Parametrized logic programming

Ricardo Gonçalves* and José Júlio Alferes

CENTRIA, Dep. Informática, FCT/Universidade Nova de Lisboa, Portugal

Abstract. Traditionally, a logic program is built up to reason about atomic first-order formulas. The key idea of parametrized logic programming is that, instead of atomic first-order formulas, a parametrized logic program reasons about formulas of a given parameter logic. Of course, the main challenge is to define the semantics of such general programs. In this work we introduce the novel definitions along with some motivating examples. This approach allows us to prove general results that can be instantiated for every particular choice of the parameter logic. Important general results we can prove include the existence of semantics and the alternating fix-point theorem of well-founded semantics. To reenforce the soundness of our general framework we show that some known approaches in the literature of logic programming, such as paraconsistent answer-sets and the MKNF semantics for hybrid knowledge bases, are obtained as particular choices of the parameter logic.

1 Introduction

Usually, we write a logic program to reason about atomic formulas. The truth value of each of these atoms does not influence the truth values of the others. Thus, in this sense, atomic first-order formulas are independent of each other. One way to increase the expressivity of a logic program is to allow dependence between the atoms. Pursuing this path, some approaches in the literature allow complex formulas to appear in the body and head of rules [4, 15].

Our approach, parametrized logic programming, also aims this increased expressivity, but has its roots in the area of combination of logics [2]. Combination mechanisms are operations that take logics as arguments and produce new logics as a result. Combined logics are not only important from a theoretical point of view but, what is more, have also a deep practical significance, namely in areas like knowledge representation in artificial intelligence. In fact, the use of formal logic as a tool for knowledge representation frequently requires the integration of several logic systems into a homogeneous environment. Fibring of logics [5] is one of the most general and flexible mechanism for combining logics and it has parametrization of logics [1] as an important special case. Roughly speaking, parametrization of logics consists of replacing the atomic part of a given logic $\mathcal{L}$ by a logic $\mathcal{L}'$, which is called the parameter logic.

* The first author was supported by FCT under the postdoctoral grant SFRH/BPD/47245/2008.

In the case of logic programming, our idea is to fix a parameter logic $\mathcal{L}$ and build up logic programs by replacing atomic sentences with formulas of $\mathcal{L}$. These more expressive atoms (the formulas of $\mathcal{L}$) are no longer independent and their interdependence is governed by $\mathcal{L}$ through its consequence relation.

It is important to stress that our approach differs from existing ones, such as [15, 3], that aim to find the "logic" of logic programming, i.e., find a suitable logic that generalizes the well-known role of classical logic in the semantics of definite logic programs. Contrarily, the keystone idea of parametrized logic programming is the decoupling between the metalevel language and the parametrized logical language. In the metalevel language we have the usual constructors of logic programs rules: $\leftarrow, ", ", not\ and" or"$. In the parametrized logical language we have the language of the parameter logic $\mathcal{L}$.

The major challenge when envisaging such a general language is to define the semantics. Here, our key idea is to generalize the concept of interpretation with the notion of logical theory. This is in fact a non-trivial novel approach and it overcomes the difficulty of defining semantics of more complex logic programs. For example, it is well-known that classical interpretations are not suitable for defining the semantics of extended logic programs, thus leading to the artificial trick of considering classically negated atoms as new atoms.

As one application, we show that our approach can be very useful in the topic of combining rules and ontologies, which is an important topic in for the Semantic Web. In fact, we prove that one of the main frameworks for combining rules and ontologies for the semantic web, the MKNF semantics for hybrid knowledge bases, can be captured as a particular case of our approach, and, moreover, our language is richer than that of MKNF.

We start by introducing the novel definitions along with some motivating examples. We then define a general version of stable model semantics and of the well-founded semantics and prove that the non-parametrized is a special case obtained by a natural choice of the parameter logic. In the last section we prove that the semantics of MKNF knowledge bases is nothing but that of logic programs parametrized by the appropriate description logic. We end by drawing some conclusions and presenting some paths for future work.

2 Parametrized normal logic programs

In this section we introduce the syntax and semantics of normal parametrized logic programs. To focus on the novel key ideas we will not consider here epistemic disjunction *or*, along with its intrinsic difficulties[1].

2.1 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

[1] In fact, a results more general than this one, namely generalising for parametrized Here-There theories [13], is subject of current work.

of a normal parametrized logic program are replaced by formulas of a parameter 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 [16] 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 *theories of* $\mathcal{L}$, i.e. the set of subsets of L closed under the relation $\vdash_{\mathcal{L}}$. It is well-known 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 $A^{\vdash}$ the smallest theory that contains A. $A^{\vdash}$ is also called the theory generated by A.

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* φ, where as usual *not* denotes negation as failure. We dub *default literal* those of the form *not* φ.

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

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

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

2.2 Semantics

Given this general language of parametrized logic programs, we define its stable model semantics and its well-founded semantics, as generalisations of the stable model semantics [8] and well-founded semantics [7] of normal logic programs.

Interpretations and models In the traditional approach a (2-valued) interpretation is just a set of atoms. In our approach, since we substitute atoms by formulas of a parameter logic, the first idea is to take sets of formulas of the parameter logic as (2-valued) 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. To account for this interdependence, we use theories (sets of formulas closed under the consequence of the logic) as the generalisation of interpretations, thus capturing the above mentioned interdependence.

Definition 3. *A (parametrized) 2-valued interpretation is a theory of $\mathcal{L}$.*

As usual, a 2-valued interpretation T can be seen as a tuple $\langle T, F\rangle$ such that T is a theory of $\mathcal{L}$, and F is the complement, wrt L, of T. Note that, defined as such, F is not a theory, viz. it is not closed under the consequence of the logic. E.g. F does not, and should not, include tautologies in the parameter logic. This must be taken into account when defining parametrized 3-valued interpretations.

In 3-valued interpretations, as defined below, formulas are either true, undefined or false. True formulas, as we just seen, must be closed under the consequence of the logic; non-false formulas (i.e. true or undefined formulas) must also be closed; false formulas are just the complement of non-false formulas, and as such are not closed.

Definition 4. *A (parametrized) 3-valued interpretation is determined by any two theories T and T_U such $T \subseteq T_U$. For similarity with the usual definition of 3-valued interpretations, we represent one such interpretation as a tuple $\langle T, F\rangle$ where $F = L - T_U$.*

Any interpretation $I = \langle T, F\rangle$ can be equivalently defined as a function $I : L \to \{0, \frac{1}{2}, 1\}$ in the usual way. We dub *empty interpretation* the 3-valued interpretation $\langle \emptyset^{\vdash}, \emptyset\rangle$. Given an interpretation I we can extend it to literals by setting $I(not\ \varphi) = 1 - I(\varphi)$ for every $\varphi \in L$.

Definition 5. *An interpretation I satisfies a rule*

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

if $Min\{I(\psi_1), \ldots, I(\psi_n), I(not\ \delta_1), \ldots, I(not\ \delta_m)\} \leq I(\varphi)$.

An interpretation is a model of logic program P if it satisfies every rule of P. We denote by $Mod_2^{\mathcal{L}}(P)$ the set of 2-valued models of P and by $Mod_3^{\mathcal{L}}(P)$ the set of 3-valued models of P.

The usual orderings defined over 2- and 3-valued interpretations can easily be generalised. Moreover, given one such ordering, minimal and least interpretations may be defined in the usual way.

Definition 6 (Classical ordering). *If I and J are two interpretations then we say that $I \leq J$ if $I(\varphi) \leq J(\varphi)$ for every $\varphi \in L$.*

Definition 7 (Fitting ordering). *If $I_1 = \langle T_1, F_1\rangle$ and $I_2 = \langle T_2, F_2\rangle$ are two 3-valued interpretations then we say that $I_1 \leq_F I_2$ if $T_1 \subseteq T_2$ and $F_1 \subseteq F_2$.*

Stable model semantics As in the case of non-parametrized, we start by assigning semantics to definite parametrized programs. The stable model of a definite program is its least 2-valued model. In order to generalise this definition to the parametrized case we need to prove that the least parametrized 2-valued model exists for every definite $\mathcal{L}$-parametrized logic program.

Theorem 1. *Every definite $\mathcal{L}$-parametrized logic program has a least 2-valued model.*

Proof. Consider the set $S_P^{\mathcal{L}} = \bigcap_{M \in Mod_2^{\mathcal{L}}(P)} M$. Note that the set $S_P^{\mathcal{L}}$ always exists since $\langle Th_{\mathcal{L}}, \subseteq \rangle$ is a complete lattice for every (monotonic) logic $\mathcal{L}$. It is clear that $S_P^{\mathcal{L}}$ is included in every model of P and it is trivial to prove that $S_P^{\mathcal{L}}$ is still a model of P.

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

To define the stable model semantics of a normal $\mathcal{L}$-parametrized logic programs we use a Gelfond-Lifschitz like operator.

Definition 8. *Let P be a normal $\mathcal{L}$-parametrized logic program and I a 2-valued interpretation. The GL-transformation of P modulo I is the program $\frac{P}{I}$ obtained from P by performing the following operations:*

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

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

Definition 9. *A 2-valued interpretation I of a $\mathcal{L}$-parametrized logic program P is a stable model of P iff $\Gamma(I) = I$. A formula φ is true under the stable model semantics iff it belongs to all stable models of P.*

Well-founded semantics Several equivalent definitions of well-founded semantics exist in the literature. In this work we follow the iterated fixed-point approach of [6].

First of all it can be readily proved that, as in the non-parametrized case, the operator Γ is antitonic, i.e, for any theories $I, J \in \mathcal{I}$ we have that if $I \subseteq J$ then $\Gamma(J) \subseteq \Gamma(I)$. Therefore, applying Γ twice, denoted by Γ^2, yields a monotone operator on the lattice of theories.

Definition 10. *A 3-valued interpretation $\langle T, F \rangle$ is a partial stable model of a normal $\mathcal{L}$-parametrized logic program P if $T = \Gamma^2(T)$ and $F = L - \Gamma(T)$.*

For defining the well-founded semantics, we first need to prove that the F-least partial stable model of a normal $\mathcal{L}$-parametrized logic program always exits.

Theorem 2. *Every normal $\mathcal{L}$-parametrized logic program has a unique F-least partial stable model.*

Proof. Let P be a normal $\mathcal{L}$-parametrized logic program. Since Γ^2 is a monotone operator we can conclude by the Knaster-Tarski thereom that Γ^2 has a least fixed-point which we denote by T^*. Consider the 3-valued interpretation $I^* = \langle T^*, L - \Gamma(T^*) \rangle$. We now prove that I^* is in fact the F-least partial stable model of P. By definition I^* is a partial stable model of P. We still have to prove that it is the F-least. Let $I = \langle T, F \rangle$ be a partial stable model of P, i.e., $\Gamma^2(T) = T$ and $F = L - \Gamma(T)$. We have that $T^* \subseteq T$ since T^* is the least fixed-point of Γ^2. Using the fact that Γ is antitonic we have that $\Gamma(T) \subseteq \Gamma(T^*)$. Therefore, $L - \Gamma(T^*) \subseteq L - \Gamma(T)$, thus proving that $I^* \leq_F I$.

We can now define the well-founded semantics of a normal $\mathcal{L}$-parametrized logic program.

Definition 11. *The well-founded semantics of a normal $\mathcal{L}$-parametrized logic program P is defined by its unique F-least partial stable model.*

2.3 Particular cases

For soundness reasons it is important to show that the non-parametrized case is a special case of our approach, using an appropriate choice of the parameter logic. This is indeed the case for normal and extended logic programs.

Normal logic programs Suppose that we fix an alphabet $\mathcal{A}$ for building the language of normal logic programs. Since we want the class of $\mathcal{L}$-parametrized logic programs to coincide with the class of normal logic programs over $\mathcal{A}$, then the language L of the parameter logic $\mathcal{L}$ must be the set of atoms over $\mathcal{A}$. We then have that $L = \mathcal{H}$, where $\mathcal{H}$ is the Herbrand base of $\mathcal{A}$. Now that we have fixed the language L of $\mathcal{L}$ we still have to define the Tarskian consequence relation $\vdash_{\mathcal{L}}$ over L. At first sight it seems that we have a large range of possibilities for defining $\vdash_{\mathcal{L}}$. Nevertheless, the low expressivity of the language enforces that the only reasonable consequence relation we can choose is the trivial one, i.e. for every $T \cup \{\varphi\} \subseteq L$ we have that $T \vdash_{\mathcal{L}} \varphi$ iff $\varphi \in T$. Therefore, $Th(\mathcal{L}) = \mathcal{P}(L)$, i.e. the theories of $\mathcal{L}$, are precisely the sets of atoms over $\mathcal{A}$. Recall that the notion of parametrized interpretation, both 2 or 3-valued, is based on theories of $\mathcal{L}$. Therefore, by construction, the parametrized versions of stable model semantics and well-founded semantics coincide with the classical ones for normal logic programs.

Extended logic programs Suppose that we fix an alphabet $\mathcal{A}$ for building the language of extended logic programs. This language is enriched with a second negation $\neg$, usually called explicit negation, which is allowed to appear in front of the atoms. An objective literal is either an atom or the explicit negation of an atom.

Recall that we want the class of $\mathcal{L}$-parametrized logic programs to coincide with the class of extended logic programs over $\mathcal{A}$. Therefore, we need to take the language L of the parameter logic $\mathcal{L} = \langle L, \vdash \rangle$ as $L = \mathcal{H} \cup \{\neg p : p \in \mathcal{H}\}$. The set $\mathcal{H} \cup \{\neg p : p \in \mathcal{H}\}$ is usually called the extended Herbrand base of P.

In the case of programs with explicit negation there is no a general consensus with respect to the semantics. In fact, there are two main approaches: the classical one that assumes the explosion principle of negation; and the paraconsistent approach that rejects the explosion principle of negation.

It is very interesting that our parametrized approach allows to easily explain why two different approaches to the semantics of extended logic programs exist. In fact, this can be explained by the fact that only two consequence relations can naturally be defined over the language L.

The first consequence relation over L, denoted by $\vdash_1$, assumes the explosion principle and is such that, for every $T \cup \varphi$, we have $T \vdash_1 \varphi$ if $\varphi \in T$ or $\{p, \neg p\} \subseteq T$ for some atom p. It can be readily proved that, taking $\mathcal{L}_1 = \langle L, \vdash_1 \rangle$, the $\mathcal{L}_1$-parametrized stable model semantics of an extended logic program coincides with its answer set semantics. The second consequence relation over L, denoted by $\vdash_2$, does not assume the explosion principle and, therefore, is such that, for every $T \cup \varphi$, we have $T \vdash_2 \varphi$ if $\varphi \in T$. The consequence relation $\vdash_2$ is in fact the 4-valued Belnap paraconsistent logic *Four* restricted to this more restricted language. It can be readily proved that, taking $\mathcal{L}_2 = \langle L, \vdash_2 \rangle$, the $\mathcal{L}_2$-parametrized stable model semantics of an extended logic program coincides with its paraconsistent answer set semantics.

Beyond extended logic programs Let us now consider a full classical language L built over a set $\mathcal{P}$ of propositional symbols using the usual connectives $(\neg, \vee, \wedge, \Rightarrow)$. Of course, many consequence relations can be defined over this language. Here we only focus on classical logic, Belnap's paraconsistent logic and intuitionistic logic. Consider the following programs:

$$P_1 \begin{cases} p \leftarrow \neg q \\ p \leftarrow q \end{cases} \qquad P_2 \ \{ p \leftarrow \neg q \vee q \qquad P_3 \begin{cases} q \leftarrow \\ (q \vee s) \Rightarrow p \leftarrow \\ r \leftarrow p \end{cases}$$

$$P_4 \begin{cases} r \leftarrow \\ \neg p \leftarrow \\ (p \vee q) \leftarrow r \\ s \leftarrow q \end{cases} \qquad P_5 \ \{ p \leftarrow not\ q, not\ \neg q \qquad P_6 \ \{ p \leftarrow not\ (q \vee \neg q)$$

Example 1 (Classical logic).

Let $\mathcal{L} = \langle L, \vdash_{CPL} \rangle$ be Classical Propositional Logic (CPL) over the language L. Let us study in detail the semantics of P_1. Note that every theory of CPL that does not contain neither p nor $\neg p$ satisfies P_1. In particular, the set $Theo$ of theorems of CPL is a model of P_1. So, $S^{CPL}_{P_1} = Theo$. This means that $p, \neg p, q, \neg q \notin S^{CPL}_{P_1}$.

Using the same kind of reasoning we can conclude that $S^{CPL}_{P_2} = \{p\}^{\vdash}$. So, in the case of P_2 we have that $p \in S^{CPL}_{P_2}$. We can also easily conclude 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 theory T of CPL we can conclude that the only stable model of P_6 is the set $Theo$ of theorems of CPL. Therefore $p \notin S^{CPL}_{P_6}$.

Example 2 (Paraconsistent logic).

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 $q, s \notin S^{Four}_{P_4}$.

Example 3 (Intuitionistic logic).

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 $S_{P_2}^{IPL} = \emptyset^{\vdash_{IPL}}$. So, contrarily to the case of CPL, we have that $p \notin S_{P_2}^{IPL}$. Using the same idea for program P_6 we can conclude, contrarily to the case of CPL, that $p \in S_{P_6}^{IPL}$.

3 MKNF

In this section we show that our approach is general enough to capture the semantics of MKNF hybrid knowledge bases [12, 14, 10], which tightly combines normal logic programs with description logic (DL) based ontologies. More precisely, the goal of this section is to prove that, taking first-order logic as the parameter logic, and translating every DL formula in the ontology into a fact of the corresponding parametrized program, we exactly obtain the MKNF semantics. Moreover, by doing so, the expressiveness of the language can naturally be extended by allowing complex DL formulas to appear anywhere in rules, as it happens e.g. in [11] but in this latter one only for definite programs.

Before we come to this main result, we recall some basic notions of MKNF, and then prove some required intermediate results.

Let $\mathcal{O}$ be a DL database. Consider a first-order signature Σ such that Σ contains the equality predicate $\approx$, all atomic concepts from $\mathcal{O}$ as unary predicates, all atomic roles from $\mathcal{O}$ as binary predicates and all individuals of $\mathcal{O}$ as constants. Let $FOL_\Sigma = \langle L_\Sigma, \vdash_{FOL_\Sigma} \rangle$ denote first-oder logic over the first-order language L_Σ obtained from Σ. As it is usual in the MKNF approach, we assume that $\mathcal{O}$ can be translated into a formula $\pi(\mathcal{O})$ of function free first-order logic over Σ. Given a MKNF knowledge base $\mathcal{K} = \langle \mathcal{O}, P \rangle$ we denote by $P_\mathcal{K}$ the FOL_Σ-parametrized logic program obtained from $\mathcal{K}$ such that $P_\mathcal{K} = P \cup \{\pi(\mathcal{O}) \leftarrow\}$. We denote by $\mathcal{I}$ be the set of all Herbrand interpretations over Σ and by Δ the Herbrand universe of Σ. Following [14] we also assume that in every member of $\mathcal{I}$ the equality predicate $\approx$ is interpreted as a congruence relation. Recall that the language of MKNF, here denoted by $L_\Sigma^{\mathbf{K}}$, is obtained by extending the first-order language over Σ with the modal operators $\mathbf{K}$ and **not**. A MKNF structure is a triple $\langle I, M, N \rangle$ where $M \cup N \cup \{I\} \subseteq \mathcal{I}$ and M and N are non-empty. Satisfiability of MKNF formulas in a MKNF structure $\langle I, M, N \rangle$ is defined as follows:

$\langle I, M, N \rangle \Vdash p(t_1, \ldots, t_n)$ iff $p(t_1, \ldots, t_n)$ is true in I
$\langle I, M, N \rangle \Vdash \neg\varphi$ iff $\langle I, M, N \rangle \nVdash \varphi$
$\langle I, M, N \rangle \Vdash \varphi \wedge \psi$ iff $\langle I, M, N \rangle \Vdash \varphi$ and $\langle I, M, N \rangle \Vdash \psi$
$\langle I, M, N \rangle \Vdash \exists x\varphi$ iff $\langle I, M, N \rangle \Vdash \varphi[\alpha/x]$ for some $\alpha \in \Delta$
$\langle I, M, N \rangle \Vdash \mathbf{K}\varphi$ iff $\langle J, M, N \rangle \Vdash \varphi$ for all $J \in M$
$\langle I, M, N \rangle \Vdash \mathbf{not}\varphi$ iff $\langle J, M, N \rangle \nVdash \varphi$ for some $J \in N$

A MKNF interpretation is a nonempty set $M \subseteq \mathcal{I}$. A MKNF interpretation M is a model of a formula φ, denoted by $M \Vdash \varphi$, if $\langle I, M, M \rangle \Vdash \varphi$ for every $I \in M$, and for each MKNF interpretation M' such that $M \subset M'$, we have that $\langle I', M', M \rangle \nVdash \varphi$ for some $I' \in M'$.

We write $M \Vdash T$ to denote that for every $I \in M$, we have that $I \Vdash \varphi$ for every $\varphi \in T$.

For $M \subseteq \mathcal{I}$ consider $T_M = \{\varphi \in L_\Sigma : I \Vdash \varphi \text{ for every } I \in M\}$. Conversely, given $T \subseteq L_\Sigma$ consider $M_T = \{I \in \mathcal{I} : I \Vdash \varphi \text{ for every } \varphi \in T\}$. It is easy to see that if $M \subseteq M'$ then $T_{M'} \subseteq T_M$. Conversely, if $T \subseteq T'$ then $M_{T'} \subseteq M_T$. It is also easy to see that the inclusions $M \subseteq M_{T_M}$ and $T \subseteq T_{M_T}$ always hold, whereas the respective converse inclusions do not always hold. When the first inclusion holds, that is $M = M_{T_M}$, the set M is called *complete*. Let $\mathcal{M}$ denote the set of all complete subsets of $\mathcal{I}$. It is easy to see that the second converse inclusion, $T_{M_T} \subseteq T$, holds for every $T \in Th(FOL_\Sigma)$.

Lemma 1. *The function $\phi : \mathcal{M} \to Th(FOL_\Sigma)$ such that $M \mapsto T_M$ is a bijection.*

Proof. We start by proving that ϕ is well-defined, that is, that T_M is a FOL_Σ-theory. Suppose that $T_M \vdash_{FOL_\Sigma} \varphi$. Then, using the fact that M is complete, we have the following sequence of equivalent sentences:
for every $I \in \mathcal{M}$, if $I \Vdash \delta$ for every $\delta \in T_M$ then $I \Vdash \varphi$ iff for every $I \in \mathcal{M}$, if $I \in M_{T_M}$ then $I \Vdash \varphi$ iff for every $I \in \mathcal{M}$, if $I \in M$ then $I \Vdash \varphi$. Therefore, we have that $\varphi \in T_M$ and we can conclude that T_M is a FOL_Σ-theory.

Let us now prove that ϕ is injective. Let $M_1, M_2 \in \mathcal{M}$ such that $T_{M_1} = T_{M_2}$. Clearly we have that $M_{T_{M_1}} = M_{T_{M_2}}$. Since M_1, M_2 are complete we have that $M_1 = M_{T_{M_1}} = M_{T_{M_2}} = M_2$.

It remains to be proved that ϕ is surjective. Let T be a FOL_Σ-theory. First we prove that M_T is complete. Since the inclusion $M_T \subseteq M_{T_{M_T}}$ always holds, it remains to prove the converse inclusion. Let $I \in M_{T_{M_T}}$. Then, $I \Vdash T_{M_T}$. Since $T \subseteq T_{M_T}$ we have that $I \Vdash T$. Therefore, $I \in M_T$ and the inclusion $M_{T_{M_T}} \subseteq M_T$ holds. It just remains to be proved that $\phi(M_T) = T$, that is, $T_{M_T} = T$. We prove the inclusion $T_{M_T} \subseteq T$ since the reverse one always holds. Let $\varphi \in T_{M_T}$. Then $M_T \Vdash \varphi$ and by definition $T \vdash_{FOL_\Sigma} \varphi$. Since T is a FOL_Σ-theory we conclude that $\varphi \in T$.

Lemma 2. *Let $M, M' \in \mathcal{M}$ and $T, T' \in Th(FOL_\Sigma)$. Then*

1. *$T \subset T'$ iff $M_{T'} \subset M_T$;*
2. *$M \subset M'$ iff $T_{M'} \subset T_M$.*

Proof. 1. Suppose $T \subset T'$. Then, if $I \in M_{T'}$ then $I \in M_T$. Suppose now that $M_{T'} \subset M_T$. Using Lemma 1 we have that $T_{M_T} = T \subset T' = T_{M_{T'}}$. 2. Suppose $M \subset M'$. Then $T_{M'} \subset T_M$. Suppose now that $T_{M'} \subset T_M$. Using Lemma 1 we have that $M_{T_M} = M \subset M' = M_{T_{M'}}$

Lemma 3. *Let $M \cup M' \cup \{I\} \subseteq \mathcal{I}$, $T \in Th(FOL_\Sigma)$ and $\varphi \in L_\Sigma$. Then,*

1. *$\langle I, M_T, M' \rangle \Vdash \mathbf{K}\varphi$ iff $\varphi \in T$;*
2. *$\langle I, M, M' \rangle \Vdash \mathbf{K}\varphi$ iff $\varphi \in T_M$;*
3. *$\langle I, M', M_T \rangle \Vdash \mathbf{not}\varphi$ iff $\varphi \notin T$;*
4. *$\langle I, M', M \rangle \Vdash \mathbf{not}\varphi$ iff $\varphi \notin T_M$.*

Proof. 1. $\langle I, M_T, M' \rangle \Vdash \mathbf{K}\varphi$ iff $M_T \Vdash \varphi$ iff $\varphi \in T$; 2. $\langle I, M, M' \rangle \Vdash \mathbf{K}\varphi$ iff $M \Vdash \varphi$ iff $\varphi \in T_M$; 3. $\langle I, M', M_T \rangle \Vdash \mathbf{not}\varphi$ iff there exists $I' \in M_T$ such that $I' \nVdash \varphi$ iff (since $T_{M_T} = T$) $\varphi \notin T$; 4. $\langle I, M', M \rangle \Vdash \mathbf{not}\varphi$ iff there exists $I' \in M$ such that $I' \nVdash \varphi$ iff $\varphi \notin T_M$.

Given $\Phi \subseteq L_\Sigma^{\mathbf{K}}$ and $M \subseteq \mathcal{I}$ consider $\Gamma(\Phi, M) = \{M' \subseteq \mathcal{I} : M' \text{ is maximal such that } \langle I, M', M \rangle \Vdash \Phi, \text{ for every } I \in M'\}$. Note that, by definition, M is a MKNF model of Φ iff $M \in \Gamma(\Phi, M)$. Moreover, it can be readily proved that if M is a MKNF model of some $\Phi \subseteq L_\Sigma^{\mathbf{K}}$ then M is complete. Given $\Phi \subseteq L_\Sigma^{\mathbf{K}}$ and $T \in Th(FOL_\Sigma)$ consider also the set $\Gamma_0(\Phi, T) = \{T' \in Th(FOL_\Sigma) : T' \text{ minimal such that } \langle I, M_{T'}, M_T \rangle \Vdash \Phi, \text{ for every } I \in M_{T'}\}$.

Lemma 4. *Let $M \cup M' \cup \{I\} \subseteq \mathcal{I}$ and P a normal FOL_Σ-parametrized logic program. Then, $\langle I, M, M' \rangle \Vdash \pi(P)$ iff $\langle I, M_{T_M}, M' \rangle \Vdash \pi(P)$.*

Proof. Recall that $\pi(P) = \bigwedge_{r \in P} \pi(r)$, and that if $r = \varphi \leftarrow \psi_1, \ldots, \psi_n, not\ \varphi_1, \ldots, not\ \varphi_m$ then $\pi(r) = (\mathbf{K}\psi_1 \wedge \ldots \wedge \mathbf{K}\psi_n \wedge \mathbf{not}\varphi_1 \wedge \ldots \wedge \mathbf{not}\varphi_m) \Rightarrow \mathbf{K}\varphi$. Consider the following sequence of equivalent conditions: $\langle I, M, M' \rangle \Vdash \pi(r)$ iff $\langle I, M, M' \rangle \Vdash \mathbf{K}\varphi$ whenever $\langle I, M, M' \rangle \Vdash \mathbf{not}\varphi_i$ for every $i \in \{1, \ldots, m\}$ and $\langle I, M, M' \rangle \Vdash \mathbf{K}\psi_j$ for every $j \in \{1, \ldots, n\}$ iff (Lemma 3) $\varphi \in T_M$ whenever $\varphi_i \notin T_{M'}$ for every $i \in \{1, \ldots, m\}$ and $\psi_j \in T_M$ for every $j \in \{1, \ldots, n\}$ iff (Lemma 3) $\langle I, M_{T_M}, M' \rangle \Vdash \mathbf{K}\varphi$ whenever $\langle I, M_{T_M}, M' \rangle \Vdash \mathbf{not}\varphi_i$ for every $i \in \{1, \ldots, m\}$ and $\langle I, M_{T_M}, M' \rangle \Vdash \mathbf{K}\psi_j$ for every $j \in \{1, \ldots, n\}$ iff $\langle I, M_{T_M}, M' \rangle \Vdash \pi(r)$.

Proposition 1. *Let $M \in \mathcal{M}$, $I \in \mathcal{I}$ and P a normal FOL_Σ-parametrized logic program. Then, $M \in \Gamma(\pi(P), M)$ iff $T_M \in \Gamma_0(\pi(P), T_M)$.*

Proof. Let us first assume that $M \in \Gamma(\pi(P), M)$. Then M is maximal such that $\langle I, M, M \rangle \Vdash \pi(P)$ for every $I \in M$. Therefore, $\langle I, M_{T_M}, M_{T_M} \rangle \Vdash \pi(P)$. We still have to prove the minimality condition for T_M. Suppose there exists $T' \subset T_M$ such that $\langle I, M_{T'}, M_{T_M} \rangle \Vdash \pi(P)$. Then, using Lemma 2 we have $M_{T_M} = M \subset M_{T'}$, which contradicts the maximality of M. Therefore, we can conclude that $T_M \in \Gamma_0(\pi(P), T_M)$.

Suppose now that $T_M \in \Gamma_0(\pi(P), T_M)$. Then T_M is minimal such that $\langle I, M_{T_M}, M_{T_M} \rangle \Vdash \pi(P)$. Since $M \in \mathcal{M}$, i.e., $M = M_{T_M}$, we have that $\langle I, M, M \rangle \Vdash \pi(P)$. We still have to prove the maximality condition for M. Suppose there exists $M \subset M'$ such that $\langle I, M', M \rangle \Vdash \pi(P)$. Using Lemma 4 we have that $\langle I, M_{T_{M'}}, M_{T_M} \rangle \Vdash \pi(P)$. Using Lemma 2 we have that $T_{M'} \subset T_M$, which contradicts the minimality condition of T_M. Therefore, we can conclude that $M \in \Gamma(\pi(P), M)$.

Lemma 5. *Let $T \in Th(FOL_\Sigma)$ and P a normal FOL_Σ-parametrized logic program. Then, $\Gamma_0(\pi(\frac{P}{T}), T) = \Gamma_0(\pi(P), T)$.*

Proof. Let $T \in Th(FOL_\Sigma)$. It suffices to prove that $\langle I, M_{T'}, M_T \rangle \Vdash \pi(P)$ iff $\langle I, M_{T'}, M_T \rangle \Vdash \pi(\frac{P}{T})$. Assume first that $\langle I, M_{T'}, M_T \rangle \Vdash \pi(P)$. Recall that every rule $r = \varphi \leftarrow \psi_1, \ldots, \psi_n$ of $\frac{P}{T}$ is obtained from a rule $\overline{r} = \varphi \leftarrow \psi_1, \ldots, \psi_n, not\ \varphi_1, \ldots, not\ \varphi_m$ of P such that $\varphi_i \notin T$ for every $i \in \{1, \ldots, m\}$. Then, using

Lemma 3 we have that $\langle I, M_{T'}, M_T\rangle \Vdash \mathbf{not}\varphi_i$ for every $i \in \{1,\ldots,m\}$. Therefore, $\langle I, M_{T'}, M_T\rangle \Vdash \pi(r)$ iff $\langle I, M_{T'}, M_T\rangle \Vdash \pi(\overline{r})$. Since we are assuming that $\langle I, M_{T'}, M_T\rangle \Vdash \pi(P)$ we can conclude that $\langle I, M_{T'}, M_T\rangle \Vdash \pi(\frac{P}{T})$.

Let us now assume that $\langle I, M_{T'}, M_T\rangle \Vdash \pi(\frac{P}{T})$ and consider the rule $r = \varphi \leftarrow \psi_1,\ldots,\psi_n, not\ \varphi_1,\ldots,not\ \varphi_m$ of P. We have two cases:

Case 1: $\varphi_i \in T$ for some $i \in \{1,\ldots,m\}$. Then, by Lemma 3 we have that $\langle I, M_{T'}, M_T\rangle \nVdash \mathbf{not}\varphi_i$. Therefore, $\langle I, M_{T'}, M_T\rangle \Vdash \pi(r)$.

Case 2: $\varphi_i \notin T$ for every $i \in \{1,\ldots,m\}$. Then, by Lemma 3 we have that $\langle I, M_{T'}, M_T\rangle \Vdash \mathbf{not}\varphi_i$ for every $i \in \{1,\ldots,m\}$. We then have that $\langle I, M_{T'}, M_T\rangle \Vdash \pi(r)$ iff $\langle I, M_{T'}, M_T\rangle \Vdash \pi(\overline{r})$, where $\overline{r} = \varphi \leftarrow \psi_1,\ldots,\psi_n \in \frac{P}{T}$. Since we are assuming that $\langle I, M_{T'}, M_T\rangle \Vdash r'$ for every rule r' of $\pi(\frac{P}{T})$ we can conclude that $\langle I, M_{T'}, M_T\rangle \Vdash \pi(r)$.

Therefore we can conclude that $\langle I, M_{T'}, M_T\rangle \Vdash \pi(P)$.

Lemma 6. *Let $T \in Th(FOL_\Sigma)$ and P a definite FOL_Σ-parametrized logic program. Then $\Gamma_0(\pi(P), T)$ is the set of all FOL_Σ-parametrized stable models of P.*

Proof. Let $T \in Th(FOL_\Sigma)$. Then we have the following sequence of equivalent sentences: T is a FOL_Σ-parametrized stable model of P iff T is minimal such that, for every rule $r = \varphi \leftarrow \psi_1,\ldots,\psi_n$, we have that $\varphi \in T$ whenever $\psi_i \in T$ for every $i \in \{1,\ldots,n\}$ iff (Lemma 3) T is minimal such that, for every rule $r = \varphi \leftarrow \psi_1,\ldots,\psi_n$, we have that $\langle I, M_T, M_T\rangle \Vdash \mathbf{K}\varphi$ whenever $\langle I, M_T, M_T\rangle \Vdash \mathbf{K}\psi_i$ for every $i \in \{1,\ldots,n\}$ iff T is minimal such that $\langle I, M_T, M_T\rangle \Vdash \pi(r)$ for every rule r of P iff $T \in \Gamma_0(\pi(P), T)$.

Theorem 3. *M is a MKNF model of $\mathcal{K}$ iff T_M is a FOL_Σ-parametrized stable model of $P_\mathcal{K}$.*

Proof. Just consider the following sequence of equivalent sentences:
M is a MKNF model of $\mathcal{K}$ iff M is a MKNF model of $\pi(P_\mathcal{K})$ iff $M \in \Gamma(\pi(P_\mathcal{K}), M)$ iff (Proposition 1) $T_M \in \Gamma_0(\pi(P_\mathcal{K}), T_M)$ iff (Lemma 5) $T_M \in \Gamma_0(\pi(\frac{P_\mathcal{K}}{T_M}), T_M)$ iff (Lemma 6) T_M is a FOL_Σ-parametrized stable model of $\frac{P_\mathcal{K}}{T_M}$ iff (by definition) T_M is a FOL_Σ-parametrized stable model of $P_\mathcal{K}$.

Using Lemma 1 the following corollary of Theorem 3 is immediate.

Corollary 1. *$T \in Th(FOL_\Sigma)$ is a FOL_Σ-parametrized stable model of $P_\mathcal{K}$ iff M_T is a MKNF model of $\mathcal{K}$.*

Example 4. Let $\mathcal{L}$ be a description logic seen as a fragment of FOL. The following program (P_7) is an adaptation of an example taken from [14].

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

Note that in our approach 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. In fact, we are able to rewrite P_7 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_7 above, to state that non married are only considered high-risk in non exceptional periods, obtaining P_8:

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

Let us study the stable model semantics of this program. If I is a 2-valued interpretation such that $I(Married(Jonh)) = 1$ then $\Gamma(I)$ is the least model of the following program $\frac{P_8}{I}$:
$\neg Married \sqsubseteq HighRisk \leftarrow$
$\exists Spouse.\top \sqsubseteq Married \leftarrow$
$Discount(x) \leftarrow Spouse(x,y), p(x), p(y)$ $\quad p(Jonh) \leftarrow$

It is clear that the smallest model of $\frac{P_8}{I}$ 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 $HighRisk(Jonh)$.

Suppose that we obtain P_9 by adding to P_8 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_9 contains $Married(Bill)$. So, the Stable Model Semantics of P_9 does not entail $\neg Married(Bill)$ nor $HighRisk(Bill)$.

Suppose now that instead we add to P_8 the facts: $Spouse(Bob, Ann) \leftarrow$, $p(Bob) \leftarrow$, and $p(Ann) \leftarrow$. Every stable model now contains $Discount(Bob)$, and so the Stable Model Semantics entails $Discount(Bob)$.

4 Conclusions

We have introduced the novel notion of parametrized logic program along with several motivating examples, and showed how some usual approaches to the semantics of logic programs can be obtained as particular choices of the parameter logic. We gave a contribution to the important problem of combining rules and ontologies, by capturing MKNF semantics of hybrid knowledge bases as particular case and, moreover, by extending the expressivity of its language. Note that, as a consequence, we are also able to capture Description Logic Programs (DLP) [9] since we can write arbitrary DL formulas in the heads and bodies of rules. Though here we only explored the stable models semantics of MKNF knowledge bases, our approach also naturally yields a well founded semantics for such knowledge bases.

The work raises several interesting paths for future research. One, which is already ongoing is the generalization to the parametrized case of the here-there

theories [15], this way allowing for combining in a general way logic programming connectives with formulas of a parameter logic. This work would also generalise for any parameter logic the general default logic of [17], which is fixed for propositional classical logic. Future work also includes a detailed study of particular choices of the parameter logic which seem very promising. One such example is the case where the parameter logic is temporal logic, to obtain logic programs expressive enough to reason about temporal logic formulas.

This first paper on parametrized logic programs is focused on the definition of the semantics and on showing its interest for capturing at least one known combination of logic programming with other logics (viz. description logic in MKNF) and, as such, left out a study on decidability and complexity issues, which must also be subject of future work.

References

1. Carlos Caleiro, Cristina Sernadas, and Amílcar Sernadas. Parameterisation of logics. In *WADT*, pages 48–62, 1998.
2. W. A. Carnielli, M. E. Coniglio, D. Gabbay, P. Gouveia, and C. Sernadas. *Analysis and Synthesis of Logics - How To Cut And Paste Reasoning Systems*, volume 35 of *Applied Logic*. Springer, 2008.
3. Carlos Viegas Damásio and Luís Moniz Pereira. Antitonic logic programs. In *LPNMR*, pages 379–392, 2001.
4. Melvin Fitting. Bilattices and the semantics of logic programming. *J. Log. Program.*, 11(2):91–116, 1991.
5. Dov Gabbay. Fibring logics. *Oxford University Press*, 1999.
6. Allen Van Gelder. The alternating fixpoint of logic programs with negation. *J. Comput. Syst. Sci.*, 47(1):185–221, 1993.
7. Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991.
8. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, pages 1070–1080, 1988.
9. Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: combining logic programs with description logic. In *WWW*, pages 48–57, 2003.
10. Matthias Knorr, José Júlio Alferes, and Pascal Hitzler. A well-founded semantics for hybrid mknf knowledge bases. In *Description Logics*, 2007.
11. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Elp: Tractable rules for owl 2. In *International Semantic Web Conference*, pages 649–664, 2008.
12. Vladimir Lifschitz. Minimal belief and negation as failure. *Artif. Intell.*, 70(1-2):53–72, 1994.
13. Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. *ACM Trans. Comput. Log.*, 2(4):526–541, 2001.
14. Boris Motik and Riccardo Rosati. A faithful integration of description logics with logic programming. In *IJCAI*, pages 477–482, 2007.
15. David Pearce. Equilibrium logic. *Ann. Math. Artif. Intell.*, 47(1-2):3–41, 2006.
16. R. Wójcicki. *Theory of Logical Calculi*. Synthese Library. Kluwer Academic Publishers, 1988.
17. Yi Zhou, Fangzhen Lin, and Yan Zhang. General default logic. *Annals of Mathematics and Artificial Intelligence*, 57(2):125–160, 2009.