic, which is restricted to be a monotonic logic. Let us start by introducing the necessary concepts related with the notion of (monotonic) logic.

Definition 1 *A (monotonic) logic is a pair $\mathcal{L} = \langle L, \vdash_{\mathcal{L}} \rangle$ where L is a set of formulas and $\vdash_{\mathcal{L}}$ is a Tarskian consequence relation (Wójcicki 1988) over L, i.e., satisfying the following conditions, for every $T \cup \Phi \cup \{\varphi\} \subseteq L$,*

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

When clear from the context we write $\vdash$ instead of $\vdash_{\mathcal{L}}$. Let $Th(\mathcal{L})$ be the set of *logical theories of* $\mathcal{L}$, i.e. the set of subsets of L closed under the relation $\vdash_{\mathcal{L}}$. One fundamental characteristic of the above definition of monotonic logic is that it has as a consequence that, for every (monotonic) logic

$\mathcal{L}$, the tuple $\langle Th(\mathcal{L}), \subseteq \rangle$ is a complete lattice with smallest element the set $Theo = \emptyset^{\vdash}$ of theorems of $\mathcal{L}$ and the greatest element the set L of all formulas of $\mathcal{L}$. Given a subset A of L we denote by $A^{\vdash_{\mathcal{L}}}$ the smallest logical theory of $\mathcal{L}$ that contains A. $A^{\vdash_{\mathcal{L}}}$ is also called the logical theory generated by A in $\mathcal{L}$.

In what follows we consider fixed a (monotonic) logic $\mathcal{L} = \langle L, \vdash_{\mathcal{L}} \rangle$ and call it the *parameter logic*. The formulas of $\mathcal{L}$ are dubbed *(parametrized) atoms* and a *(parametrized) literal* is either a parametrized atom φ or its negation $not\ \varphi$, where as usual not denotes negation as failure. We dub *default literal* those of the form $not\ \varphi$.

Definition 2 *A* normal $\mathcal{L}$ parametrized logic program *is a set of rules*

$$\varphi \leftarrow \psi_1, \ldots, \psi_n, not\ \delta_1, \ldots, not\ \delta_m \quad (1)$$

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

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

As usual, the symbol $\leftarrow$ represents rule implication, the symbol "," represents conjunction and the symbol not represents default negation. A rule as (1) has the usual reading that φ should hold whenever $\psi_1, \ldots, \psi_n$ hold and $\delta_1, \ldots, \delta_m$ are not known to hold. If $n = 0$ and $m = 0$ then we just write $\varphi \leftarrow$.

Given a rule r of the form (1), we define $head(r) = \varphi$, $body^+(r) = \{\psi_1, \ldots, \psi_n\}$, $body^-(r) = \{\delta_1, \ldots, \delta_m\}$ and $body(r) = body^+(r) \cup body^-(r)$. Given a parametrized logic program $\mathcal{P}$ we define $form(\mathcal{P})$ to be the set of all formulas of the parameter language L appearing in $\mathcal{P}$, i.e., $form(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} (\{head(r)\} \cup body(r))$. We also define the set $head(\mathcal{P}) = \{head(r) : r \in \mathcal{P}\}$.

Semantics

Given this general language of parametrized logic programs, we define its stable model semantics, as generalization of the stable model semantics (Gelfond and Lifschitz 1988) of normal logic programs.

In the traditional approach an interpretation is just a set of atoms. In a parametrized logic program, since we substitute atoms by formulas of a parameter logic, the first idea is to take sets of formulas of the parameter logic as interpretations. The problem is that, contrary to the case of atoms, the parametrized atoms are not independent of each other. This interdependence is governed by the consequence relation of the parameter logic. For example, if we take classical propositional logic (CPL) as the parameter logic, we have that if the parametrized atom $p \wedge q$ is true then so are the parametrized atoms p and q. If we take, for example, standard deontic logic SDL (von Wright 1951) as parameter, we have that, since $\mathbf{O}(p \vee q), \mathbf{O}(\neg p) \vdash_{SDL} \mathbf{O}(q)$, any SDL logical theory containing both $\mathbf{O}(p \vee q)$ and $\mathbf{O}(\neg p)$ also contains $\mathbf{O}(q)$.

To account for this interdependence, we use logical theories (sets of formulas closed under the consequence of the logic) as the generalization of interpretations, thus capturing the above mentioned interdependence.

Definition 3 *A (parametrized) interpretation is a logical theory of* $\mathcal{L}$.

Definition 4 *An interpretation* T *satisfies a rule*

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

if $\varphi \in T$ *whenever* $\psi_i \in T$ *for every* $i \in \{1, \ldots, n\}$ *and* $\delta_j \notin T$ *for every* $j \in \{1, \ldots, m\}$.

An interpretation is a model of logic program $\mathcal{P}$ if it satisfies every rule of $\mathcal{P}$. We denote by $Mod_{\mathcal{L}}(P)$ the set of models of $\mathcal{P}$.

The ordering over interpretations is the usual one: If T_1 and T_2 are two interpretations then we say that $T_1 \leq T_2$ if $T_1 \subseteq T_2$. Moreover, given such ordering, minimal and least interpretations may be defined in the usual way.

As in the case of non parametrized programs, we start by assigning semantics to definite parametrized programs. Recall that the stable model of a definite logic program is its least model. In order to generalize this definition to the parametrized case we need to establish that the least parametrized model exists for every definite $\mathcal{L}$ parametrized logic program.

Theorem 1 *Every definite* $\mathcal{L}$ *parametrized logic program has a least model.*

We denote by $S_P^{\mathcal{L}}$ the least model of a definite program P.

It is important to note that Theorem 1 holds for every choice of the parameter logic $\mathcal{L}$.

The stable model semantics of a normal $\mathcal{L}$ parametrized logic program is defined using a Gelfond-Lifschitz like operator.

Definition 5 *Let* $\mathcal{P}$ *be a normal* $\mathcal{L}$ *parametrized logic program and* T *an interpretation. The GL-transformation of* $\mathcal{P}$ *modulo* T *is the program* $\frac{P}{T}$ *obtained from P by performing the following operations:*

- *remove from* $\mathcal{P}$ *all rules which contain a literal* $not\ \varphi$ *such that* $T \vdash_{\mathcal{L}} \varphi$*;*
- *remove from the remaining rules all default literals.*

Since $\frac{P}{T}$ *is a definite* $\mathcal{L}$ *parametrized program, it has an unique least model* J*. We define* $\Gamma(T) = J$.

Stable models of a parametrized logic program are then defined as fixed points of this Γ operator.

Definition 6 *An interpretation* T *of an* $\mathcal{L}$ *parametrized logic program* $\mathcal{P}$ *is a stable model of* $\mathcal{P}$ *iff* $\Gamma(T) = T$*. A formula* φ *is true under the stable model semantics, denoted by* $\mathcal{P} \models_{SMS} \varphi$ *iff it belongs to all stable models of* $\mathcal{P}$.

An important feature of parametrized logic programming is that its stable model semantics is independent of the semantics of the parameter logic, since the central concept is the consequence relation of the parameter logic.

Let us now show an example of how parametrized logic programs can be used to combine a monotonic formalism with a non-monotonic one. We choose three different logics over the same propositional language.

Example 1 (Propositional logic programs) *Let us now consider a full propositional language L built over a set $\mathcal{P}$ of propositional symbols using the usual connectives $(\neg, \vee, \wedge, \Rightarrow)$. Many consequence relations can be defined over this language. We present three interesting examples: classical logic, Belnap's paraconsistent logic and intuitionistic logic. Consider the following programs:*

$$P_1 \begin{cases} p \leftarrow \neg q \\ p \leftarrow q \end{cases} \qquad P_2 \ \{ \ p \leftarrow \neg q \vee q$$

$$P_3 \begin{cases} q \leftarrow \\ (q \vee s) \Rightarrow p \leftarrow \\ r \leftarrow p \end{cases} \qquad P_4 \begin{cases} r \leftarrow \\ \neg p \leftarrow \\ (p \vee q) \leftarrow r \\ s \leftarrow q \end{cases}$$

$$P_5 \ \{ \ p \leftarrow not\ q, not\ \neg q \qquad P_6 \ \{ \ p \leftarrow not\ (q \vee \neg q)$$

Let $\mathcal{L} = \langle L, \vdash_{CPL} \rangle$ be Classical Propositional Logic (CPL) over the language L. Let us study the semantics of P_1. Note that every logical theory of CPL that does not contain neither p nor $\neg p$ satisfies P_1. In particular, the set $Taut$ of tautologies of CPL is a model of P_1. So, $S^{CPL}_{P_1} = Taut$. This means that $p, \neg p, q, \neg q \notin S^{CPL}_{P_1}$. We also have that $S^{CPL}_{P_2} = \{p\}^{\vdash}$. So, in the case of P_2 we have that $p \in S^{CPL}_{P_2}$. Also, we have that $r \in S^{CPL}_{P_3}$ and $s \in S^{CPL}_{P_4}$.

In the case of P_5 its stable models are the theories of CPL that contain p and do not contain q and $\neg q$. Therefore, we can conclude that $p \in S^{CPL}_{P_5}$. In the case of P_6, since $(p \vee \neg p) \in T$ for every logical theory T of CPL we can conclude that the only stable model of P_6 is the set $Theo$ of theorems of CPL. Therefore $p \notin S^{CPL}_{P_6}$.

Consider now $\mathcal{L} = \langle L, \vdash_4 \rangle$ the 4-valued Belnap paraconsistent logic $Four$. Consider the program P_4. Contrarily to the case of CPL, in $Four$ it is not the case that $\neg p, (p \vee q) \vdash_4 q$. Therefore we have that $q, s \notin S^{Four}_{P_4}$.

Let now $\mathcal{L} = \langle L, \vdash_{IPL} \rangle$ be the propositional intuitionistic logic IPL. It is well-known that $q \vee \neg q$ is not a theorem of IPL. Therefore, considering program P_2 we have $S^{IPL}_{P_2} = \emptyset^{\vdash_{IPL}}$. So, contrarily to the case of CPL, we have that $p \notin S^{IPL}_{P_2}$. Using the same idea for program P_6 we can conclude, contrarily to the case of CPL, that $p \in S^{IPL}_{P_6}$.

Combining rules and ontologies

In this section we discuss the use of description logics as parameter logic in the framework of parametrized logic programming. We will then illustrate the expressivity of the resulting framework to combine non-monotonic rules and ontologies.

In what follows, and for simplicity, we use description logic $\mathcal{ALC}$ (Schmidt-Schaubß and Smolka 1991). We start by briefly recalling the syntax and semantics of $\mathcal{ALC}$. For a more general and thorough introduction to DLs we refer to (Baader et al. 2010). The language of $\mathcal{ALC}$ is defined over countably infinite sets of *concept names* $\mathsf{N_C}$, *role names* $\mathsf{N_R}$, and *individual names* $\mathsf{N_I}$ as shown in the upper part of Table 1. Building on these, *complex concepts* are introduced in the middle part of Table 1, which, together with atomic concepts, form the set of *concepts*. We conveniently denote individuals by a and b, (atomic) roles by R and S, atomic concepts by A and B, and concepts by C and D. All expressions in the lower part of Table 1 are *axioms*. A *concept equivalence* $C \equiv D$ is an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. Concept and role assertions are *ABox axioms* and all other axioms *TBox axioms*, and an *ontology* is a finite set of axioms.

Table 1: Syntax and semantics of $\mathcal{ALC}$.

	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	$a \in \mathsf{N_I}$	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
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	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
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	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

The semantics of $\mathcal{ALC}$ is defined in terms of *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \ ^{\mathcal{I}})$, which consist of a non-empty domain $\Delta^{\mathcal{I}}$ and an *interpretation function* $^{\mathcal{I}}$. The latter is defined for (arbitrary) concepts, roles, and individuals as in Table 1. Moreover, an interpretation $\mathcal{I}$ *satisfies* an axiom α, written $\mathcal{I} \models \alpha$, if the corresponding condition in Table 1 holds. If $\mathcal{I}$ satisfies all axioms in an ontology $\mathcal{O}$, then $\mathcal{I}$ is a *model* of $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$. If $\mathcal{O}$ has at least one model, then it is called *consistent*, otherwise *inconsistent*. Also, $\mathcal{O}$ *entails* axiom α, written $\mathcal{O} \models \alpha$, if every model of $\mathcal{O}$ satisfies α.

Given the consequence relation of $\mathcal{ALC}$ we can now illustrate how $\mathcal{ALC}$ can be used as parameter logic.

Example 2 *The following program (P_1) is an adaptation of an example taken from (Motik and Rosati 2007), which uses MKNF knowledge bases to combine rules and ontologies. The scenario is about determining the car insurance pre-*

mium based on various information about the driver.

$$NotMarried \equiv \neg Married \leftarrow$$
$$NotMarried \sqsubseteq HighRisk \leftarrow$$
$$\exists Spouse.\top \sqsubseteq Married \leftarrow$$
$$NotMarried(x) \leftarrow p(x), not\ Married(x)$$
$$Discount(x) \leftarrow Spouse(x,y), p(x), p(y)$$
$$p(Jonh) \leftarrow$$

Note that in parametrized logic programming the combination of an ontology with a rule system can be done in a natural way, simply by adding the ontology elements as facts of the rule system. As usual in logic programming, variables in rules stand for all their possible instantiations by individuals appearing in the program.

Program P_1 can be rewritten in order to remove its first rule, which is nothing but an artificial tool to overcome the impossibility of having complex DL formulas in the head of MKNF rules (in this case, having the classical negation of an atom in a head). Moreover, we may also add bodies to the facts coming from the ontology. E.g. we can add a non-monotonic condition to the second statement of P_1 above, to state that non married are only considered high-risk in non exceptional periods, obtaining P_2:

$$\neg Married \sqsubseteq HighRisk \leftarrow not\ exceptionalPeriod$$
$$\exists Spouse.\top \sqsubseteq Married \leftarrow$$
$$\neg Married(x) \leftarrow p(x), not\ Married(x)$$
$$Discount(x) \leftarrow Spouse(x,y), p(x), p(y)$$
$$p(Jonh) \leftarrow$$

Let us now study the stable model semantics of this program. We should again stress that such stable model semantics does not depend on the semantics of $\mathcal{ALC}$, but only on its consequence relation. If I is a 2-valued interpretation such that $I(Married(Jonh)) = 1$ then $\Gamma(I)$ is the least model of the following program $\frac{P_2}{I}$:

$$\neg Married \sqsubseteq HighRisk \leftarrow$$
$$\exists Spouse.\top \sqsubseteq Married \leftarrow$$
$$Discount(x) \leftarrow Spouse(x,y), p(x), p(y)$$
$$p(Jonh) \leftarrow$$

It is clear that the smallest model of $\frac{P_2}{I}$ does not contain $Married(Jonh)$, and so, such interpretation I cannot be a stable model. Therefore, every stable model must satisfy $\neg Married(Jonh)$ and consequently $HighRisk(Jonh)$.

Consider now program P_3 obtained by adding to P_2 the following facts: $p(Bill) \leftarrow$, $\exists Spouse.\top(Bill) \leftarrow$, and $exceptionalPeriod \leftarrow$. Note that, although every stable model now contains $\neg Married(John)$, we no longer conclude $HighRisk(Jonh)$ since we have $exceptionalPeriod$. Every stable model of P_3 contains $Married(Bill)$. So, the Stable Model Semantics of P_3 does not entail $\neg Married(Bill)$ nor $HighRisk(Bill)$.

Consider now program P_4 obtained by adding to P_2 the facts: $Spouse(Bob, Ann) \leftarrow$, $p(Bob) \leftarrow$, and $p(Ann) \leftarrow$. Every stable model of P_4 contains $Discount(Bob)$, and so it entails $Discount(Bob)$.

Conclusions

In this paper we have discussed the use of the framework of parametrized logic programming for combining non-monotonic rules and ontologies. This approach is quite expressive since it allows complex DL axioms to appear both in the body and in the head of non-monotonic rules.

In (Gonçalves and Alferes 2010) the authors show how parametrized logic programming can capture the semantics of MKNF hybrid knowledge bases (Motik and Rosati 2010) by an appropriate choice of the parameter logic. As future work we aim to study the relation between parametrized logic programs and other frameworks for combining rules and ontologies, e.g., the DL-programs of (Eiter et al. 2008).

Aknowledgments

Ricardo Gonçalves was supported by FCT under project ERRO (PTDC/EIA-CCO/121823/2010).

References

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F. 2010. The description logic handbook, theory, implementation, and applications (2nd edition). Cambridge University Press.

Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2008. Combining answer set programming with description logics for the semantic web. *Artif. Intell.* 172(12-13):1495–1539.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. 1070–1080. MIT Press.

Gonçalves, R., and Alferes, J. J. 2013. Decidability and implementation of parametrized logic programs. In Cabalar, P., and Son, T. C., eds., *LPNMR*, volume 8148 of *LNCS*, 361–373. Springer.

Gonçalves, R., and Alferes, J. J. 2010. Parametrized logic programming. In Janhunen, T., and Niemelä, I., eds., *JELIA*, volume 6341 of *LNCS*, 182–194. Springer.

Gonçalves, R., and Alferes, J. J. 2012. An embedding of input-output logic in deontic logic programs. In Ågotnes, T.; Broersen, J.; and Elgesem, D., eds., *DEON*, volume 7393 of *LNCS*, 61–75. Springer.

Motik, B., and Rosati, R. 2007. A faithful integration of description logics with logic programming. In *IJCAI*, 477–482.

Motik, B., and Rosati, R. 2010. Reconciling description logics and rules. *J. ACM* 57(5).

Schmidt-Schaubß M., and Smolka, G. 1991. Attributive concept descriptions with complements. *Artif. Intell.* 48(1):1–26.

von Wright, G. H. 1951. Deontic logic. *Mind* 60:1–15.

Wójcicki, R. 1988. *Theory of Logical Calculi*. Synthese Library. Kluwer Academic Publishers.