Queries to Hybrid MKNF Knowledge Bases through Oracular Tabling

José Júlio Alferes, Matthias Knorr, and Terrance Swift

CENTRIA, Dep. Informática, Faculdade de Ciências e Tecnologia
Univ. Nova de Lisboa, 2825-516 Caparica, Portugal

Abstract. An important issue for the Semantic Web is how to combine open-world ontology languages with closed-world (non-monotonic) rule paradigms. Several proposals for hybrid languages allow concepts to be simultaneously defined by an ontology and rules, where rules may refer to concepts in the ontology and the ontology may also refer to predicates defined by the rules. Hybrid MKNF knowledge bases are one such proposal, for which both a stable and a well-founded semantics have been defined. The definition of Hybrid MKNF knowledge bases is parametric on the ontology language, in the sense that non-monotonic rules can extend any decidable ontology language. In this paper we define a query-driven procedure for Hybrid MKNF knowledge bases that is sound with respect to the original stable model-based semantics, and is correct with respect to the well-founded semantics. This procedure is able to answer conjunctive queries, and is parametric on an inference engine for reasoning in the ontology language. Our procedure is based on an extension of a tabled rule evaluation to capture reasoning within an ontology by modeling it as an interaction with an external oracle and, with some assumptions on the complexity of the oracle compared to the complexity of the ontology language, maintains the data complexity of the well-founded semantics for hybrid MKNF knowledge bases.

1 Introduction

Ontologies and Rules offer distinctive strengths for the representation and transmission of knowledge in the Semantic Web. Ontologies offer the deductive advantages of first-order logics with an open domain while guaranteeing decidability. Rules offer non-monotonic (closed-world) reasoning that can be useful for formalizing scenarios under (local) incomplete knowledge; they also offer the ability to reason about fixed points (e.g. reachability) which cannot be expressed within first-order logic. Interest in both and their combination is demonstrated by the pervasive interest in Ontology languages for the Semantic Web and the growing interest on Rule languages for the Semantic Web, cf. the RIF and the RuleML initiatives.

The two most common semantics for rules are the well-founded semantics (WFS) [19] and the answer-sets semantics [6]. Both semantics are widely used; both offer closed-world reasoning and allow the representation of fixed points;

furthermore the relationship between the two semantics has been fully explored. Of the two, the well-founded semantics is weaker (in the sense that it is more skeptical), but has the advantage that its lower complexity allows it to be integrated into the general-purpose programming language Prolog. Thus in addition to its features for knowledge representation, WFS rules can provide a reactive or procedural component missing from ontologies. Several formalisms have concerned themselves with combining decidable ontologies with WFS rules [3,5,9]. Among these, the Well-Founded Semantics for Hybrid MKNF knowledge bases ($MKNF_{WFS}$), introduced in [9] and overviewed in Section 2 below, is the only one which allows knowledge about instances to be fully inter-definable between rules and an ontology that is taken as a parameter of the formalism. Using this parameterized ontology, $MKNF_{WFS}$ is defined using a monotonic fixpoint operator that computes in each iteration step, besides the usual immediate consequences from rules, the set of all atoms derivable from the ontology whose ABox is augmented with the already proven atomic knowledge. The least fixpoint of the $MKNF_{WFS}$ operator coincides with the original WFS [19] if the DL-component is empty, and when dealing with tractable description logics $MKNF_{WFS}$ retains a tractable data complexity. Furthermore, $MKNF_{WFS}$ is sound wrt. to that of [12] for MKNF knowledge bases, which is based on answer-set semantics and coincides with the answer-sets semantics if the DL-part is empty.

In one sense, the fixpoint operator of $MKNF_{WFS}$ provides a way to compute, in a naive bottom-up fashion, all consequences of a knowledge base. However, such an approach is far from practical for large knowledge bases, as in the Semantic Web context. As a concrete example, consider a medical knowledge base about patients in a large research study. Such a knowledge base might use a standard OWL-ontology representing pathologies, treatment procedures, pharmaceuticals, and so on (e.g. http://www.mindswap.org/2003/CancerOntology). At the same time rules may be used to represent complex temporal constraints that a research study imposes on a given patient, to interface with a patient's electronic health record, and even to extend the ontology with local procedures or policies. To be practical this requires efficient techniques to answer queries about patients, health-care workers, and other objects of interest.

This paper presents a querying mechanism, called **SLG($\mathcal{O}$)**, that is sound and complete for $MKNF_{WFS}$, and sound for MKNF knowledge bases of [12]. **SLG($\mathcal{O}$)** accepts DL-safe conjunctive queries, (i.e. conjunctions of predicates with variables where queries have to be ground when processed in the ontology), returning all correct answer substitutions for variables in the query. To the best of our knowledge, this is the first query-driven, top-down like, procedure for knowledge bases that tightly combine an ontology with non-monotonic rules.

The Gist of the Approach

The main element of our approach addresses the interdependency of the ontology and rules. In particular, our program evaluation method SLG($\mathcal{O}$), presented in Section 4, extends SLG resolution [2], which evaluates queries to normal logic programs (i.e. sets of non-disjunctive non-monotonic rules) under WFS. SLG

is a form of tabled resolution that handles loops within the program, and does not change the data complexity of WFS. It does that by resorting to already computed results, in a forest of derivation trees, a technique also known as *tabling*. To adjoin an ontology to rules, the first thing that needs to be done is to allow an SLG evaluation to make calls to an inference engine for the ontology. Since MKNF is parametric on any given decidable ontology formalism[1], the inference engine is viewed in SLG as an oracle. In fact, every time SLG selects an atom that is (perhaps jointly) defined in the ontology, the oracle's inference engine must be called, in case the atom is not provable by the rules alone. Such a queried atom, say $P(a)$, might thus be provable but only if a certain set of atoms in turn is provable via rules. Our approach captures this by allowing the oracle to return a new program clause, say $P(a)$:- *Goals*, which has the property that (possibly empty) *Goals*, in addition to the axioms in the ontology and the atoms already proven by the program would be sufficient to prove $P(a)$. **SLG($\mathcal{O}$)** then treats these new clauses just as if they were program clauses. Note that, getting these conditional answers does not endanger decidability (or tractability, if it is the case) of reasoning in the ontology alone. In fact, it is easy to conceive a modification of a tableaux based inference engine for an ontology, that is capable of returning these conditional answers and is decidable if the tableaux algorithm is: add all those atoms that are defined in the program to the ABox; then proceed with the tableaux as usual, but collect all those added facts that have been used in the proof. Under some assumptions on the complexity of the oracle, it is shown (in Section 5 along with some other properties) that **SLG($\mathcal{O}$)** also retains tractability.

The other element of our approach arises from the need to combine the classical negation of an ontology with the non-monotonic negation of rules. This problem is similar to the issue of *coherence* that arises when adding strong negation to logic programs [6,13,14]: the strong (or classical) negation must imply negation by default. In our case, if the ontology entails that some atom A is false, then perforce the default negation *not A* must hold in the program. The derivation must accordingly be modified since the proof of *not A* cannot simply rely on the failure of the proof of A as it is usual in logic programming. For simplicity, instead of modifying **SLG($\mathcal{O}$)**, our proposal (in Section 3) transforms the original knowledge base $\mathcal{K}$ to ensure coherence. **SLG($\mathcal{O}$)** is then applied to the transformed $\mathcal{K}$. This transformation itself provides an alternative formulation of $MKNF_{WFS}$ and is another original result of the paper.

2   Preliminaries

2.1   Syntax of Hybrid MKNF Knowledge Bases

We presume a basic understanding of the well-founded semantics [19] and first-order logics, in particular notions related to logic programming and resolution

---

[1] The limitation to decidability is theoretically not strictly necessary but a choice to achieve termination and complexity results in accordance with the decidable ontology languages like OWL (`http://www.w3.org/2004/OWL/`).

(see e.g. [11]). Hybrid MKNF knowledge bases as introduced in [12] are essentially formulas in the logics of minimal knowledge and negation as failure (MKNF) [10], i.e. first-order logics with equality and two modal operators **K** and **not** allowing inspection of the knowledge base: intuitively, given a first-order formula φ, $\mathbf{K}\varphi$ asks whether φ is known while $\mathbf{not}\varphi$ is used to check whether φ is not known. Hybrid MKNF knowledge bases consist of two components, a decidable description logics (DL) knowledge base[2], translatable into first-order logics, and a finite set of rules.

Definition 1. *Let $\mathcal{O}$ be a DL knowledge base built over a language $\mathcal{L}$ with distinguished sets of countably infinitely many variables N_V, and finitely many individuals N_I, and predicates N_C. An atom $P(t_1, \ldots, t_n)$ where $P \in N_C$ and $t_i \in N_V \cup N_I$ is called a DL-atom if P occurs in $\mathcal{O}$, otherwise it is called non-DL-atom. An MKNF rule r has the following form where H_i, A_i, and B_i are atoms: $\mathbf{K}H \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m$. H is called the (rule) head and the sets $\{\mathbf{K}A_i\}$, and $\{\mathbf{not}B_j\}$ form the (rule) body. Literals[3] are positive literals $\mathbf{K}A$ or negative literals $\mathbf{not}A$. A rule r is positive if $m = 0$ and a fact if $n = m = 0$. A program $\mathcal{P}$ is a finite set of MKNF rules and a hybrid MKNF knowledge base $\mathcal{K}$ is a pair $(\mathcal{O}, \mathcal{P})$.*

We will usually omit the modal operators **K** in the rule head and the positive body, though they remain implicit however. Furthermore, we sometimes also omit the terms t_i of an atom as well (in the context of description logics).

For decidability DL-safety is applied which basically constrains the use of rules to individuals actually appearing in the knowledge base under consideration. Formally, an MKNF rule r is *DL-safe* if every variable in r occurs in at least one non-DL-atom $\mathbf{K}B$ occurring in the body of r. A hybrid MKNF knowledge base $\mathcal{K}$ is *DL-safe* if all its rules are DL-safe[4]. Likewise, to ensure decidability, grounding the knowledge base, i.e. its rules, is restricted to individuals appearing in the knowledge base and not to the whole infinite domain[5]. Therefore, given a hybrid MKNF knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$, the *ground instantiation of $\mathcal{K}$* is the KB $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ where $\mathcal{P}_G$ is obtained from $\mathcal{P}$ by replacing each rule r of $\mathcal{P}$ with a set of rules substituting each variable in r with constants from $\mathcal{K}$ in all possible ways (for more details we refer to [12] and [9]). DL-safety is also imposed on (conjunctive) queries:

Definition 2. *A* conjunctive query *q is a non-empty set, i.e. conjunction, of literals where each variable in q occurs in at least one non-DL atom in q. We also write q as a rule $q(X_i) \leftarrow A_1, \ldots, A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m$ where X_i is the (possibly empty) set of variables, appearing in the body, which are requested.*

---

[2] For a thorough introduction on description logics we refer to [1].

[3] In [9], the term modal atom is used and modal atoms in MKNF are in general not restricted to first-order atoms but in this paper it is essentially their only appearance.

[4] In the following all hybrid MKNF knowledge bases are assumed to be DL-safe.

[5] As well-known, description logics semantics usually require an infinite domain to admit the intended semantics for statements involving unknown individuals.

The restriction of conjunctive queries to DL-safety is not always necessary: for DLs like SHIQ conjunctive query answering is possible ([7]) and we may also make use of such existing algorithms, however, when there is no algorithm for conjunctive query answering yet or it is even not decidable (like for $\mathcal{EL}^{++}$ [15]) then the limitation is required to achieve decidability in the combined approach.

2.2 Well-Founded Semantics of Hybrid MKNF Knowledge Bases

The well-founded MKNF semantics as presented in [9] is based on a complete three-valued extension of the original MKNF semantics. However, here we are not interested in obtaining the entire semantics where a model consists of two sets of sets of first-order interpretations. Instead we limit ourselves here to the computation of what is called the well-founded partition in [9]: basically the literals which are true and false. For that reason, and in correspondence to logic programming, we will name this partition the well-founded model. At first, we recall some notions from [9] which will be useful in the definition of the operators for obtaining that well-founded model.

Definition 3. *Consider a hybrid MKNF knowledge base* $\mathcal{K} = (\mathcal{O}, \mathcal{P})$. *The set of* **K**-*atoms of* $\mathcal{K}$, *written* $\mathsf{KA}(\mathcal{K})$, *is the smallest set that contains (i) all positive literals occurring in* $\mathcal{P}$, *and (ii) a literal* $\mathbf{K}\xi$ *for each literal* $\mathbf{not}\xi$ *occurring in* $\mathcal{K}$. *Furthermore, for a set of literals* S, S_{DL} *is the subset of DL-atoms of* S, *and* $\widehat{S} = \{\xi \mid \mathbf{K}\xi \in S\}$.

Basically all literals appearing in the rules are collected in $\mathsf{KA}(\mathcal{K})$ as a set of positive literals and the other two notions provide restrictions on such a set.

To guarantee that all atoms that are false in the ontology are also false by default in the rules, we introduce new positive DL atoms which represent first-order false DL atoms, and another program transformation making these new literals available for reasoning in the respective rules.

Definition 4. *Let* $\mathcal{K}$ *be a hybrid MKNF knowledge base. We obtain* $\mathcal{K}^+ = (\mathcal{O}^+, P)$ *from* $\mathcal{K}$ *by adding an axiom* $\neg P \sqsubseteq NP$ *for every DL atom* P *which occurs as head in at least one rule in* $\mathcal{K}$ *where* NP *is a new predicate not already occurring in* $\mathcal{K}$. *Moreover, we obtain* $\mathcal{K}^*$ *from* $\mathcal{K}^+$ *by adding* $\mathbf{not}NP(t_1, \ldots, t_n)$ *to the body of each rule with a DL atom* $P(t_1, \ldots, t_n)$ *in the head.*

By $\mathcal{K}^+$, NP represents $\neg P$ (with its corresponding arguments) and $\mathcal{K}^*$ introduces a restriction on each rule with such a DL atom in the head saying intuitively that the rule can only be used to conclude the head if the negation of its head does not hold already[6].

We continue now by defining an operator $T_{\mathcal{K}}$ which allows to draw conclusions from positive hybrid MKNF knowledge bases.

---

[6] Note that $\mathcal{K}^+$ and $\mathcal{K}^*$ are still hybrid MKNF knowledge bases, so we only refer to $\mathcal{K}^+$ and $\mathcal{K}^*$ explicitly when it is necessary.

Definition 5. *For $\mathcal{K}$ a positive hybrid MKNF knowledge base, $R_{\mathcal{K}}$, $D_{\mathcal{K}}$, and $T_{\mathcal{K}}$ are defined on the subsets of $\mathsf{KA}(\mathcal{K}^*)$ as follows:*

$$R_{\mathcal{K}}(S) = S \cup \{\mathbf{K}H \mid \mathcal{K} \text{ contains a rule of the form (1) such that } \mathbf{K}A_i \in S$$
$$\text{for each } 1 \leq i \leq n\}$$
$$D_{\mathcal{K}}(S) = \{\mathbf{K}\xi \mid \mathbf{K}\xi \in \mathsf{KA}(\mathcal{K}^*) \text{ and } \mathcal{O} \cup \widehat{S}_{DL} \models \xi\} \cup \{\mathbf{K}Q(b_1, \ldots, b_n) \mid$$
$$\mathbf{K}Q(a_1, \ldots, a_n) \in S \setminus S_{DL}, \; \mathbf{K}Q(b_1, \ldots, b_n) \in \mathsf{KA}(\mathcal{K}^*), \text{ and}$$
$$\mathcal{O} \cup \widehat{S}_{DL} \models a_i \approx b_i \;\; \text{for } 1 \leq i \leq n\}$$
$$T_{\mathcal{K}}(S) = R_{\mathcal{K}}(S) \cup D_{\mathcal{K}}(S)$$

$R_{\mathcal{K}}$ derives consequences from the rules while $D_{\mathcal{K}}$ obtains knowledge from the ontology $\mathcal{O}$, respectively from non-DL-atoms and the equalities occuring in $\mathcal{O}$.

The operator $T_{\mathcal{K}}$ is shown to be monotonic in [9] so, by the Knaster-Tarski theorem, it has a unique least fixpoint, denoted $\mathsf{lfp}(T_{\mathcal{K}})$, which is reached after a finite number of iteration steps.

The computation follows the alternating fixpoint construction [18] of the well-founded semantics for logic programs which neccesitates turning a hybrid MKNF knowledge base into a positive one to make $T_{\mathcal{K}}$ applicable.

Definition 6. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and let $S \subseteq \mathsf{KA}(\mathcal{K}_G)$. The* MKNF transform $\mathcal{K}_G/S = (\mathcal{O}, \mathcal{P}_G/S)$ *is obtained by $\mathcal{P}_G/S$ containing all rules $H \leftarrow A_1, \ldots, A_n$ for which there exists a rule $\mathbf{K}H \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m$ in $\mathcal{P}_G$ with $\mathbf{K}B_j \notin S$ for all $1 \leq j \leq m$.*

This resembles the transformation known from answer-sets [6] of logic programs and the following two operators are defined.

Definition 7. *Let $\mathcal{K}$ be a hybrid MKNF knowledge base and $S \subseteq \mathsf{KA}(\mathcal{K}^*)$.*

$$\Gamma_{\mathcal{K}}(S) = \mathsf{lfp}(T_{\mathcal{K}_G^+/S}) \qquad \Gamma'_{\mathcal{K}}(S) = \mathsf{lfp}(T_{\mathcal{K}_G^*/S})$$

Both operators are shown to be antitonic [9] and form the basis for defining the well-founded MKNF model. Here we present its alternating computation.

$$\mathbf{P}_0 = \emptyset \qquad\qquad \mathbf{N}_0 = \mathsf{KA}(\mathcal{K}^*)$$
$$\mathbf{P}_{n+1} = \Gamma_{\mathcal{K}}(\mathbf{N}_n) \qquad\qquad \mathbf{N}_{n+1} = \Gamma'_{\mathcal{K}}(\mathbf{P}_n)$$
$$\mathbf{P}_{\omega} = \bigcup \mathbf{P}_n \qquad\qquad \mathbf{N}_{\omega} = \bigcap \mathbf{N}_n$$

Note that by finiteness of the ground knowledge base the iteration stops before reaching ω. It was shown in [9] that the sequences are monotonically increasing, decreasing respectively, and that $\mathbf{P}_{\omega}$ and $\mathbf{N}_{\omega}$ form the well-founded MKNF model.

Definition 8. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base and let $\mathbf{P}_{\mathcal{K}}, \mathbf{N}_{\mathcal{K}} \subseteq \mathsf{KA}(\mathcal{K})$ with $\mathbf{P}_{\mathcal{K}}$ being $\mathbf{P}_{\omega}$ and $\mathbf{N}_{\mathcal{K}}$ being $\mathbf{N}_{\omega}$, both restricted to the literals only occurring in $\mathsf{KA}(\mathcal{K})$. Then $M_{WF} = \{\mathbf{K}A \mid A \in \mathbf{P}_{\mathcal{K}}\} \cup \{\mathbf{K}\pi(\mathcal{O})\} \cup \{\mathbf{not}A \mid A \in \mathsf{KA}(\mathcal{K}) \setminus \mathbf{N}_{\mathcal{K}}\}$ is the* well-founded MKNF model *of $\mathcal{K}$.*

All literals in M_{WF} are true, its counterparts are false (e.g. if $\mathbf{K}H$ is true then $\mathbf{not}H$ is false) and all other literals from $\mathsf{KA}(\mathcal{K})$ are undefined. Note that $\mathbf{K}\pi(\mathcal{O})$ appears in the model for conciseness with [9].

3   Alternative Computation of $MKNF_{WFS}$

As we have seen, the bottom-up computation of the well-founded MKNF model requires essentially two operators each with its own transformation of the knowledge base. Using directly this as a basis for the top-down procedure, would complicate it, in that we would have to consider two different copies of the program, and use them alternately in different parts of the procedure. This is why, in this section, we define that computation in a different way. Namely, we double the rules in $\mathcal{K}$ using new predicates, transform them appropriately, and double the ontology, so that we can apply just one operator and still obtain the well-founded MKNF model.

Definition 9. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base. We introduce new predicates, i.e. a predicate A^d for each predicate A appearing in $\mathcal{K}$, and define $\mathcal{K}^d$ as the knowledge base obtained by adding $\mathcal{O}^+$ to $\mathcal{O}$ where each predicate A in $\mathcal{O}$ is substituted by A^d, and transforming each $H(t_i) \leftarrow A_1, \ldots, A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m$ occuring in $\mathcal{P}$, t_i representing the arguments of H, into the following two rules:*

(1) $H(t_i) \leftarrow A_1, \ldots, A_n, \mathbf{not}B_1^d, \ldots, \mathbf{not}B_m^d$ and either
(2a) $H^d(t_i) \leftarrow A_1^d, \ldots, A_n^d, \mathbf{not}B_1, \ldots, \mathbf{not}B_m, \mathbf{not}NH(t_i)$ if H is a DL-atom;
     or
(2b) $H^d(t_i) \leftarrow A_1^d, \ldots, A_n^d, \mathbf{not}B_1, \ldots, \mathbf{not}B_m$ otherwise

Note that the predicate $\mathbf{not}NH$ is in fact the one introduced by $\mathcal{K}^+$.
    We can now define a new operator Γ^d on $\mathcal{K}^d$ only[7].

Definition 10. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base and $S \subseteq \mathsf{KA}(\mathcal{K}^d)$. We define $\Gamma_{\mathcal{K}}^d(S) = \mathsf{lfp}(T_{\mathcal{K}_G^d/S})$ and $\Upsilon_{\mathcal{K}}(S) = \Gamma_{\mathcal{K}}^d(\Gamma_{\mathcal{K}}^d(S))$.*

The operator $\Gamma_{\mathcal{K}}^d$ is antitonic just like $\Gamma_{\mathcal{K}}$, and so $\Upsilon_{\mathcal{K}}$ is a monotonic operator. Therefore $\Upsilon_{\mathcal{K}}$ also has a least and a greatest fixpoint and we can formulate their iteration in the same manner as for $\mathbf{P}_\omega$ and $\mathbf{N}_\omega$.

$$\begin{aligned}\mathbf{P}_0^d &= \emptyset & \mathbf{N}_0^d &= \mathsf{KA}(\mathcal{K}^d) \\ \mathbf{P}_{n+1}^d &= \Gamma_{\mathcal{K}}^d(\mathbf{N}_n^d) & \mathbf{N}_{n+1}^d &= \Gamma_{\mathcal{K}}^d(\mathbf{P}_n^d) \\ \mathbf{P}_\omega^d &= \bigcup \mathbf{P}_n^d & \mathbf{N}_\omega^d &= \bigcap \mathbf{N}_n^d\end{aligned}$$

We can now state the relation of the least and the greatest fixpoint of $\Upsilon_{\mathcal{K}}$ to $\mathbf{P}_\omega$ and $\mathbf{N}_\omega$, from which the well-founded MKNF model is obtained.

Theorem 1. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base and let $\mathbf{P}_\omega^d$ be the least fixpoint of $\Upsilon_{\mathcal{K}}$ and $\mathbf{N}_\omega^d$ be the greatest fixpoint of $\Upsilon_{\mathcal{K}}$. We have:*

– $A \in \mathbf{P}_\omega$ if and only if $A \in \mathbf{P}_\omega^d$
– $B \notin \mathbf{N}_\omega$ if and only if $B^d \notin \mathbf{N}_\omega^d$

It follows immediately from this theorem that we can use $\Upsilon_{\mathcal{K}}$ to compute the well-founded MKNF model. We also derive from the theorem that we have to use the new predicates A^d if we query for negative literals.

---

[7] Note that the operators in Definition 5 are now defined for subsets of $\mathsf{KA}(\mathcal{K}^d)$.

4   Tabled SLG($\mathcal{O}$)-Resolution for Hybrid MKNF

Now we present the new SLG-resolution, **SLG($\mathcal{O}$)**, for Hybrid MKNF Bases which extends [2]. We should mention that the definition of **SLG($\mathcal{O}$)** is quite involved and requires that certain definitions are interlinked with each other (links are provided in each of these cases). It is based on sets of trees of derivations (forests). Tree nodes contain sets of literals which we also call *goals* (cf. Def.12). To deal with termination, some literals must be delayed during the tabled resolution, and so we have to define delay literals (cf. Def.11). Also, a notion of completely evaluated goals in a forest is needed for termination (cf. Def.14). The trees, and the delaying of literals, are constructed according to the set of operations in Def.18. Some of these operations require resolution of selected literals with program rules and, since delayed literals may exist, a slightly changed resolution is required (cf. Def.13). For dealing with the ontology, derivation steps must also take into account an oracle; thus, a suitable definition of what is expected from the oracle is required (cf. Def.17). We begin the presentation of **SLG($\mathcal{O}$)** by defining the delay literals, and then forests:

Definition 11. *A negative delay literal has the form* **not** *A, where A is a ground atom. Positive delay literals have the form A_{Answer}^{Call}, where A is an atom whose truth value depends on the truth value of some literal Answer for the literal Call. If θ is a substitution, then $(A_{Answer}^{Call})\theta = (A\theta)_{Answer}^{Call}$.*

Positive delay literals can only appear as a result of resolution. Its special form as indicated by the names is meant to keep track of the answer and call used for that resolution and possible substitutions are thus only applied to A itself.

Definition 12. *A node has the form*

$$Answer\_Template \text{ :- } Delays|Goals \qquad or \qquad fail.$$

In the first form, Answer_Template is an atom, Delays is a sequence of (positive and negative) delay literals and Goals is a sequence of literals. The second form is called a failure node. A program tree T is a tree of nodes whose root is of the form S :- $|S$ for some atom S: S is the root node for T and T is the tree for S. An SLG forest $\mathcal{F}$ is a set of program trees. A node N is an answer when it is a leaf node for which Goals is empty. If Delays of an answer is empty it is termed an unconditional answer, otherwise, it is a conditional answer. Program trees T may be marked as complete.

Whenever *Goals* contains various elements we effectively have to select one of them, by using a *selection function*. The only requirement for such a selection function, is that DL-atoms are not selected until they are ground (which is always possible given DL-safety).

The definition of answer resolution is slightly different from the usual one to take delay literals in conditional answers into account.

Definition 13. *Let N be a node A :- $D|L_1, ..., L_n$, where $n > 0$. Let $Ans = A'$:- $D'|$ be an answer whose variables have been standardized apart from N.*

N is SLG resolvable *with Ans if $\exists i$, $1 \leq i \leq n$, such that L_i and A' are unifiable with an mgu θ. The SLG resolvent of N and Ans on L_i has the form:*

$$(A \;\text{:-}\; D | L_1, ..., L_{i-1}, L_{i+1}, ..., L_n)\theta$$

if D' is empty; otherwise the resolvent has the form:

$$(A \;\text{:-}\; D, L_{i\,A'}^{L_i} | L_1, ..., L_{i-1}, L_{i+1}, ..., L_n)\theta$$

We delay L_i rather than propagating the answer's delay list. This is necessary, as shown in [2], to ensure polynomial data complexity[8].

At a certain point in SLG($\mathcal{O}$) resolution, a set of goals may be completely evaluated, i.e. it can produce no more answers.

Definition 14. *A set $\mathcal{S}$ of literals is* completely evaluated *if at least one of the conditions holds for each $S \in \mathcal{S}$*

1. *The tree for S contains an answer $S \;\text{:-}\; |$; or*
2. *For each node N in the tree for S:*
    (a) *The underlying subgoal[9] of the selected literal of N is completed; or*
    (b) *The underlying subgoal of the selected literal of N is in $\mathcal{S}$ and there are no applicable* New Subgoal, Program Clause Resolution, Oracle Resolution, Equality Resolution, Positive Return, Negative Return *or* Delaying *operations (Definition 18) for N.*

Once a set of literals is determined to be completely evaluated, the Completion operation marks the trees for each literal (Definition 12). Such completely evaluated trees can then be used to simplify other trees in the evaluation.

According to Definition 13, if a conditional answer is resolved against the selected literal in the set *Goals* of a node, the information about the delayed literals in the answer is not propagated. However, in certain cases, the propagation of conditional answers can lead to a set of *unsupported answers* — conditional answers that are false in the well founded model (see e.g. Example 1 of [17])[10].

Definition 15. *Let $\mathcal{F}$ be an SLG forest, S a root of a tree in $\mathcal{F}$, and Answer be an atom that occurs in the head of some answer of S. Then Answer is* supported *by S in $\mathcal{F}$ if and only if:*

1. *S is not completely evaluated; or*
2. *there exists an answer node Answer :- Delays| of S such that for every positive delay literal D_{Ans}^{Call}, Ans is supported by Call.*

We can obtain an interpretation from an SLG forest representing the truth values of the roots of its trees. This interpretation will later also correspond to M_{WF} (cf. Theorem 4).

---

[8] If we propagated the delay lists, we would propagate all derivations which could be exponential in bad cases.

[9] The *underlying subgoal* of literal L is L if L is positive and S if $L = \text{not } S$.

[10] As an aside, we note that unsupported answers appear to be uncommon in practical evaluations which minimize the use of delay such as [16].

Definition 16. *Let $\mathcal{F}$ be a forest. Then the* interpretation induced by $\mathcal{F}$, $I_{\mathcal{F}}$, *is the smallest set such that:*

- *A (ground) atom $A \in I_{\mathcal{F}}$ iff A is in the ground instantiation of some unconditional answer Ans :- | in $\mathcal{F}$.*
- *A (ground) literal not $A \in I_{\mathcal{F}}$ iff A is in the ground instantiation of a completely evaluated literal in $\mathcal{F}$, and A is not in the ground instantiation of any answer in a tree in $\mathcal{F}$.*

An atom S is successful *(*failed*) in $I_{\mathcal{F}}$ if S' (not S') is in $I_{\mathcal{F}}$ for every S' in the ground instantiation of S. A negative delay literal not D is successful (failed) in a forest $\mathcal{F}$ if D is (failed) successful in $\mathcal{F}$. Similarly, a positive delay literal D_{Ans}^{Call} is successful (failed) in a $\mathcal{F}$ if $Call$ has an unconditional answer Ans :- | in $\mathcal{F}$.*

In order to describe a tabled evaluation that is parameterized by an oracle, we need to characterize the behavior of an abstract oracle, $\mathcal{O}$[11] that computes entailment according to a theory, i.e. the ontology. For that purpose, we define an oracle transition function that in just one step computes all possible atoms required to prove the goal. In other words, such an oracle, when posed a query S non-deterministically returns in one step a set of atoms defined in the program (i.e. atoms for which there is at least one rule with it in the head) such that, if added to the oracle theory, immediately derives S.

Definition 17. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base, S a goal, and L a set of ground atoms which appear in at least one rule head in $\mathcal{P}_G$. The* complete transition function *for $\mathcal{O}$, denoted $compT_{\mathcal{O}}$, is defined by*

$$compT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L) \text{ iff } \mathcal{O} \cup I_{\mathcal{F}_n} \cup L \models S$$

We are now able to characterize $SLG(\mathcal{O})$ operations.

Definition 18 ($SLG(\mathcal{O})$ Operations). *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base. Given a forest $\mathcal{F}_n$ of an $SLG(\mathcal{O})$ evaluation of $\mathcal{K}$, $\mathcal{F}_{n+1}$ may be produced by one of the following operations.*

1. NEW SUBGOAL: *Let $\mathcal{F}_n$ contain a tree with non-root node*

   $$N = Ans \text{ :- } Delays | G, Goals$$

   where G is the selected literal S or not S. Assume $\mathcal{F}_n$ contains no tree with root S. Then add the tree S :- $|S$ to $\mathcal{F}_n$.
2. PROGRAM CLAUSE RESOLUTION: *Let $\mathcal{F}_n$ contain a tree with root node $N = S$:- $|S$ and C be a rule $Head$:- $Body$ such that $Head$ unifies with S with mgu θ. Assume that in $\mathcal{F}_n$, N does not have a child $N_{child} = (S$:- $|Body)\theta$. Then add N_{child} as a child of N.*

---

[11] We overload $\mathcal{O}$ syntactically to represent the oracle and the ontology, since semantically they are the same anyway.

3. ORACLE RESOLUTION: *Let $\mathcal{F}_n$ contain a tree with root node $N = S :- |S$ and S and all $G \in Goals$ be DL-atoms. Assume that $compT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, Goals)$. If N does not have a child $N_{child} = S :- |Goals$ in $\mathcal{F}_n$ then add N_{child} as a child*

4. EQUALITY RESOLUTION: *Let $\mathcal{F}_n$ contain a tree with root node $N = S :- |S$ where S and $G \in Goal$ are ground non-DL-atoms with the identical predicate. Assume that $compT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, Goal)$. If N does not have a child $N_{child} = S :- |Goal$ in $\mathcal{F}_n$ then add N_{child} as a child*

5. POSITIVE RETURN: *Let $\mathcal{F}_n$ contain a tree with non-root node N whose selected literal S is positive. Let Ans be an answer for S in $\mathcal{F}_n$ and N_{child} be the SLG resolvent of N and Ans on S. Assume that in $\mathcal{F}_n$, N does not have a child N_{child}. Then add N_{child} as a child of N.*

6. NEGATIVE RETURN: *Let $\mathcal{F}_n$ contain a tree with a leaf node, whose selected literal not S is ground*

$$N = Ans :- Delays|not\ S, Goals.$$

   (a) NEGATION SUCCESS: *If S is failed in $\mathcal{F}$ then create a child for N of the form: Ans :- Delays|Goals.*
   (b) NEGATION FAILURE: *If S succeeds in $\mathcal{F}$, then create a child for N of the form* fail.

7. DELAYING: *Let $\mathcal{F}_n$ contain a tree with leaf $N = Ans :- Delays|not\ S, Goals$, such that S is ground, in $\mathcal{F}_n$, but S is neither successful nor failed in $\mathcal{F}_n$. Then create a child for N of the form Ans :- Delays, not S|Goals.*

8. SIMPLIFICATION: *Let $\mathcal{F}_n$ contain a tree with leaf node $N = Ans :- Delays|$, and let $L \in Delays$*
   (a) *If L is failed in $\mathcal{F}$ then create a child* fail *for N.*
   (b) *If L is successful in $\mathcal{F}$, then create a child Ans :- Delays'| for N, where $Delays' = Delays - L$.*

9. COMPLETION: *Given a completely evaluated set $\mathcal{S}$ of literals (Definition 14), mark the trees for all literals in $\mathcal{S}$ as completed.*

10. ANSWER COMPLETION: *Given a set of unsupported answers $\mathcal{UA}$, create a failure node as a child for each answer $Ans \in \mathcal{UA}$.*

The only thing now missing is the formalization of the initialization of an SLG evaluation process.

Definition 19. *Let $\mathcal{K}$ be a hybrid MKNF knowledge base and let q be a query of the form $q(X_i) \leftarrow A_1, \ldots, A_n, \mathbf{not} B_1, \ldots, \mathbf{not} B_m$ where X_i is the (possibly empty) set of requested variables. We set $\mathcal{F}_0 = \{q(X_i) : - \ | \ q(X_i)\}$ to be the initial forest of an SLG($\mathcal{O}$) evaluation of $\mathcal{K}^d$ for q.*

Of course, if the query is atomic we can usually simply start with that atomic query. Note that since we use $\mathcal{K}^d$, the technically correct way to query negative literals is to use $\mathbf{not} B^d$ instead of $\mathbf{not} B$ for any atom B.

Example 1. In order to illustrate the actions of SLG($\mathcal{O}$) we consider a derivation of an answer to the query `?- discount(bill)` to the KB due to [12][12]:

---

[12] We adopt that only DL-atoms start with a capital letter. Also, to ease the reading, and since it has no influence in this example, instead of $\mathcal{K}^d$ we operate on $\mathcal{K}$ directly.

$$NonMarried \equiv \neg Married \qquad\qquad \neg Married \sqsubseteq HighRisk$$
$$\exists Spouse.T \sqsubseteq Married \qquad\qquad bill \in (\exists Spouse.michelle)$$
$$NonMarried(X) \leftarrow \textbf{not } Married(X).$$
$$discount(X) \leftarrow \textbf{not } HighRisk(X)$$

Note that both TBox and ABox information are each distributed over both the description logic and the program. Figure 1 shows the final forest for this evaluation, where elements are marked in the order they are created. The initial forest for the evaluation consists of node 0 only. Since the selected literal of node 0, `discount(bill)` is a non-DL-atom and there are no equalities in the KB, we can only apply PROGRAM CLAUSE RESOLUTION which produces node 1, followed by a NEW SUBGOAL to produce node 2. Node 2 is a DL-atom, there are no rules applicable for `HighRisk(bill)`, but an ORACLE RESOLUTION operation can be applied to derive $bill \in NonMarried$ (node 3). Then via a NEW SUBGOAL operation node 4 is obtained. The selected literal for node 4, `NonMarried(bill)` is a DL-atom that also is the head of a rule, so the oracle and the program evaluation may both try to derive the atom. On the program side, PROGRAM CLAUSE RESOLUTION produces nodes 5 and 6. The selected literal of node 6, `Married(bill)`, is a DL-atom that is not the head of a program rule, so once again the only possibility is to use ORACLE RESOLUTION, and derive `Married(bill)`; using this a NEGATIVE RETURN operation produces node 8, and the tree for `Married(bill)` can be early completed. The tree for `NonMarried(bill)` which does not have an answer must be completed (step 10), and the same for `HighRisk(bill)` (step 11). Once this occurs, a NEGATIVE RETURN operation is enabled to produce node 12.
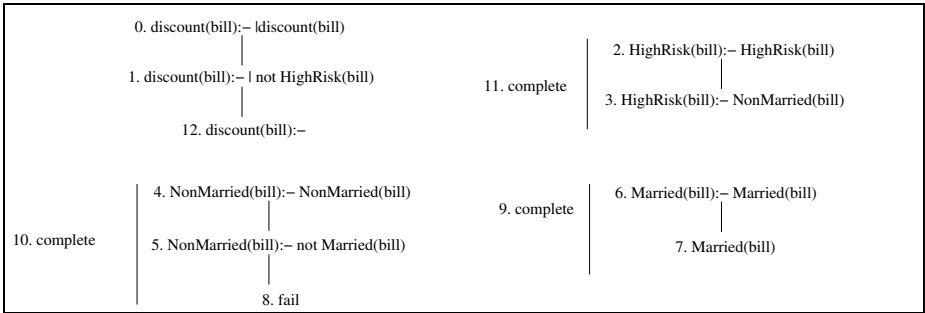

Fig. 1. Final Forest for query `?- discount(bill)` to $\mathcal{K}_1$

The evaluation illustrates several points. First, the evaluation makes use of classical negation in the ontology along with closed world negation in the rules. From an operational perspective, the actions of the description logic prover and the program are interleaved, with the program "calling" the oracle by creating new trees for DL-atoms, and the oracle "calling" the rule system through ORACLE RESOLUTION operations. As a result, trees for DL-atoms must either be early-completed, or explicitly completed by the tabulation system.

5   Properties

Theorem 2. *Let $q = L$ be a query to a hybrid MKNF knowledge base $\mathcal{K}$. Then any* **SLG($\mathcal{O}$)** *evaluation of q will terminate after finitely many steps, producing a finite final forest.*

The way **SLG($\mathcal{O}$)** is defined there is no real order in which to apply any of the operations possible in a forest $\mathcal{F}_i$. Some orders of application are in general more efficient than others but it was shown in [2] that any order yields the same outcome for any query. We adopt this statement here to **SLG($\mathcal{O}$)**.

Theorem 3. *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two* **SLG($\mathcal{O}$)** *evaluations of a query $q = L$ to a hybrid knowledge based $\mathcal{K}_G^d$. Let $\mathcal{F}_1$ be the final forest of $\mathcal{E}_1$ and $\mathcal{F}_2$ be the final forest of $\mathcal{E}_2$. Then, $I_{\mathcal{F}_1} = I_{\mathcal{F}_2}$.*

This theorem will also be helpful when it comes to proving that **SLG($\mathcal{O}$)** is in fact a query procedure for $MKNF_{WFS}$ and may terminate within the same complexity bounds as the semantics defined in [9]. At first, we will show that the procedure presented in the previous section coincides with $MKNF_{WFS}$. Intuitively, what we have to show is that the well-founded MKNF model, as presented in section 2 and based on the computation presented in Section 3, and the interpretation $I_{\mathcal{F}}$ induced by $\mathcal{F}_n$ for some query q to $\mathcal{K}^d$ coincide for each ground literal appearing in $\mathcal{K}_G^d$. We can simplify that by showing for each literal L appearing in $\mathcal{K}_G^d$ that $L \in M_{WF}$ if and only if $L \in I_{\mathcal{F}}$ with query $q = L$ and $\mathcal{F}_n$ for some n. Additionally, we prove that the same correspondence holds for (positive)[13] atoms only appearing in the ontology.

Theorem 4. *Let $\mathcal{K}$ be a hybrid MKNF knowledge base and L be a modal atom which appears in $\mathcal{K}_G^d$.* **SLG($\mathcal{O}$)** *resolution is correct and complete wrt.* MKNF_{WFS}, *i.e. $L \in M_{WF}$ if and only if $L \in I_{\mathcal{F}}$ where $I_{\mathcal{F}}$ is induced by the forest $\mathcal{F}$ of an* **SLG($\mathcal{O}$)** *evaluation of $\mathcal{K}_G^d$ for query $q = L$ and, for atoms P not appearing in any rule, $M_{WF} \models P$ if and only if $P \in I_{\mathcal{F}}$.*

Given the soundness of $MKNF_{WFS}$ wrt. the semantics of MKNF knowledge bases of [12], it follows easily that:

Corollary 1. *Let $\mathcal{K}$ be a consistent hybrid MKNF knowledge base and L be a modal atom which appears in $\mathcal{K}_G^d$. If $L \in I_{\mathcal{F}}$, where $I_{\mathcal{F}}$ is induced by the forest $\mathcal{F}$ of an* **SLG($\mathcal{O}$)** *evaluation of $\mathcal{K}_G^d$ for query $q = L$, then L belongs to all MKNF two-valued models (as in [12]) of $\mathcal{K}$.*

In addition to the interpretation of the final forest $I_{\mathcal{F}}$ being sound with respect to the 2-valued MKNF model, the conditional answers in $\mathcal{F}$ can be seen as a well-founded reduct of the rules in $\mathcal{K}$, augmented with conditional answers derived through ORACLE RESOLUTION and EQUALITY RESOLUTION operations. As a

---

[13] We cannot query directly for explicit negated atoms, however, a simple transformation similar to the one yielding $\mathcal{K}^+$ provides a solution to that problem.

result, the final forest can be seen as a *residual program*: a sound transformation not only of the rules, but of information from the oracle, and can be used to construct a partial 2-valued stable model.

Regarding complexity, it is clear that the complexity of the whole $\mathbf{SLG}(\mathcal{O})$ depends on the complexity of the oracle, and also on the number of results returned by each call to the oracle. Clearly, the complexity associated to the computation of one result of the oracle function is a lower-bound of the complexity of $\mathbf{SLG}(\mathcal{O})$. Moreover, even if e.g. the computation of one result of the oracle is tractable, if exponentially many solutions are generated by the oracle (e.g. returning all supersets of a solution), then the complexity of $\mathbf{SLG}(\mathcal{O})$ becomes exponential. This is so, because our definition of the oracle is quite general, and in order to prove interesting complexity results some assumptions most be made about the oracle. We start by defining a correct partial oracle:

Definition 20. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base, S a goal, and L a set of ground atoms which appear in at least one rule head in $\mathcal{P}_G$ (called program atoms). A* partial oracle *for $\mathcal{O}$, denoted $pT_{\mathcal{O}}$, is a relation $pT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L)$ such that if $pT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L)$ then $\mathcal{O} \cup I_{\mathcal{F}_n} \cup L \models S$.*

A partial oracle $pT_{\mathcal{O}}$ is correct *iff when replacing $compT_{\mathcal{O}}$ in $\mathbf{SLG}(\mathcal{O})$ succeeds for exactly the same set of queries.*

Note that the complete oracle is indeed generating unnecessarily many answers, and it can be replaced by a partial one which assures correctness. E.g. consider a partial oracle that does not return supersets of other results. Such a partial oracle is obviously correct. Making assumptions on the complexity and number of results of an oracle, complexity results of $\mathbf{SLG}(\mathcal{O})$ are obtained.

Theorem 5. *Let $pT_{\mathcal{O}}$ be a correct partial oracle for the hybrid MKNF knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$, such that for every goal S, the cardinality of $pT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L)$ is bound by a polynomial on the number of program atoms. Moreover, assume that computing each element of $pT_{\mathcal{O}}$ is decidable with data complexity $\mathcal{C}$. Then, the $\mathbf{SLG}(\mathcal{O})$ evaluation of a query in $\mathcal{K}_G^d$ is decidable with data complexity $\mathrm{P}^{\mathcal{C}}$.*

In particular, this means that if the partial oracle is tractable, and only with polynomial many results, then $\mathbf{SLG}(\mathcal{O})$ is also tractable. Clearly, for an ontology part of the knowledge base which is a tractable fragment, it is possible to come up with a correct partial oracle that is also tractable. Basically, all it needs to be done is to proceed with the usual entailment method, assuming that all program atoms hold, and collecting them for the oracle result. To guarantee that the number of solutions of the oracle is bound by a polynomial, and still keeping with correctness, might be a bit more difficult. It amounts to find a procedure that returns less results, and at the same time does not damage the completeness proof (similar to that of Theorem 4). At least for the tractable case this is possible, albeit the oracle being the (polynomial complexity) bottom-up procedure that defines $MKNF_{WFS}$.

6   Discussion and Conclusions

Together with the alternate computation method of Section 3, **SLG($\mathcal{O}$)** provides a sound and complete querying method for hybrid MKNF knowledge bases, that unlike others (cf. below) freely allows bidirectional calls between the ontology and the rules, and that does not impose a burden of complexity beyond that of the ontology. As such it presents a significant step towards making hybrid MKNF knowledge bases practically usable for the Semantic Web. In fact, work has begun on a prototype implementation of the **SLG($\mathcal{O}$)** method presented here using XSB Prolog and its ontology management library CDF [8]. Because the CDF theorem prover is implemented directly using XSB, the ORACLE RESOLUTION and EQUALITY RESOLUTION operations of Section 4 are more easily implemented than they would be using a separate prover, as is the detection of when a mutually dependent set of subgoals is completely evaluated (Definition 14), and the guarantee of the polynomial size of the oracle. The resulting implementation will enable further study into how hybrid MKNF knowledge bases can be practically used and will indicate needed optimizations. For instance, since XSB supports constraint processing, temporal or spatial constraints can be added to the ABox. From a systems perspective, the multi-threading of XSB can allow for the construction of hybrid MKNF knowledge servers that make use of either Prolog rules or F-logic rules (via FLORA-2, which is implemented using XSB). As mentioned in Section 5 the final forest of a **SLG($\mathcal{O}$)** evaluation produces a well-founded reduct of the rules and oracle information. This reduct, which is materialized in a table in XSB, can be sent to a stable model generator through XSB's XASP library to obtain a partial stable MKNF model of [12].

There are two other semantics which define a well-founded model for a combination of rules and ontologies, namely [5] and [3]. The approach of [5] combines ontologies and rules in a modular way, i.e. keeps both parts and their semantics separate, thus having similarities with our approach. The interface is done by the dlv hex system [4]. Though with identical data complexity to the well-founded MKNF semantics for a tractable DL, it has a less strong integration, having limitations in the way the ontology can call back program atoms (see [5] for details). Hybrid programs of [3] are even more restrictive in the combination: in fact it only allows to transfer information from the ontology to the rules and not the other way around. Moreover, the semantics of this approach differs from MKNF [12,9] and also[5] in that if an ontology expresses $B_1 \vee B_2$ then the semantics in [3] derives p from rules $p \leftarrow B_1$ and $p \leftarrow B_2$, p while MKNF and [5] do not.

While queries posed to KBs without an ontology are handled in the same way as in SLG, strictly speaking the queries posed to the (oracle) DL fragment, are not conjunctive queries in the sense of [7] where boolean queries may contain anonymous variables which never get instantiated. Here we ask whether a ground atom holds when querying the oracle. We nevertheless obtain conjunctive queries up to a certain extent in the sense of [7] only wrt. the entire KB, and our queries are not limited to fit the tree-shaped models there. One line of future work will thus be an extension to such queries which is supported by possible anonymous variables in XSB, the system in which the semantics is currently implemented.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications, 2nd edn. Cambridge University Press, Cambridge (2007)
2. Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. J. ACM 43(1), 20–74 (1996)
3. Drabent, W., Małuszynski, J.: Well-founded semantics for hybrid rules. In: Marchiori, M., Pan, J.Z., de Marie, C.S. (eds.) RR 2007. LNCS, vol. 4524, pp. 1–15. Springer, Heidelberg (2007)
4. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Effective integration of declarative rules with external evaluations for semantic web reasoning. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 273–287. Springer, Heidelberg (2006)
5. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Well-founded semantics for description logic programs in the semantic web. In: Antoniou, G., Boley, H. (eds.) RuleML 2004. LNCS, vol. 3323, pp. 81–97. Springer, Heidelberg (2004)
6. Gelfond, M., Lifschitz, V.: Logic programs with classical negation. In: Warren, Szeredi (eds.) ICLP 1990. MIT Press, Cambridge (1990)
7. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic SHIQ. J. of Artificial Intelligence Research 31, 151–198 (2008)
8. Gomes, A.S.: Derivation methods for hybrid knowledge bases with rules and ontologies. Master's thesis, Univ. Nova de Lisboa (2009)
9. Knorr, M., Alferes, J.J., Hitzler, P.: A coherent well-founded model for hybrid MKNF knowledge bases. In: Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N. (eds.) Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008, pp. 99–103. IOS Press, Amsterdam (2008)
10. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: International Joint Conferences on Artifical Intelligence, IJCAI 1991, pp. 381–386 (1991)
11. Lloyd, J.W.: Foundations of Logic Programming. Springer, Berlin (1984)
12. Motik, B., Rosati, R.: A faithful integration of description logics with logic programming. In: 20th Int. Joint Conf on Artificial Intelligence (IJCAI), Hyderabad, India, January 6–12 2007, pp. 477–482. AAAI Press, Menlo Park (2007)
13. Pearce, D., Wagner, G.: Reasoning with negative information I: Strong negation in logic programs. In: Haaparanta, L., Kusch, M., Niiniluoto, I. (eds.) Language, Knowledge and Intentionality, Acta Philosophica Fennica, vol. 49, pp. 430–453 (1990)
14. Pereira, L.M., Alferes, J.J.: Well founded semantics for logic programs with explicit negation. In: European Conference on Artifical Intelligence, ECAI, pp. 102–106 (1992)
15. Rosati, R.: On conjunctive query answering in EL. In: DL 2007.CEUR Electronic Workshop Proceedings (2007)
16. Sagonas, K., Swift, T., Warren, D.S.: The limits of fixed-order computation. Theoretical Computer Science 254(1-2), 465–499 (2000)
17. Swift, T., Pinto, A., Pereira, L.: Incremental answer completion. In: Hill, P.A., Warren, D.S. (eds.) ICLP 2009. LNCS, vol. 5649, pp. 519–524. Springer, Heidelberg (2009)
18. van Gelder, A.: The alternating fixpoint of logic programs with negation. In: Principles of Database Systems, pp. 1–10. ACM Press, New York (1989)
19. van Gelder, A., Ross, K.A., Schlipf, J.S.: Unfounded sets and well-founded semantics for general logic programs. Journal of the ACM 38(3), 620–650 (1991)