Integrating ontologies and rules: semantic and computational issues

Riccardo Rosati

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
rosati@dis.uniroma1.it

Abstract. We present some recent results on the definition of logic-based systems integrating ontologies and rules. In particular, we take into account ontologies expressed in Description Logics and rules expressed in Datalog (and its nonmonotonic extensions). We first introduce the main issues that arise in the integration of ontologies and rules. In particular, we focus on the following aspects: (i) from the semantic viewpoint, ontologies are based on open-world semantics, while rules are typically interpreted under closed-world semantics. This semantic discrepancy constitutes an important obstacle for the definition of a meaningful combination of ontologies and rules; (ii) from the reasoning viewpoint, the interaction between an ontology and a rule component is very hard to handle, and does not preserve decidability and computational properties: e.g., starting from an ontology in which reasoning is decidable and a rule base in which reasoning is decidable, reasoning in the formal system obtained by integrating the two components may not be a decidable problem. Then, we briefly survey the main approaches for the integration of ontologies and rules, with special emphasis on how they deal with the above mentioned issues, and present in detail one of such approaches, i.e., $\mathcal{DL}$+*log*. Finally, we illustrate the main open problems in this research area, pointing out what still prevents us from the development of both effective and expressive systems able to integrate ontologies and rules.

1 Introduction

1.1 Ontologies and Description Logics

The integration of ontologies and rules has recently received considerable attention in the research on ontologies and the Semantic Web (see e.g.,[24, 2]). Description Logics (DLs) [6] are currently playing a central role in this field. DLs are a family of knowledge representation formalisms based on first-order logic (in fact, almost all DLs coincide with decidable fragments of function-free first-order logic with equality) and exhibiting well-understood computational properties. In the last years, a significant body of the Semantic Web research was devoted to defining a suitable language for ontology modeling [33]. In 2004, this endeavor resulted in the Web Ontology Language (OWL). OWL is based on Description

Logics, and has successfully been applied to numerous problems in computer science, such as information integration or metadata management. Prototypes of OWL reasoners, such as RACER, FaCT++, Pellet, or KAON2, have been implemented and applied in research projects; commercial implementations and projects using them are currently emerging.

1.2 Limitations of current ontology formalisms

However, the experience in building practical applications has revealed several shortcomings of OWL and, in general, of Description Logics. In particular, the typical expressiveness of DLs does not allow for addressing the following aspects:

- the possibility of defining predicates of arbitrary arity (not just unary and binary)
- the use of variable quantification beyond the tree-like structure of DL concepts (many DLs actually correspond to subsets of the two-variable fragment of first-order logic)
- the possibility of formulating expressive queries over DL knowledge bases (beyond concept subsumption and instance checking)
- the possibility of formalizing various forms of closed-world reasoning over DL knowledge bases
- more generally, the possibility of expressing forms of nonmonotonic knowledge, like default rules [34]

The issue of how to overcome these limitations of OWL and DLs is currently receiving a lot of attention in the Semantic Web community [1]. In this respect, we observe that several of the representational abilities which are missing in DLs require *nonmonotonicity* of the underlying logical formalism. This is in contrast with the well-known monotonic nature of classical first-order logic, which corresponds to the following property: if a theory T entails a conclusion ϕ, then, for every formula ψ, the theory $T \cup \{\psi\}$ entails ϕ. Such a property dos not hold anymore in the presence of closed-world knowledge and default knowledge [34, 7].

This implies that the attempt to extend the expressive abilites of DLs, in order to fully overcome the above limitations, requires to leave the realm of classical first-order logic, and to look at nonmonotonic logic.

1.3 Rule-based knowledge representation

Almost all the kinds of knowledge that cannot be formally addressed in a classical, first-order logic setting have a "rule-like" form, i.e., can be expressed by statements of the form "if the precondition ψ holds then the conclusion ϕ holds", where the precondition and the conclusion are logical properties.

However, such a piece of knowledge cannot simply be formalized through the standard material implication of classical logic: in other words, it is not possible

to capture the intended meaning of the above statement by an implication in classical first-order logic of the form $\psi \rightarrow \phi$.

In this respect, a very relevant role is played by research in logic programming. In fact, logic program rules are implications with a non-standard semantics. And, in the context of ontologies, nonmonotonic extensions of logic programming are of particular interest [7].

Therefore, rule-based formalisms grounded in logic programming have repeatedly been proposed as a possible solution to overcome the above limitations, so adding a rule layer on top of OWL is nowadays seen as the most important task in the development of the Semantic Web language stack. The Rule Interchange Format (RIF) working group of the World Wide Web Consortium (W3C) is currently working on standardizing such a language.

Most of the proposals in this field focus on logic programs expressed in Datalog (and its nonmonotonic extensions) [14]. With respect to DLs, Datalog allows for using predicates of arbitrary arity, the explicit use of variables, and the ability of expressing more powerful queries. Moreover, its nonmonotonic features (in particular, the negation-as-failure operator *not*) allow for expressing default rules and forms of closed-world reasoning.

1.4 Integrating DLs and rules: Main issues

Many semantic and computational problems have emerged in the combination of DLs and rule-based representation formalisms. Among them, we concentrate on the following main issues/goals:

(1) *OWA vs. CWA:* DLs are fragments of first-order logic (FOL), hence their semantics is based on the *Open World Assumption* (OWA) of classical logic, while rules are based on a *Closed World Assumption* (CWA), imposed by the different semantics for logic programming and deductive databases (which formalize various notions of information closure). How to integrate the OWA of DLs and the CWA of rules in a "proper" way? i.e., how to merge monotonic and nonmonotonic logical subsystems from a semantic viewpoint?

(2) *UNA vs. non-UNA:* some DLs, in particular the ones specifically tailored for the Semantic Web, i.e., OWL and OWL-DL, are not based on the *Unique Name Assumption* (UNA) (we recall that the UNA imposes that different terms denote different objects). On the other hand, the standard semantics of Datalog rules is based on the UNA (see e.g. [12] for a discussion on this semantic discrepancy). How to define a non-UNA-based semantics for DLs and rules? and most importantly, is it possible to reason under the non-UNA-based semantics by exploiting standard (i.e., UNA-based) Datalog engines?

(3) *decidability preservation:* as shown by the first studies in this field [28], decidability (and complexity) of reasoning is a crucial issue in systems combining DL knowledge bases and Datalog rules. In fact, in general this combination does not preserve decidability, i.e., starting from a DL knowledge base in which reasoning is decidable and a set of rules in which reasoning is decidable, reasoning in the knowledge base obtained by integrating these two components may not be a decidable problem.

(4) *modularity of reasoning:* can reasoning in DL knowledge bases augmented with rules be performed in a modular way, strongly separating reasoning about the DL component and reasoning about the rule component? This is a very desirable property, since it allows for defining reasoning techniques (and engines) on top of deductive methods (and implemented systems) developed separately for DLs [6] and for Datalog and its nonmonotonic extensions [16].

1.5 Structure of the paper

The paper is structured in the following way. We start by briefly introducing Description Logics in Section 2, and Datalog and its nonmonotonic extensions in Section 3. Then, in Section 4 we analyze the main issues that arise when integrating Description Logics and rules. In Section 5 we review the main approaches to the integration of ontologies and Datalog rules. Then, in Section 6 we present $\mathcal{DL}$*+log*, one of the most powerful formalisms integrating Descritpion Logics and Datalog rules: in particular, we show how $\mathcal{DL}$*+log* deals with the main issues previously discussed. Finally, in Section 7 we briefly illustate some of the main open problems towards the integration of Description Logics and Datalog rules.

2 Description Logics

We start by introducing Description Logics. For a more detailed introduction to this topic, we refer the reader to [6].

Description Logics (DLs) are logics that represent the domain of interest in terms of *concepts*, denoting sets of objects, and *roles*, denoting binary relations between (instances of) concepts. Complex concept and role expressions are constructed starting from a set of atomic concepts and roles by applying suitable constructs.

Different DLs allow for different constructs. Properties of concepts and roles are specified through inclusion assertions, stating that every instance of a concept (respectively, role) is also an instance of another concept (respectively, role).

As an example of a DL, in the following we formally introduce $\mathcal{ALC}$, which actually constitutes a subset of the DLs of the OWL family defined as ontology languages.

2.1 Syntax

In $\mathcal{ALC}$, concepts and roles are formed according to the following syntax:

$$C ::= \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid \exists P.C \mid \forall P.C$$

where A denotes an atomic concept, P denotes an atomic role, and C_1, C_2 denote general concept expressions.

A DL *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ represents the domain of interest in terms of two components, a *TBox* $\mathcal{T}$, specifying the intensional knowledge, and an *ABox* $\mathcal{A}$, specifying extensional knowledge.

A TBox is formed by a set of *inclusion assertions* of the form

$$C_1 \sqsubseteq C_2$$

where C_1 and C_2 are general concepts. As we said before, such an inclusion assertion expresses that all instances of concept C_1 are also instances of concept C_2.

An ABox is formed by a set of *membership assertions* on atomic concepts and on atomic roles, of the form

$$C(a), \qquad P(a, b)$$

stating respectively that the object denoted by the constant a is an instance of the concept C and that the pair of objects denoted by the pair of constants (a, b) is an instance of the role P.

2.2 Semantics

The semantics of a DL is given in terms of standard first-order interpretations. Formally, a *DL-interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of an interpretation domain $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ defined as follows. First, $\mathcal{I}$ assigns to each atomic concept A a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and to each role P a binary relation $P^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\
P^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}
\end{aligned}$$

Based on the above interpretation of atomic predicates, $\mathcal{I}$ assigns a subset of $\Delta^{\mathcal{I}}$ to general concept expression. For the constructs of $\mathcal{ALC}$, the interpretation of general concepts is defined inductively as follows:

$$\begin{aligned}
\neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
C_1 \sqcap C_2^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
C_1 \sqcup C_2^{\mathcal{I}} &= C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \\
\exists P.C^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \exists d'.(d, d') \in P^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\} \\
\forall P.C^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \forall d'.(d, d') \in P^{\mathcal{I}} \text{ implies } d' \in C^{\mathcal{I}}\}
\end{aligned}$$

A concept C is *satisfiable* if there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$, otherwise C is unsatisfiable. An interpretation $\mathcal{I}$ is a *model* of a concept C if $\mathcal{I}$ satisfies C.

A DL-interpretation $\mathcal{I}$ is a *model* of an inclusion assertion $C_1 \sqsubseteq C_2$, if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.

To specify the semantics of membership assertions, we extend the interpretation function to constants, by assigning to each constant a an object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.[1] A DL-interpretation $\mathcal{I}$ is a model of a membership assertion $C(a)$, (resp., $P(a, b)$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp., $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$).

Given an (inclusion, or membership) assertion α, and a DL-interpretation $\mathcal{I}$, we denote by $\mathcal{I} \models \alpha$ the fact that $\mathcal{I}$ is a model of α. A *model of a KB* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is a DL-interpretation $\mathcal{I}$ that is a model of all assertions in $\mathcal{T}$ and $\mathcal{A}$. A KB is *satisfiable* if it has at least one model. A KB $\mathcal{K}$ *entails* an assertion α, written $\mathcal{K} \models \alpha$, if all models of $\mathcal{K}$ are also models of α. Analogously, a TBox $\mathcal{T}$ entails an assertion α, written $\mathcal{T} \models \alpha$, if all models of $\mathcal{T}$ are also models of α.

Observe that $\mathcal{ALC}$ (and, in practice, every DL) is actually a fragment of function-free first-order logic, with a special syntax which avoids the explicit use of variable symbols. In fact, it is immediate to verify that a DL knowledge base $\mathcal{K}$ is semantically equivalent to a FOL theory $FO(\mathcal{K})$ in which each assertion in the knowledge base is expressed by a first-order sentence (for details on such a translation see [6]). For instance, the TBox inclusion assertion

$$A_1 \sqcap \exists P_1.A_2 \sqsubseteq \forall P_2.A_3 \sqcup \neg A_4$$

is equivalent to the first-order sentence

$$\forall x.A_1(x) \wedge (\exists y.P_1(x, y) \wedge A_2(y)) \rightarrow (\forall z.P_2(x, z) \rightarrow A_3(z)) \vee \neg A_4(x)$$

Finally, we remark that, due to the above FOL semantics, DLs are interpreted over an unbound (possibly infinite) domain. Moreover, unique names are not always assumed[2].

3 Disjunctive Datalog

In this section be briefy recall disjunctive Datalog [14], denoted by Datalog$^{\neg\vee}$, which is the well-known nonmonotonic extension of Datalog with negation as failure and disjunction.

3.1 Syntax

We start from a predicate alphabet, a constant alphabet, and a variable alphabet. An *atom* is an expression of the form $p(X)$, where p is a predicate of arity n

[1] We recall that, if we enforce the unique name assumption on constants, then the interpretation $a^{\mathcal{I}}$ of each constant a must be such that, for each constant b different from a, $b^{\mathcal{I}} \neq a^{\mathcal{I}}$ [6].

[2] Even though some DLs are based on the UNA, the most expressive ones, like the ones in the OWL family, are not.

and X is a n-tuple of variables and constants. If no variable symbol occurs in X, then $p(X)$ is called a *ground atom* (or *fact*).

A Datalog$^{\neg\vee}$ rule R is an expression of the form

$$p_1(X_1) \vee \ldots \vee p_n(X_n) \leftarrow r_1(Y_1), \ldots, r_m(Y_m), not\, s_1(W_1), \ldots, not\, s_k(W_k)$$

such that $n \geq 0$, $m \geq 0$, $k \geq 0$, each $p_i(X_i)$, $r_i(Y_i)$, $s_i(W_i)$ is an atom and every variable occurring in R must appear in at least one of the atoms $r_1(Y_1), \ldots, r_m(Y_m)$. This last condition is known as the *Datalog safeness* condition for variables. The variables occurring in the atoms $p_1(X_1), \ldots, p_n(X_n)$ are called the *head variables* of R. If $n = 0$, we call R a *constraint*.

A a *Datalog*$^{\neg\vee}$ program is a set of Datalog$^{\neg\vee}$ rules. If, for all $R \in \mathcal{P}$, $n \leq 1$, $\mathcal{P}$ is called a *Datalog*$^{\neg}$ program. If, for all $R \in \mathcal{P}$, $k = 0$, $\mathcal{P}$ is called a *positive disjunctive Datalog* program. If, for all $R \in \mathcal{P}$, $n \leq 1$ and $k = 0$, $\mathcal{P}$ is called a *positive Datalog* program. If there are no occurrences of variable symbols in a rule R, then R is called a *ground* rule. A *ground* program is a program containing only ground rules.

3.2 Semantics

The semantics of disjunctive Datalog is given in terms of *stable models* of a program $\mathcal{P}$, which we recall below.

The *ground instantiation of* $\mathcal{P}$, denoted by $G(P)$, is the program obtained from $\mathcal{P}$ by replacing every rule R in $\mathcal{P}$ with the set of ground rules obtained by applying all possible substitutions of variables in R with constants occurring in $\mathcal{P}$ (such a set of constants is called the Herbrand universe of $\mathcal{P}$).

We denote by $HB(\mathcal{P})$ the *Herbrand base* of $\mathcal{P}$, i.e. the set of all ground instantiations of predicates occurring in $\mathcal{P}$ over the Herbrand universe of $\mathcal{P}$.

A *Datalog interpretation* I of $\mathcal{P}$ is a subset of $HB(\mathcal{P})$. I satisfies a positive ground rule

$$p_1 \vee \ldots \vee p_n \leftarrow r_1, \ldots, r_m \tag{1}$$

if the following condition holds: if each atom in $\{r_1, \ldots, r_m\}$ belongs to I, then at least one atom p_i belongs to I.

I is a *model* of $\mathcal{P}$ if I satisfies each rule in $G(\mathcal{P})$. A model of $\mathcal{P}$ is *minimal* if it does not properly contain any other model of $\mathcal{P}$.

Given a Datalog interpretation $I \subseteq HB(\mathcal{P})$, the *GL-reduct* of $\mathcal{P}$ with respect to I (denoted as $GL(\mathcal{P}, I)$) is the program obtained from $G(\mathcal{P})$ by removing all clauses of the form (1) such that there exists $s_j \in I$ for some $j \in \{1, \ldots, k\}$, and by removing all negated predicates of the form $not\ s_i$ from the remaining clauses.

A Datalog interpretation $I \subseteq HB(\mathcal{P})$ is a *stable model* of $\mathcal{P}$ if I is a minimal model of $GL(G(\mathcal{P}), I)$.

We say that a program $\mathcal{P}$ entails a ground query (i.e., a ground literal predicate) $q(\overline{a})$, denoted as $\mathcal{P} \models q(\overline{a})$, if $q(\overline{a})$ belongs to all stable models of $\mathcal{P}$.

We remark that, based on the above semantics, every disjunctive Datalog program is interpreted over a finite domain, which coincides with the set of constants occurring in the program. Moreover, every Datalog interpretation enforces the unique name assumption (different constants are interpreted as different objects).

4 Integrating DLs and rules: Main issues

In this section we address the main issues arising when trying to combine DLs and (disjunctive) Datalog in a single formalism.

Syntax From the syntactic viewpoint, integrating a DL with (disjunctive) Datalog simply means the possibility of writing a "hybrid" knowledge base containing a TBox, an ABox, and a set of Datalog rules.

Semantics From the semantic viewpoint, the meaning of such an integrated knowledge base can be provided in two ways:

1. the whole knowledge base is considered as a first-order theory, by interpreting Datalog rules as first-order implications. More specifically, let R be the following Datalog$^{\neg\vee}$ rule:

$$\begin{aligned} p_1(X_1, c_1) \vee \ldots \vee p_n(X_n, c_n) \leftarrow & \\ & r_1(Y_1, d_1), \ldots, r_m(Y_m, d_m), \\ & s_1(Z_1, e_1), \ldots, s_k(Z_k, e_k), \\ & \mathit{not}\ u_1(W_1, f_1), \ldots, \mathit{not}\ u_h(W_h, f_h) \end{aligned}$$

   where each X_i, Y_i, Z_i, W_i is a set of variables and each c_i, d_i, e_i, f_i is a set of constants. Then, we denote by $FO(R)$ the first-order sentence

$$\begin{aligned} \forall \overline{x}_1, \ldots, \overline{x}_n, \overline{y}_1, \ldots, \overline{y}_m, \overline{z}_1, \ldots, \overline{z}_k, \overline{w}_1, \ldots, \overline{w}_h. & \\ r_1(\overline{y}_1, d_1) \wedge \ldots \wedge r_m(\overline{y}_m, d_m) \wedge & \\ s_1(\overline{z}_1, e_1) \wedge \ldots \wedge s_k(\overline{z}_k, e_k) \wedge & \\ \neg u_1(\overline{w}_1, f_1) \wedge \ldots \wedge \neg u_h(\overline{w}_h, f_h) \rightarrow\ & p_1(\overline{x}_1, c_1) \vee \ldots \vee p_n(\overline{x}_n, c_n) \end{aligned}$$

   and, given a Datalog$^{\neg\vee}$ program $\mathcal{P}$, we denote by $FO(\mathcal{P})$ the set of first-order sentences $\{FO(R) \mid R \in \mathcal{P}\}$.
   Finally, the semantics of a knowledge base $(\mathcal{K}, \mathcal{P})$ composed of a DL-KB $\mathcal{K}$ and a Datalog program $\mathcal{P}$ is given by the first-order theory corresponding to the union of $FO(\mathcal{P})$ and the first-order translation $FO(\mathcal{K})$ of $\mathcal{K}$.
   While the above semantic account has the advantage of being clear and easy to define, it has the drawback of not being conservative with respect to the semantics of Datalog rules. In other words, the meaning of a Datalog program $\mathcal{P}$ in the new semantics is different from its meaning according to the standard Datalog semantics (the CWA of Datalog is missing in the new semantics).

2. the semantics is defined in a way such that it is a "conservative extension" of both the DL and Datalog. However, this is not as immediate as the above semantic account, due to the different semantic nature of the two formalisms: in fact, one has to simultaneously deal with two semantic discrepancies: the OWA of DLs and the CWA of Datalog on the one side, and the UNA of Datalog and the absence of the UNA of (some) DLs on the other side. In Section 6 we will define such a semantics.

Reasoning From the reasoning perspective, an important aspect is the "degree of integration" of the two components (the DL-KB and the Datalog program). Indeed, as we will explain in Section 5, the complexity of reasoning in systems combining DLs and rules is directly related to such a degree of integration. In particular, it is well-known that the "full" interaction between a DL-KB and a Datalog program leads to undecidability of reasoning under the above presented FOL semantics, even for extremely simple DLs [28]. On the other side of the spectrum, rules may not interact at all with the DL-KB, and of course this kind of (uninteresting) integration is not problematic at all with respect to the reasoning task, since the two components can be processed separately by standard (DL and Datalog) reasoners.

Obviously, in order to represent some kind of significant interaction, the DL KB and the rules have to share some predicate symbols. A measure of the degree of interaction between the two components depends on these shared predicates, and on how they can be used within DL statements and rules.

More specifically, the alphabet of predicates is divided into *DL predicates* and *Datalog predicates*, where Datalog predicates are the ones that do not occur in the DL-KB, while DL predicates may occur both in the DL-KB and in rules. Then:

- the full interaction does not make any assumption on the form of rules based on the above classification of predicates;
- the loose interaction imposes some limitations on the use of DL predicates in rules.

For instance, as we will illustrate in Section 5, a common approach to the loose integration of DLs and rules is realized through the so-called *DL-safeness* condition for Datalog rules. This is a syntactic condition that can be expressed as follows: *every variable occurring in an atom with a DL predicate must occur in a atom with a Datalog predicate in the body of the rule.* Such a condition is sufficient to allow for a nice computational behaviour of reasoning, but has the drawback of restricting the expressiveness of the combined language thus defined. E.g., DL-safe rules are not able to express arbitrary *conjunctive queries* to the DL-KB. Conjunctive queries correspond to a simple form of non-recursive Datalog rules, are computable in many DLs, and there are known algorithms for conjunctive query algorithms in many DLs [9, 32]. Therefore, DL-safeness seems to imply a too severe limitation in the expressiveness of rules.

Finally, another measure of the degree of integration lies in the direction of the *information flow* between DL-KB and rules, which may be either bidirectional (from the DL-KB to the rules and vice versa), or unidirectional (only from the DL-KB to the rules). In the latter case, the presence of rules does not affect the semantics of DL predicates. Often, the restriction to the unidirectional flow is realized through the syntactic restriction that DL predicates may not occur in the head of rules (they can only occur in the body of rules).

We conclude this section with two examples of knowledge bases combining DLs and rules.

$PERSON \sqsubseteq \exists FATHER^{-}.MALE$
$MALE \sqsubseteq PERSON$
$FEMALE \sqsubseteq PERSON$
$FEMALE \sqsubseteq \neg MALE$
$MALE(Bob)$
$PERSON(Mary)$
$PERSON(Paul)$

(a) DL-KB $\mathcal{K}$ (ontology about persons)

$boy(X) \leftarrow enrolled(X, c1), PERSON(X), not\ girl(X)$ $[R1]$
$girl(X) \leftarrow enrolled(X, c2), PERSON(X)$ $[R2]$
$boy(X) \vee girl(X) \leftarrow enrolled(X, c3), PERSON(X)$ $[R3]$
$FEMALE(X) \leftarrow girl(X)$ $[R4]$
$MALE(X) \leftarrow boy(X)$ $[R5]$
$enrolled(Paul, c1)$
$enrolled(Mary, c1)$
$enrolled(Mary, c2)$
$enrolled(Bob, c3)$

(b) disjunctive Datalog program $\mathcal{P}$ (rules about students)

Fig. 1. Knowledge base $\mathcal{B} = (\mathcal{K}, \mathcal{P})$ of Example 1

Example 1. Let $\mathcal{B} = (\mathcal{K}, \mathcal{P})$ be the knowledge base reported in Figure 1, where the DL-KB $\mathcal{K}$ defines an ontology about persons, and the disjunctive Datalog program $\mathcal{P}$ defines nonmonotonic rules about students. For the sake of readability, we denote DL predicates by uppercase names, and denote Datalog predicates by lowercase names.

It is immediate to verify that $\mathcal{B}$ satisfies the DL-safe condition described above. ∎

$RICH \sqcap UNMARRIED \sqsubseteq \exists WANTS\text{-}TO\text{-}MARRY^{-}.\top$
$UNMARRIED(Mary)$
$UNMARRIED(Joe)$

(a) DL-KB $\mathcal{K}$

$happy(X) \leftarrow famous(X), WANTS\text{-}TO\text{-}MARRY(Y, X)$ [R1]
$RICH(X) \leftarrow famous(X), not\ scientist(X)$ [R2]
$famous(Mary)$
$famous(Paul)$
$famous(Joe)$
$scientist(Joe)$

(b) disjunctive Datalog program $\mathcal{P}$

Fig. 2. Knowledge base $\mathcal{B} = (\mathcal{K}, \mathcal{P})$ of Example 2

Example 2. Let $\mathcal{B} = (\mathcal{K}, \mathcal{P})$ be the knowledge base reported in Figure 2.

Again, DL predicates are denoted by uppercase names, while Datalog predicates are denoted by lowercase names. In this case, the rules in $\mathcal{P}$ (in particular, rule R1) do *not* satisfy the DL-safeness condition. ∎

5 A brief state of the art

In this section we briefly survey recent work in integrating ontologies and rules.[3] We divide such studies in two main streams: (i) approaches dealing with forms of DL-safe (and, more generally, loose) interaction between DL-KBs and rules; (ii) approaches concerning forms of "non-DL-safe" (or tight) interaction.

5.1 Loose integration

The first formal proposal for the integration of Description Logics and rules is $\mathcal{AL}$-log [13]. $\mathcal{AL}$-log is a framework which integrates KBs expressed in the description logic $\mathcal{ALC}$ and positive Datalog programs. Then, *disjunctive* $\mathcal{AL}$*-log* was proposed in [35] as an extension of $\mathcal{AL}$-log, based on the use of Datalog$^{\neg\vee}$ instead of positive Datalog, and on the possibility of using binary predicates (roles) besides unary predicates (concepts) in rules. Such approaches realize a form of loose integration between DLs and Datalog that precisely corresponds to the DL-safeness condition described in the previous section. Moreover, both in $\mathcal{AL}$-log and in disjunctive $\mathcal{AL}$-log DL predicates can occur only in the bodies of rules, which forces the information flow to be unidirectional.

[3] For other surveys on this topic see, e.g., [5, 15].

The framework of $\mathcal{AL}$-log has been extended in a different way in [30]. There, the problem of extending OWL-DL with positive Datalog programs is analyzed. Again, the interaction between OWL-DL and rules is restricted through the DL-safeness condition. With respect to disjunctive $\mathcal{AL}$-log, in [30] a more expressive DL and a less expressive rule language (interpreted under first-order semantics) are adopted: moreover, the information flow is bidirectional, i.e., DL predicates may appear in the head of rules.

All the above approaches based on DL-safeness have been generalized in [36] to the integration of arbitrary, decidable, first-order theories and disjunctive Datalog rules. This paper establishes an important computational result, which states that the DL-safe based integration preserves (under very general conditions) decidability of reasoning.

The work presented in [21] can also be seen as an approach based on a form of safe interaction between the DL-KB and the rules: in particular, a rule language is defined such that it is possible to encode a set of rules into a semantically equivalent DL-KB. As a consequence, such a rule language is very restricted.

A different approach is presented in [23, 22], which proposes Conceptual Logic Programming (CLP), an extension of answer set programming (i.e., Datalog$^{\neg\vee}$) towards infinite domains. In order to keep reasoning decidable, a syntactic restriction on CLP program rules is imposed. This approach is related to integrating DLs and rules, since the authors also show that CLPs can embed expressive DL-KBs, which in turn implies decidability of adding CLP rules to such DLs. However, the syntactic restriction on CLP rules, whose purpose is to impose a "forest-like" structure to the models of the program, is different from the safeness conditions analyzed so far, which makes it impossible to compare this approach with the studies previously mentioned.

Another approach for extending DLs with Datalog$^{\neg}$ rules is presented in [17, 18]. Differently from the other approaches above described, this proposal allows for specifying, in rule bodies, *queries* to the DL component, where every query also allows for specifying an input from the rule component, and thus for an information flow from the rule component to the DL-KB. The meaning of such queries in rule bodies is given at the meta-level, through the notion of skeptical entailment in the DL-KB. Thus, from the semantic viewpoint, this form of interaction-via-entailment between the two components is more restricted than in the approaches previously mentioned; on the other hand, such an increased separation in principle allows for more modular reasoning methods, which are able to completely separate reasoning about the DL-KB and reasoning about the Datalog program. For a more detailed description of this approach see [15].

Finally, [3, 2, 4] present approaches for the combination of defeasible reasoning with Description Logics, under a safe interaction-via-entailment scheme which is semantically analogous to the one proposed in [17]. Besides the differences with the studies on nonmonotonic extensions of DL-KBs previously mentioned due to the semantics of nonmonotonic rules, a main characteristic of these proposals consists in the fact the information flow is unidirectional, i.e., it only goes from the DL-KB to the rules.

5.2 Tight integration

Research in non-safe interaction of DLs and rules actually started with the work on CARIN [26–28], which established very important decidability and undecidability results concerning the integration of DL-KBs and rules. Roughly speaking, such results clearly indicate that, in case of unrestricted interaction between a DL-KB and a set of rules, decidability of reasoning holds only if at least one of the two components has very limited expressive power: e.g., in order to retain decidability of reasoning, allowing recursion in rules imposes very severe restrictions on the expressiveness of DL-KB.

Then, we remark that query answering over a knowledge base can be seen as a problem of reasoning in a DL-KB augmented with rules which encode the query. In this respect, an important undecidability result concerning query answering over databases with integrity constraints is reported in [10]. More precisely, it is shown that answering recursive Datalog queries over a database with simple integrity constraints (keys and foreign keys), interpreted as a knowledge base, i.e., under an open-world assumption, is undecidable. This setting also can be viewed as a DL-KB with non-DL-safe interaction between a knowledge base (database with integrity constraints) and a rule component (the query).

As already observed, it is difficult to provide a good semantic account for non-safe interaction between DL-KBs and nonmonotonic rules, due to the classical, open-world semantics of DL-KBs, and the closed-world assumption underlying nonmonotonic systems. For instance, [29] illustrates the problems in providing a semantic account for non-safe interaction of ontologies and Datalog$^{\neg\vee}$ programs.

Finally, another recent proposals in this field is SWRL [24], a non-safe approach to the integration of rules and DL-KBs in which rules are interpreted under the classical FOL semantics. The addition of this kind of rules to DLs leads to undecidability of reasoning.

5.3 Loose vs. tight integration

Summarizing, what emerges from the studies in the integration of DL-KBs and rules is that while, on the one hand, a safe form of interaction between DLs and rules generally allows for decidable reasoning and nice computational properties, on the other hand, the results concerning non-safe interaction indicate that a tight connection between the two components can only be obtained at the price of severely restricting the expressive power of either the DL-KB or the rules.

In the next section we present in detail $\mathcal{DL}$*+log*, which is currently one of the most expressive and decidable combinations of Description Logics and disjunctive Datalog. $\mathcal{DL}$*+log* overcomes the DL-safeness condition to obtain a tighter form of interaction between DLs and rules.

6 The $\mathcal{DL}$*+log* approach

In this section we introduce $\mathcal{DL}$*+log* (we refer to [38] for more details).

6.1 Syntax

We start from three mutually disjoint predicate alphabets:

- an alphabet of concept names Σ_C;
- an alphabet of role names Σ_R;
- an alphabet of Datalog predicates Σ_D.

We call a predicate p a *DL predicate* if either $p \in \Sigma_C$ or $p \in \Sigma_R$.[4] Then, we denote by $\mathcal{C}$ a countably infinite alphabet of constant names.

An *atom* is an expression of the form $p(X)$, where p is a predicate of arity n and X is a n-tuple of variables and constants.[5] If no variable symbol occurs in X, then $p(X)$ is called a *ground atom* (or *fact*). If $p \in \Sigma_C \cup \Sigma_R$, the atom is called a *DL-atom*, while if $p \in \Sigma_D$, it is called a *Datalog atom*.

To define a $\mathcal{DL}$*+log* knowledge base, we can start from any description logic $\mathcal{DL}$: in other words, the construction defined in the following is parametric with respect to the description logic used to express the DL-KB.

Definition 1. *Given a description logic $\mathcal{DL}$, a $\mathcal{DL}$+log-*knowledge base $\mathcal{B}$ *is a pair $(\mathcal{K}, \mathcal{P})$, where:*

- *$\mathcal{K}$ is a $\mathcal{DL}$-KB, i.e., a pair $(\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is the TBox and $\mathcal{A}$ is the ABox;*
- *$\mathcal{P}$ is a set of Datalog$^{\neg\vee}$ rules, where each rule R has the form*

$$p_1(X_1) \vee \ldots \vee p_n(X_n) \leftarrow r_1(Y_1), \ldots, r_m(Y_m), s_1(Z_1), \ldots, s_k(Z_k), \\ not\, u_1(W_1), \ldots, not\, u_h(W_h)$$

such that $n \geq 0$, $m \geq 0$, $k \geq 0$, $h \geq 0$, each $p_i(X_i)$, $r_i(Y_i)$, $s_i(Z_i)$, $u_i(W_i)$ is an atom and:

- *each p_i is either a DL predicate or a Datalog predicate;*
- *each r_i, u_i is a Datalog predicate;*
- *each s_i is a DL predicate;*
- *(Datalog safeness) every variable occurring in R must appear in at least one of the atoms $r_1(Y_1), \ldots, r_m(Y_m), s_1(Z_1), \ldots, s_k(Z_k)$;*
- (weak safeness) *every head variable of R must appear in at least one of the atoms $r_1(Y_1), \ldots, r_m(Y_m)$.*

We remark that the above notion of weak safeness allows for the presence of variables that only occur in DL-atoms in the body of R. On the other hand, the notion of *DL-safeness* of variables adopted in previous approaches [35, 31, 36] can be expressed as follows: *every variable of R must appear in at least one of the atoms $r_1(Y_1), \ldots, r_m(Y_m)$*. Therefore, DL-safeness forces every variable of R to occur also in the Datalog atoms in the body of R, while weak safeness allows for the presence of variables that only occur in DL-atoms in the body of R.

Without loss of generality, in the rest of the paper we assume that in a $\mathcal{DL}$*+log*-KB $(\mathcal{K}, \mathcal{P})$ all constants occurring in $\mathcal{K}$ also occur in $\mathcal{P}$.

[4] For DLs which allow for using equality, we assume that the equality predicate is a DL predicate.

[5] As usual, atoms involving equalities are written using the infix notation $t_1 = t_2$.

6.2 Semantics

We now define a semantics for $\mathcal{DL}$+*log*-KBs which is a "conservative extension" of both the open-world semantics of DLs and the closed-world semantics of disjunctive Datalog.

Given an interpretation $\mathcal{I}$ and a predicate alphabet Σ, we denote by $\mathcal{I}_\Sigma$ the projection of $\mathcal{I}$ to Σ, i.e., $\mathcal{I}_\Sigma$ is obtained from $\mathcal{I}$ by restricting it to the interpretation of the predicates in Σ.

Given a set of constants $\mathcal{C}$, the *ground instantiation of* $\mathcal{P}$ *with respect to* $\mathcal{C}$, denoted by $gr(\mathcal{P},\mathcal{C})$, is the program obtained from $\mathcal{P}$ by replacing every rule R in $\mathcal{P}$ with the set of rules obtained by applying all possible substitutions of variables in R with constants in $\mathcal{C}$.

Given an interpretation $\mathcal{I}$ of an alphabet of predicates $\Sigma' \subset \Sigma$, and a ground program $\mathcal{P}_g$ over the predicates in Σ, the *projection of* $\mathcal{P}_g$ *with respect to* $\mathcal{I}$, denoted by $\Pi(\mathcal{P}_g,\mathcal{I})$, is the ground program obtained from $\mathcal{P}_g$ as follows. For each rule $R \in \mathcal{P}_g$:

- delete R if there exists an atom $r(t)$ in the head of R such that $r \in \Sigma'$ and $t^\mathcal{I} \in r^\mathcal{I}$;
- delete each atom $r(t)$ in the head of R such that $r \in \Sigma'$ and $t^\mathcal{I} \notin r^\mathcal{I}$;
- delete R if there exists an atom $r(t)$ in the body of R such that $r \in \Sigma'$ and $t^\mathcal{I} \notin r^\mathcal{I}$;
- delete each atom $r(t)$ in the body of R such that $r \in \Sigma'$ and $t^\mathcal{I} \in r^\mathcal{I}$;

Informally, the projection of $\mathcal{P}_g$ with respect to $\mathcal{I}$ corresponds to evaluating $\mathcal{P}_g$ with respect to $\mathcal{I}$, thus eliminating from $\mathcal{P}_g$ every atom whose predicate is interpreted in $\mathcal{I}$. Thus, when $\Sigma' = \Sigma_C \cup \Sigma_R$, all occurrences of DL predicates are eliminated in the projection of $\mathcal{P}_g$ with respect to $\mathcal{I}$, according to the evaluation in $\mathcal{I}$ of the atoms with DL predicates occurring in $\mathcal{P}_g$.

Then, we introduce the notions of minimal model and stable model for Datalog$^{\neg\vee}$ in the absence of the UNA.[6]

Given two interpretations $\mathcal{I}_1$, $\mathcal{I}_2$ of the set of predicates Σ and the set of constants $\mathcal{C}$, we write $\mathcal{I}_1 \subset_{\Sigma,\mathcal{C}} \mathcal{I}_2$ if (i) for each $p \in \Sigma$ and for each tuple t of constants from $\mathcal{C}$, if $t^{\mathcal{I}_1} \in p^{\mathcal{I}_1}$ then $t^{\mathcal{I}_2} \in p^{\mathcal{I}_2}$, and (ii) there exist $p \in \Sigma$ and tuple t of constants from $\mathcal{C}$ such that $t^{\mathcal{I}_1} \notin p^{\mathcal{I}_1}$ and $t^{\mathcal{I}_2} \in p^{\mathcal{I}_2}$.

Given a positive ground Datalog$^{\neg\vee}$ program $\mathcal{P}$ over an alphabet of predicates Σ and an interpretation $\mathcal{I}$, we say that $\mathcal{I}$ is a *minimal model* of $\mathcal{P}$ if: (i) $\mathcal{I}$ satisfies the first-order translation $FO(\mathcal{P})$ of $\mathcal{P}$ (defined in Section 4); (ii) there is no interpretation $\mathcal{I}'$ such that $\mathcal{I}'$ satisfies $FO(\mathcal{P})$ and $\mathcal{I}' \subset_{\Sigma,\mathcal{C}} \mathcal{I}$.

Given a ground Datalog$^{\neg\vee}$ program $\mathcal{P}$ and an interpretation $\mathcal{I}$ for $\mathcal{P}$, the *GL-reduct* [19] of $\mathcal{P}$ with respect to $\mathcal{I}$, denoted by $GL(\mathcal{P},\mathcal{I})$, is the positive ground program obtained from $\mathcal{P}$ as follows. For each rule $R \in \mathcal{P}$:

[6] Observe that the notions of minimal model and stable model presented here slightly differs from the standard ones for Datalog$^{\neg\vee}$ presented in Section 3, since they are expressed in a more general framework in which unique names are not assumed. Consequently, the interpretation of constants must be considered in the definition of minimal and stable model.

1. delete R if there exists a negated atom $not\, r(t)$ in the body of R such that $t^{\mathcal{I}} \in r^{\mathcal{I}}$;
2. delete each negated atom $not\, r(t)$ in the body of R such that $t^{\mathcal{I}} \notin r^{\mathcal{I}}$.

Given a ground Datalog$^{\neg\vee}$ program $\mathcal{P}$ and an interpretation $\mathcal{I}$, $\mathcal{I}$ is a *stable model* for $\mathcal{P}$ iff $\mathcal{I}$ is a minimal model of $GL(\mathcal{P}, \mathcal{I})$.

Definition 2. *An interpretation $\mathcal{I}$ of $\Sigma_C \cup \Sigma_R \cup \Sigma_D$ is a model for $\mathcal{B} = (\mathcal{K}, \mathcal{P})$ if the following conditions hold:*

1. *$\mathcal{I}_{\Sigma_C \cup \Sigma_R}$ satisfies $\mathcal{K}$;*
2. *$\mathcal{I}_{\Sigma_D}$ is a stable model for $\Pi(gr(\mathcal{P}, \mathcal{C}), \mathcal{I}_{\Sigma_C \cup \Sigma_R})$.*

$\mathcal{B}$ is called satisfiable *if $\mathcal{B}$ has at least a model.*

We say that a ground atom $p(c)$ is entailed by $\mathcal{B}$ iff, for each model $\mathcal{I}$ of $\mathcal{B}$, $\mathcal{I}$ satisfies $p(c)$.

According to the above semantics, DL predicates are interpreted under the open-world assumption, while Datalog predicates are interpreted under the closed-world assumption of disjunctive Datalog (see [37] for a detailed discussion of this aspect).

Notice that, under the above semantics, entailment can be reduced to satisfiability, since it is possible to express constraints in the Datalog program. More precisely, it is immediate to verify that $(\mathcal{K}, \mathcal{P})$ entails $p(c)$ iff $(\mathcal{K}, \mathcal{P} \cup \{\leftarrow p(c)\})$ is unsatisfiable. In a similar way, it can be seen that *conjunctive query answering* can be reduced to satisfiability in $\mathcal{DL}$*+log* (see [38]). Consequently, in the following we concentrate on the satisfiability problem in $\mathcal{DL}$*+log*-KBs.

Example 1.(contd.) Let us consider again the knowledge base $\mathcal{B} = (\mathcal{K}, \mathcal{P})$ reported in Figure 1, where the DL-KB $\mathcal{K}$ defines an ontology about persons, and the disjunctive Datalog program $\mathcal{P}$ defines nonmonotonic rules about students.

First, since all rules in $\mathcal{P}$ are DL-safe, the rules in $\mathcal{P}$ also satisfy the weak safeness condition of Definition 1: consequently, $\mathcal{B}$ is a $\mathcal{DL}$*+log*-KB.

Then, it can be easily verified that all models for $\mathcal{B}$ satisfy the following ground atoms:

- *boy*(*Paul*) (since rule R1 is always applicable for $X = Paul$ and R1 acts like a *default rule*, which can be read as follows: if X is a person enrolled in course $c1$, then X is a boy, unless we know for sure that X is a girl);
- *girl*(*Mary*) (since rule R2 is always applicable for $X = Mary$)
- *boy*(*Bob*) (since rule R3 is always applicable for $X = Bob$, and, by rule R4, the conclusion *girl*(*Bob*) is inconsistent with $\mathcal{K}$);
- *MALE*(*Paul*) (due to rule R5);
- *FEMALE*(*Mary*) (due to rule R4).

Notice that $\mathcal{B} \models FEMALE(Mary)$, while $\mathcal{K} \not\models FEMALE(Mary)$. In other words, adding rules has indeed an effect on the conclusions one can draw about DL predicates. ∎

Example 2.(contd.) Let us consider again the knowledge base $\mathcal{B} = (\mathcal{K}, \mathcal{P})$ reported in Figure 2.

First, observe that $\mathcal{B}$ is a $\mathcal{DL}$*+log*-KB: in particular, the variable Y in rule R1 is weakly-safe according to Definition 1 (we also recall that rule R1 is *not* DL-safe, since Y does not occur in any Datalog predicate in rule R1).

Then, it can be easily verified that all models for $\mathcal{B}$ satisfy the following formulas:

- *RICH*(*Paul*) and *RICH*(*Mary*), since the default rule R2 is always applicable for $X = Paul$ and $X = Mary$, but not for $X = Joe$, since the fact *scientist*(*Joe*) holds in every model for $\mathcal{B}$;
- $\exists WANTS\text{-}TO\text{-}MARRY^{-}.\top(Mary)$, due to the first axiom of the DL-KB and to the fact that both *RICH*(*Mary*) and *UNMARRIED*(*Mary*) hold in every model of the $\mathcal{DL}$*+log*-KB $\mathcal{B}$ (while $\exists WANTS\text{-}TO\text{-}MARRY^{-}.\top(Paul)$ is not forced by such axiom to hold in every model of $\mathcal{B}$, because *UNMARRIED*(*Paul*) is not forced to hold in every such model);
- *happy*(*Mary*), due to the above conclusions and to the rule R1. Indeed, since $\exists WANTS\text{-}TO\text{-}MARRY^{-}.\top(Mary)$ holds in every model of $\mathcal{B}$, it follows that in every model there exists a constant x such that *WANTS-TO-MARRY*$(x, Mary)$ holds in the model, consequently from rule R1 it follows that *happy*(*Mary*) also holds in the model.

■

6.3 Reasoning

In this section we study reasoning in $\mathcal{DL}$*+log*. In particular, we study satisfiability for finite $\mathcal{DL}$*+log*-KBs (as mentioned above, entailment can be easily reduced to satisfiability in $\mathcal{DL}$*+log*).

For ease of exposition, in the following we deal with the case when the DL is interpreted under the UNA: however, the algorithm can be easily extended to the case when unique names are not assumed in the DL (in a way analogous to the technique reported in [37] in the case of DL-safe rules).

We start by introducing Boolean conjunctive queries (CQs) and Boolean unions of conjunctive queries (UCQs), and the containment problem for such queries. A Boolean UCQ over a predicate alphabet Σ is a first-order sentence of the form $\exists \boldsymbol{x}.conj_1(\boldsymbol{x}) \vee \ldots \vee conj_n(\boldsymbol{x})$, where $\boldsymbol{x}$ is a tuple of variable symbols and each $conj_i(\boldsymbol{x})$ is a set of atoms whose predicates are in Σ and whose arguments are either constants or variables from $\boldsymbol{x}$. A Boolean CQ corresponds to a Boolean UCQ in the case when $n = 1$.

Given a $\mathcal{DL}$-TBox $\mathcal{T}$, a Boolean CQ Q_1 and a Boolean UCQ Q_2 over the alphabet $\Sigma_C \cup \Sigma_R$, Q_1 *is contained in* Q_2 *with respect to* $\mathcal{T}$, denoted by $\mathcal{T} \models Q_1 \subseteq Q_2$, iff, for every model $\mathcal{I}$ of $\mathcal{T}$, if Q_1 is satisfied in $\mathcal{I}$ then Q_2 is satisfied in $\mathcal{I}$. In the following, we call the problem of deciding $\mathcal{T} \models Q_1 \subseteq Q_2$ the *Boolean CQ/UCQ containment problem.*[7]

[7] This problem was called *existential entailment* in [28].

Algorithm Given a program $\mathcal{P}$, we denote by $\mathcal{C}_\mathcal{P}$ the set of constants occurring in $\mathcal{P}$.

In the following definition, we assume that a rule R in $\mathcal{P}$ has the form $\alpha_R(\boldsymbol{x}) \leftarrow \beta_R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}), \gamma_R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, where $\gamma_R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is the set of DL-atoms occurring in the body of R (and, of course, $\beta_R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w})$ is the set of Datalog atoms in the body of R), $\boldsymbol{x}$ are the head variables in R, $\boldsymbol{y}$ are the existential variables occurring both in DL-atoms and in Datalog atoms in R, and $\boldsymbol{z}$ (respectively, $\boldsymbol{w}$) are the existential variables of R that only occur in DL-atoms (respectively, Datalog atoms) in R.

Definition 3. *Let $\mathcal{B} = (\mathcal{K}, \mathcal{P})$ be a $\mathcal{DL}$+log-KB. The* DL-grounding of $\mathcal{P}$, *denoted by $gr_p(\mathcal{P})$, is the following set of Boolean CQs:*

$$
\begin{aligned}
gr_p(\mathcal{P}) = \{&\gamma_R(\boldsymbol{c}_1/\boldsymbol{x}, \boldsymbol{c_2}/\boldsymbol{y}, \boldsymbol{z}) \mid R \in \mathcal{P} \text{ and } \boldsymbol{c}_1, \boldsymbol{c_2} \text{ are tuples of constants in } \mathcal{C}_\mathcal{P}\} \\
&\cup \\
&\{p(\boldsymbol{c}/\boldsymbol{x}) \mid p \text{ is a DL predicate occurring in a rule head in } \mathcal{P} \\
&\qquad \text{and } \boldsymbol{c} \text{ is a tuple of constants in } \mathcal{C}_\mathcal{P}\}
\end{aligned}
$$

Notice that $gr_p(\mathcal{P})$ constitutes a *partial* grounding of the conjunctions of DL-atoms that occur in $\mathcal{P}$ with respect to the constants in $\mathcal{C}_\mathcal{P}$, since the variables that only occur in DL-atoms in the body of rules are not replaced by constants in $gr_p(\mathcal{P})$.

Let G be a set of Boolean CQs. Then, we denote by $CQ(G)$ the Boolean CQ corresponding to the conjunction of all the Boolean CQs in G, i.e., $CQ(G) = \bigwedge_{\gamma \in G} \gamma$. We also denote by $UCQ(G)$ the Boolean UCQ corresponding to the disjunction of all the Boolean CQs in G, namely $UCQ(G) = \bigvee_{\gamma \in G} \gamma$.[8]

Similarly to $gr(\mathcal{P}, \mathcal{C}_\mathcal{P})$, we define the *partial grounding of $\mathcal{P}$ on $\mathcal{C}_\mathcal{P}$* (denoted by $pgr(\mathcal{P}, \mathcal{C}_\mathcal{P})$) as the program obtained from $\mathcal{P}$ by grounding with the constants in $\mathcal{C}_\mathcal{P}$ all variables *except the existential variables of R that only occur in DL-atoms.*

Finally, given a partition (G_P, G_N) of $gr_p(\mathcal{P})$, we denote by $\mathcal{P}(G_P, G_N)$ the ground Datalog$^{\neg\vee}$ program obtained from $pgr(\mathcal{P}, \mathcal{C}_\mathcal{P})$ by:

- deleting all occurrences of the conjunction γ from the body of the rules, for each $\gamma \in G_P$;
- deleting each rule in which γ occurs in the body, for each $\gamma \in G_N$;
- deleting each rule in which γ occurs in the head, for each $\gamma \in G_P$;
- deleting all occurrences of the conjunction γ from the head of the rules, for each $\gamma \in G_N$.

Notice that $\mathcal{P}(G_P, G_N)$ is a ground Datalog$^{\neg\vee}$ program over Σ_D, i.e., no DL predicate occurs in such a program.

[8] Without loss of generality, we assume that each γ in G uses different existential variable symbols, so that the expression $\bigwedge_{\gamma \in G} \gamma$ can be immediately turned into a Boolean CQ by factoring out all existential quantifications (an analogous simple transformation is needed for turning $UCQ(G)$ into a Boolean UCQ).

We are now ready to present the algorithm $\mathcal{DL}$+*log*-SAT for deciding satisfiability of $\mathcal{DL}$+*log*-KBs. The algorithm is shown in Figure 3. The algorithm has a very simple structure, since it decides satisfiability by looking for a guess (G_P, G_N) of the Boolean CQs in $gr_p(\mathcal{P})$ that is consistent with the $\mathcal{DL}$-KB $\mathcal{K}$ and such that the Datalog$^{\neg\vee}$ program $\mathcal{P}(G_P, G_N)$ has a stable model.

Algorithm $\mathcal{DL}$+*log*-SAT($\mathcal{B}$)
Input: $\mathcal{DL}$+*log*-KB $\mathcal{B} = (\mathcal{K}, \mathcal{P})$ with $\mathcal{K} = (\mathcal{T}, \mathcal{A})$
Output: true if $\mathcal{B}$ is satisfiable, false otherwise
begin
  if there exists partition (G_P, G_N) of $gr_p(\mathcal{P})$
  such that
    (a) $\mathcal{P}(G_P, G_N)$ has a stable model **and**
    (b) $\mathcal{T} \not\models CQ(\mathcal{A} \cup G_P) \subseteq UCQ(G_N)$
  then return true
  else return false
end

Fig. 3. The algorithm $\mathcal{DL}$+*log*-SAT

Correctness of the algorithm is based on the following property, which relates consistency of a guess (G_P, G_N) of Boolean CQs with the problem of containment of a Boolean CQ in a Boolean UCQ with respect to a $\mathcal{DL}$-TBox.

Lemma 1. *There exists a model $\mathcal{M}$ for $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ such that every Boolean CQ in G_P is satisfied in $\mathcal{M}$ and every Boolean CQ in G_N is not satisfied in $\mathcal{M}$ if and only if $\mathcal{T} \not\models CQ(\mathcal{A} \cup G_P) \subseteq UCQ(G_N)$.*

Based on the above lemma, we are able to prove correctness of the algorthm $\mathcal{DL}$+*log*-SAT.

Theorem 1. *Let $\mathcal{B}$ be a $\mathcal{DL}$+log-KB. Then, $\mathcal{B}$ is satisfiable iff $\mathcal{DL}$+log-SAT($\mathcal{B}$) returns* true.

Decidability and complexity First, from the analysis of the algorithm $\mathcal{DL}$+*log*-SAT presented above, we are able to prove a very general property that states decidability of reasoning in $\mathcal{DL}$+*log* whenever the Boolean CQ/UCQ containment problem is decidable in $\mathcal{DL}$.

Theorem 2. *For every description logic $\mathcal{DL}$, satisfiability of $\mathcal{DL}$+log-KBs is decidable iff Boolean CQ/UCQ containment is decidable in $\mathcal{DL}$.*

From the above theorem and from previous results on query answering and query containment in DLs, we are able to state decidability of reasoning in

$\mathcal{DL}$+*log* in the case when $\mathcal{DL}$ corresponds to several known DLs. In particular, in the following we briefly analyze the description logics $\mathcal{ALCNR}$, $\mathcal{SHIQ}$, and *DL-Lite*.

First, we observe that, for the description logic $\mathcal{ALCNR}$ it is known that Boolean CQ/UCQ containment is decidable [28], hence reasoning in $\mathcal{ALCNR}$+*log*-KBs is decidable.

Theorem 3. *Satisfiability of $\mathcal{ALCNR}$+log-KBs is decidable.*

Of course, this result implies decidability of adding weakly-safe Datalog$^{\neg\vee}$ rules to all the DLs that are subsets of $\mathcal{ALCNR}$.

For (a large fragment of) the description logic $\mathcal{SHIQ}$ [25], it is known that answering conjunctive queries is decidable (see [32, 20]), but decidability of Boolean CQ/UCQ containment in $\mathcal{SHIQ}$ has not been established yet, therefore satisfiability in $\mathcal{SHIQ}$+*log* is still an open problem: however, we conjecture that Boolean CQ/UCQ containment in $\mathcal{SHIQ}$ is decidable as well, and hence that reasoning in $\mathcal{SHIQ}$+*log* is decidable.

Finally, for the description logic *DL-Lite* [8], there are known results about the complexity of query answering, which allow us to establish the computational complexity of reasoning in *DL-Lite*+*log* for different classes of Datalog programs. More precisely, the following theorem refers to *data complexity* of satisfiability, which in the framework of $\mathcal{DL}$+*log* corresponds to the analysis of the computational complexity of the problem when we only consider the size of the ABox $\mathcal{A}$ and of the EDB of $\mathcal{P}$, i.e., the set of facts contained in $\mathcal{P}$. In other words, data complexity considers the TBox $\mathcal{T}$ and the rules not corresponding to facts (i.e., the IDB) in $\mathcal{P}$ as fixed, hence they are not part of the input. Data complexity is a very significant measure when the size of the data, i.e., the ABox and the EDB of $\mathcal{P}$, is much larger than the size of the intensional knowledge, i.e., the TBox and the IDB of $\mathcal{P}$.

The following results are based on the analysis of the previous algorithms and on the fact that conjunctive query answering in *DL-Lite* is in PTIME in data complexity (actually it is in LOGSPACE) [8].

Theorem 4. *Let $\mathcal{B} = (\mathcal{K}, \mathcal{P})$ be a DL-Lite+log-KB. Then:*

- *if $\mathcal{P}$ is a positive Datalog program, then deciding satisfiability of $\mathcal{B}$ is PTIME-complete with respect to data complexity;*
- *if $\mathcal{P}$ is a positive disjunctive Datalog program, then deciding satisfiability of $\mathcal{B}$ is NP-complete with respect to data complexity;*
- *if $\mathcal{P}$ is an arbitrary Datalog$^{\neg\vee}$ program, then deciding satisfiability of $\mathcal{B}$ is Σ_2^p-complete with respect to data complexity.*

Therefore, in DL-lite, under both semantics, the data complexity does not increase with respect to the data complexity of the Datalog program alone [11]. In other words, connecting a *DL-Lite*-KB to a Datalog program does not increase complexity of reasoning in the size of the data. We also point out that *DL-Lite* with arbitrary, non-weakly-safe recursive Datalog rules is undecidable (which follows from the results in [28, 10]).

7 Open problems

We conclude the paper by pointing out some of the most interesting open problems in the integration of DLs and rules.

Semantics A first and crucial issue concerns the semantic account for logical systems integrating Description Logics and rules. In the paper, we have illustrated the technical problems due to the OWA of DLs and the CWA of nonmonotonic Datalog. However, there is also an orthogonal problem which can be summarized as follows: what is the "intended" semantics of a system combining DLs and rules? Research in this field is still far from providing an ultimate answer to the above question. With respect to this issue, in this paper we have only claimed that a minimal requirement that an appropriate semantics for such systems should satisfy is to constitute a "conservative extension" of both the semantics of DLs and the semantics of disjunctive Datalog.

Expressiveness Another important problem (which is directly related to the previous issue) concerns the expressiveness of a language integrating DLs and rules. In fact, the representational abilities that a system combining DLs and rules should provide to match "practical" needs are not completely clear.

In this respect, we believe that one of the most important expressive limitations of many of the current approaches to the integration of DLs and rules is the rigid separation between DL predicates and Datalog predicates. For instance, in $\mathcal{DL}$+*log*, since DL predicates have an open interpretation while Datalog predicates have a closed interpretation, it is not possible to express complex pieces of information in which the same predicate is interpreted in different ways (i.e., both under an open-world assumption and under a closed-world assumption) in different parts of the same knowledge base.

Reasoning As we have explained in the paper, decidability (and complexity) of reasoning is a crucial issue in systems combining DLs and rules. In this respect, there are numerous computational open problems, and the results obtained so far can be seen as the first, preliminary results towards the identification of general computational properties for systems combining DLs and rules.

One important general goal in this direction concerns the identification of the frontier between decidability and undecidability of reasoning with respect to the semantics and the expressiveness (in particular, the "degree of integration") of the formalism combining DLs and rules. In more abstract terms, this corresponds to analyze the trade-off between the expressiveness and the computational properties of such formalism, as usual in Knowledge Representation.

With respect to the above general goal, examples of more specific open problems are the following: (i) it is possible to identify tighter forms of decidable interaction between DL-KBs and rules, which are able to overcome the limitations of $\mathcal{DL}$+*log*? (ii) within the interaction between DLs and rules imposed by

the $\mathcal{DL}$+*log* framework, is it possible to establish more general computational properties? for instance, is it possible to establish decidability of $\mathcal{DL}$+*log* for very expressive DLs (like OWL-DL)?

Implementation There is still a considerable distance between the current state of the art in the integration of DLs and rules and the implementation of effective systems. In many approaches, reasoning techniques have not been defined yet, and even in the approaches which have addressed the reasoning problem, the proposed techniques for reasoning in DLs combined with rules have the main goal of establishing general computational properties (decidability and worst-case complexity) of the combined language. Therefore, the problem of turning such techniques into effective and implementable algorithms is still open and mainly unexplored.

As we have explained in the introduction, an important property towards this goal is modularity, i.e., the possibility of reducing reasoning in a system combining a DL component and a rule component to reasoning as "locally" as possible in the single components. On the other hand, it is clear that this modularity is in contrast with the representational goal of increasing the interaction between the DL component and the rule component. So, again, it is necessary to identify suitable trade-offs between such desiderata.[9]

Relationship between rules and queries Finally, the relationship between the integration of DLs and rules and query answering in DLs has not been fully explored yet. As described in the paper, the two problems are very strictly related, since queries can in principle be expressed in terms of rules. Therefore, the known results concerning query answering in DLs could be profitably used towards the design of an expressive and computationally attractive rule language for DLs (and vice versa). The $\mathcal{DL}$+*log* approach presented above constitutes a first step in this direction.

Acknowledgments

The author is grateful to a lot of people for many stimulating discussions on the subject of this paper. In particular, the author wishes to warmly thank Enrico Franconi, Ian Horrocks, Boris Motik, Marie-Christine Rousset, and Sergio Tessaris. A special acknowledgment goes to some of the (present and former) members of the Artificial Intelligence research group and the Data and Knowledge Bases research group at the Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", for introducing the author to this research topic, for their continuous help and for their pioneering work in this topic: Diego Calvanese, Giuseppe De Giacomo, Francesco Donini, Maurizio Lenzerini, Daniele Nardi, Andrea Schaerf.

[9] Modular techniques for dealing with the DL-safe integration of DLs and rules are described in [36, 37].

References

1. Rule interchange format working group charter. `http://www.w3.org/2005/rules/wg/charter`.
2. Grigoris Antoniou. A nonmonotonic rule system using ontologies. In *Proc. of the First International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2002)*, volume 60 of *CEUR Workshop Proceedings*, 2002.
3. Grigoris Antoniou. Nonmonotonic rule systems on top of ontology layers. In *Proc. of the 2002 International Semantic Web Conference (ISWC 2002)*, pages 394–398, 2002.
4. Grigoris Antoniou, A. Bikakis, and Gerd Wagner. A system for nonmonotonic rules on the web. In *Proc. of the Third International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004)*, pages 23–36, 2004.
5. Grigoris Antoniou, Carlos Viegas Damsio, Benjamin Grosof, Ian Horrocks, Michael Kifer, Jan Maluszynski, and Peter F. Patel-Schneider. Combining rules and ontologies. A survey. REWERSE Deliverable, `http://rewerse.net/publications#REWERSE-DEL-2005-I3-D3`.
6. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
7. Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *J. of Logic Programming*, 19–20:73–148, 1994.
8. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.
9. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.
10. Diego Calvanese and Riccardo Rosati. Answering recursive queries under keys and foreign keys is undecidable. In *Proc. of the 10th Int. Workshop on Knowledge Representation meets Databases (KRDB 2003)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-79/`, 2003.
11. Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.
12. Jos de Bruijn, Ruben Lara, Axel Polleres, and Dieter Fensel. OWL DL vs. OWL flight: conceptual modeling and reasoning for the semantic web. In *Proc. of the 14th international conference on World Wide Web (WWW 2005)*, pages 623–632, 2005.
13. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. $\mathcal{AL}$-log: Integrating Datalog and description logics. *J. of Intelligent Information Systems*, 10(3):227–252, 1998.
14. Thomas Eiter, Georg Gottlob, and Heikki Mannilla. Disjunctive Datalog. *ACM Trans. on Database Systems*, 22(3):364–418, 1997.
15. Thomas Eiter, Giovambattista Ianni, Axel Polleres, Roman Schindlauer, and Hans Tompits. Reasoning with rules and ontologies. In Pedro Barahona, François Bry, Enrico Franconi, Ulrike Sattler, and Nicola Henze, editors, *Reasoning Web, Second International Summer School 2005, Tutorial Lectures*, Lecture Notes in Computer Science. Springer, 2006.

16. Thomas Eiter, Nicola Leone, Cristinel Mateis, Gerald Pfeifer, and Francesco Scarcello. The KR system dlv: Progress report, comparison and benchmarks. In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.
17. Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. In *Proc. of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 141–151, 2004.
18. Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Well-founded semantics for description logic programs in the semantic web. In *Proc. of the Third International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004)*, pages 81–97, 2004.
19. Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
20. Birte Glimm, Ian Horrocks, and Ulrike Sattler. Conjunctive query answering for description logics with transitive roles. In *Proc. of the 2006 Description Logic Workshop (DL 2006)*, 2006.
21. Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proc. of the 12th Int. World Wide Web Conf. (WWW 2003)*, pages 48–57, 2003.
22. Stijn Heymans, Davy Van Nieuwenborgh, and Dirk Vermeir. Semantic web reasoning with conceptual logic programs. In *Proc. of the Third International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004)*, pages 113–127, 2004.
23. Stijn Heymans and Dirk Vermeir. Integrating description logics and answer set programming. In *Proc. of the 2003 International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2003)*, pages 146–159, 2003.
24. Ian Horrocks and Peter F. Patel-Schneider. A proposal for an OWL rules language. In *Proc. of the 13th international conference on World Wide Web (WWW 2004)*, pages 723–731, 2004.
25. Ian Horrocks and Ulrike Sattler. Decidability of SHIQ with complex role inclusion axioms. *Artificial Intelligence*, 160(1–2):79–104, 2004.
26. Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining Horn rules and description logics. In *Proc. of the 12th Eur. Conf. on Artificial Intelligence (ECAI'96)*, pages 323–327, 1996.
27. Alon Y. Levy and Marie-Christine Rousset. The limits on combining recursive Horn rules with description logics. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, pages 577–584, 1996.
28. Alon Y. Levy and Marie-Christine Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.
29. Jing Mei, Shengping Liu, Anbu Yue, and Zuoquan Lin. An extension to OWL with general rules. In *Proc. of the Third International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004)*, pages 155–169, 2004.
30. Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. In *Proc. of the 2004 International Semantic Web Conference (ISWC 2004)*, pages 549–563, 2004.
31. Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. *Web Semantics*, 3(1):41–60, 2005.

32. Maria Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of the 21th Nat. Conf. on Artificial Intelligence (AAAI 2006)*, 2006.
33. Peter F. Patel-Schneider, Patrick J. Hayes, Ian Horrocks, and Frank van Harmelen. OWL web ontology language; semantics and abstract syntax. W3C candidate recommendation, http://www.w3.org/tr/owl-semantics/, november 2002.
34. Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
35. Riccardo Rosati. Towards expressive KR systems integrating Datalog and description logics: Preliminary report. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 160–164. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.
36. Riccardo Rosati. On the decidability and complexity of integrating ontologies and rules. *Web Semantics*, 3(1):61–73, 2005.
37. Riccardo Rosati. Semantic and computational advantages of the safe integration of ontologies and rules. In *Proc. of the 2005 International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2005)*, volume 3703 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2005.
38. Riccardo Rosati. $\mathcal{DL}$+log: Tight integration of description logics and disjunctive datalog. In *Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, 2006. To appear.