A Goal-Directed Implementation of Query Answering for Hybrid MKNF Knowledge Bases

Ana Sofia Gomes, José Júlio Alferes and Terrance Swift
CENTRIA, Departamento de Informática
Faculdade Ciências e Tecnologias
Universidade Nova de Lisboa
2829-516 Caparica, Portugal

submitted 1 June 2010; revised 18 October 2010; accepted 10 March 2011

Abstract

Ontologies and rules are usually loosely coupled in knowledge representation formalisms. In fact, ontologies use open-world reasoning while the leading semantics for rules use non-monotonic, closed-world reasoning. One exception is the tightly-coupled framework of Minimal Knowledge and Negation as Failure (MKNF), which allows statements about individuals to be jointly derived via entailment from an ontology and inferences from rules. Nonetheless, the practical usefulness of MKNF has not always been clear, although recent work has formalized a general resolution-based method for querying MKNF when rules are taken to have the well-founded semantics, and the ontology is modeled by a general oracle. That work leaves open what algorithms should be used to relate the entailments of the ontology and the inferences of rules. In this paper we provide such algorithms, and describe the implementation of a query-driven system, *CDF-Rules*, for hybrid knowledge bases combining both (non-monotonic) rules under the well-founded semantics and a (monotonic) ontology, represented by a CDF Type-1 ($\mathcal{ALCQ}$) theory.

KEYWORDS: Knowledge representation, well-founded semantics, description logics, implementation.

1 Introduction

Ontologies and rules offer distinctive strengths for the representation and transmission of knowledge over the Semantic Web. Ontologies offer the deductive advantages of first-order logics with an open domain while guaranteeing decidability. Rules employ non-monotonic (closed-world) reasoning that can formalize scenarios under locally incomplete knowledge; rules also offer the ability to reason about fixed points (e.g. reachability) which cannot be expressed within first-order logic.

Example 1
Consider a scenario application for a Customs House where an ontology is used to assess and classify aspects of imports and exports. An ontology with such characteristics would embrace several thousand axioms. In these axioms, as an example,

one can define several axioms about countries (e.g. Scandinavian countries are European countries; Norway is a Scandinavian country; etc). Moreover, one can also model in the ontology knowledge that is certain, in the sense that it does not allow for exception. In such certain knowledge, first-order logics deduction is what is desired. For example, in this scenario, one could state that Scandinavian countries are always considered safe countries. All of this can be easily represented by an ontology defined in a Description Logic (DL) language (Baader et al. 2007)[1] (1).

$$\begin{array}{l} ScandinavianCountry \sqsubseteq EuropeanCountry \\ Norway : ScandinavianCountry \\ ScandinavianCountry \sqsubseteq SafeCountry \end{array} \tag{1}$$

Besides these axioms about countries, one may want to specify some additional knowledge, e.g. for defining conditions about whether or not to inspect entering shipments based on their country of origin. For example, we may want to state that one should inspect any vessel containing a shipment coming from a country that is not guaranteed to be safe. Note here that the closed world assumption is needed to define such a statement. In fact, intuitively one wants in general to assume that, if one is unsure about a given country being safe or not, then (at least for the sake of this statement) we should consider that it is not, and proceed with the inspection. By using description logic, with classical negation, this behavior would not be obtained, as a condition $\neg SafeCountry(country)$ would only be true in case one knows for sure that the country is not safe. However, this statement can be easily expressed by using a (non-monotonic) rule with default negation[2] as in (2).

$$inspect(X) \leftarrow hasShipment(X, Country), \mathbf{not}\ safeCountry(Country). \tag{2}$$

Rules like this one are non-monotonic in the sense that further knowledge, in this case about safety of countries, can invalidate previous conclusions about inspection. This kind of non-monotonic rule is quite useful to specify default knowledge that may be subject to exceptions. In this sense rule (2) can be seen as stating that by default shipments should be inspected, but an exception to this default rule are shipments coming from safe countries.

Note that, in general, having defined some default statements about a predicate does not necessarily mean that *all* statements defining that predicate are default rules. For example, we may want to state that diplomatic shipments are not subject to inspection, regardless their country of origin. Here no default reasoning is involved. Moreover, negation here is not the default negation of logic programming,

[1] We write DL formulas with the usual notation and the usual DL operators (see (Baader et al. 2007)), and where the argument variable of a unary predicate is not displayed, and the first letter of the predicate's name is capitalized.

[2] Here we write rules in the usual notation for logic programming, where predicates names are non-capitalized (even if common to the ontology, where they are capitalized), and variables are capitalized and implicitly universally quantified in the rule.

but rather classical negation. Having the possibility to use an ontology, formalized as a fragment of first-order logic, it is easy to express such a statement:

$$DiplomaticShipment \sqsubseteq \neg Inspect \tag{3}$$

This example briefly illustrates some advantages of combining features of ontologies with features of logic programming-like rules, and of doing so in a way that allows knowledge about instances to be fully inter-definable between rules and an ontology (as happens with the predicate *inspect*/1 above, which is both defined in the rules and the ontology).

In fact, this combination of ontology and rule languages gained particular importance in the context of the Semantic Web (Horrocks et al. 2006). In this context, a family of languages for representing background knowledge in the Web, OWL 2 (Hitzler et al. 2009), has been recommended by the World Wide Web Consortium (W3C). OWL 2 languages are based on Description Logics (Baader et al. 2007), which, as in the example above, employ open-world assumption. In addition, the rule interchange language RIF (Morgenstern et al. 2010) has recently been formally recommended to the W3C. RIF has many similarities with the logic programming language used in the example above, and adopts the closed-world assumption.

The existence of both rules and ontology languages should make it possible to combine open and closed world reasoning, and this combination is indeed important in several domains in the Semantic Web. As a further example where this combination is desired, consider the large case study described in (Patel et al. 2007), containing millions of assertions about matching patient records for clinical trials criteria. In this case study, open world reasoning is needed in deductions about domains such as radiology and laboratory data: unless a lab or radiology test asserts a negative finding no arbitrary assumptions about the results of the test can be made (e.g. we can only be certain that some patient does not have a specific kind of cancer if the corresponding test is known to have negative result). However, as observed in (Patel et al. 2007), closed world reasoning can and should be used with pharmacy data to infer that a patient is not on a medication if this is not asserted. The work of (Patel et al. 2007) applies only open world reasoning but claims that the usage of closed world reasoning in pharmacy data would be highly desirable. Similar situations occur e.g. in matchmaking using Semantic Web Services (cf. (Grimm and Hitzler 08)), where again a combination of ontology languages relying on open-world reasoning, with rule languages relying on closed-world reasoning is considered highly desirable.

Several factors influence the decision of how to combine rules and ontologies into a hybrid knowledge base. The choice of semantics for the rules, such as the answer-set semantics (Gelfond and Lifschitz 1990) or the well-founded semantics (WFS) (van Gelder et al. 1991), can greatly affect the behavior of the knowledge base system. The answer-set semantics offers several advantages: for instance, description logics can be translated into the answer-set semantics providing a solid basis for combining the two paradigms (Baral 2002; Eiter et al. 2004a; Swift 2004; Motik 2006). On the other hand, WFS is weaker than the answer-set semantics (in the sense that it is more skeptical), but has the advantage of a lower complexity, and that it can be evaluated in a query-oriented fashion and so has been integrated into Prolog systems.

Several formalisms have concerned themselves with combining ontologies with WFS rules (Drabent and Małuszynski 2007; Eiter et al. 2004b; Knorr et al. 2008). Among these, the Well-Founded Semantics for Hybrid MKNF knowledge bases ($MKNF_{WFS}$), introduced in (Knorr et al. 2008) 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. $MKNF_{WFS}$ assigns a well-founded semantics to Hybrid MKNF knowledge bases, is sound w.r.t. the original semantics of (Motik and Rosati 2007) and, as in (Motik and Rosati 2007), allows the knowledge base to have both closed- and open-world (classical) negation.

Example 2

The following fragment, adapted from an example in (Motik and Rosati 2007), concerning car insurance premiums illustrates several properties of $MKNF_{WFS}$. The ontology consists of the following axioms, which state that $Married$ and $NonMarried$ are complementary concepts, that anyone who is not married is high risk, and that anyone with a spouse is married:

$$\begin{aligned} NonMarried &\equiv \neg Married \\ \neg Married &\sqsubseteq HighRisk \\ \exists Spouse.\top &\sqsubseteq Married. \end{aligned}$$

The rule base consists of the following rules, which state that anyone who is not known to be married is to be assumed as being non-married, and that those who are known to be high-risk should have a surcharge:

$$\begin{aligned} &\mathbf{K}\ nonMarried(X) \leftarrow \mathbf{K}\ person(X), \mathbf{not}\ married(X). \\ &\mathbf{K}\ surcharge(X) \leftarrow \mathbf{K}\ highRisk(X), \mathbf{K}\ person(X). \end{aligned}$$

Note that *married* and *nonMarried* are defined both by axioms in the ontology and by rules. Within the rule bodies, literals with the **K** or **not** operators (e.g. $\mathbf{K}\ highRisk(X)$) may require information both from the ontology and from other rules; other literals are proven directly by the other rules (e.g. *person(X)*). Intuitively, $\mathbf{K}\varphi$ stands for "φ is known to be true", whereas **not** φ stands for "φ is not known to be true".

Suppose *person(john)* were added as a fact (in the rule base). Under closed-world negation, the first rule would derive *nonMarried(john)*. By the first ontology axiom, $\neg$*married(john)* would hold, and by the second axiom *highRisk(john)* would also hold. By the last rule, $surcharge(john)$ would hold as well. Thus the proof of $surcharge(john)$ involves interdependencies between the rules with closed-world negation, and the ontology with open-world negation. At the same time the proof of $surcharge(john)$ is *relevant* in the sense that properties of other individuals, not related to *john* either through rules nor axioms in the ontology, do not need to be considered.

In the original definition of $MKNF_{WFS}$, the inter-dependencies of the ontology and rules were captured by a bottom-up fixed-point operator with multiple levels of iterations. Recently, a query-based approach to hybrid MKNF knowledge bases, called SLG($\mathcal{O}$), has been developed using tabled resolution (Alferes et al. 2009). SLG($\mathcal{O}$) is sound and complete, as well as terminating for various classes of programs (e.g. datalog). In addition SLG($\mathcal{O}$) is relevant in the sense of Example 2, i.e. in general one does not need to compute the whole model (for every object in the knowledge base) to answer a specific query. This relevancy is a critical requirement for scalability in numerous practical applications (e.g. in the area of Semantic Web): without relevance a query about a particular individual I may need to derive information about other individuals even if those individuals were not connected to I through rules or axioms. This is also clear e.g. in the above mentioned case study about matching patient records for clinical trials criteria (Patel et al. 2007): where one is interested in finding out whether a given patient matches the criteria for a given trial or, at most, which patients match a given criteria; for scalability, it is crucial that such a query does not need to go over all patients and all criteria. In a similar manner, when assessing a shipment in Example 1, it is unfeasible to have to consider all shipments into a country: the query must be relevant to be useful.

SLG($\mathcal{O}$) serves as a theoretical framework for query evaluation of $MKNF_{WFS}$ knowledge bases, but it models the inference mechanisms of an ontology abstractly, as an oracle. While this abstraction allows the resolution method to be parameterized by different ontology formalisms in the same manner as $MKNF_{WFS}$, it leaves open details of how the ontology and rules should interact and these details must be accounted for in an implementation.

This paper describes, in Section 4, the design and implementation of a prototype query evaluator[3] for $MKNF_{WFS}$, called *CDF-Rules*, which fixes the ontology part to $\mathcal{ALCQ}$ theories, and makes use of the prover from XSB's ontology management library, the Coherent Description Framework (CDF) (Swift and Warren 2003) (overviewed in Section 3). To the best of our knowledge, this implementation is the first working query-driven implementation for Hybrid MKNF knowledge bases, combining rules and ontology and complete w.r.t. the well-founded semantics.

2 MKNF Well-Founded Semantics

Hybrid MKNF knowledge bases, as introduced in (Motik and Rosati 2007), are essentially formulas in the logics of minimal knowledge and negation as failure (MKNF) (Lifschitz 1991), i.e. first-order logics with equality represented as the congruence relation $\approx$, and two modal operators **K** and **not**, which allow inspection of the knowledge base. Intuitively, given a first-order formula φ, $\mathbf{K}\varphi$ asks whether

[3] The implementation is available from the XSB CVS repository as part of the CDF package in the subdirectory `packages/altCDF/mknf`.

φ is known, while $\mathbf{not}\varphi$ is used to check whether φ is not known. A Hybrid MKNF knowledge base consists of two components: a decidable description logic (DL) knowledge base, translatable into first-order logic; and a finite set of rules of modal atoms.

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, along with finitely many individuals N_I and predicates (also called concepts) N_C. An atom $P(t_1, \dots, t_n)$, 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 is of the form

$$\mathbf{K}H \leftarrow \mathbf{K}A_1, \dots, \mathbf{K}A_n, \mathbf{not}B_1, \dots, \mathbf{not}B_m \tag{4}$$

where H_i, A_i, and B_i are atoms. H is called the *(rule) head* and the sets $\{\mathbf{K}A_i\}$, and $\{\mathbf{not}B_j\}$ form the *(rule) body*. Atoms of the form $\mathbf{K}A$ are also called *positive literals* or *modal* $\mathbf{K}$*-atoms* while atoms of the form $\mathbf{not}A$ are called *negative literals* or *modal* $\mathbf{not}$*-atoms*. 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 $(\mathcal{O}, \mathcal{P})$.

In Definition 1, the modal operators of MKNF logics are only applied to atoms in the rules, which might indicate that it would suffice to use a simpler logic that does not deal with the application of the modal operators to complex formulae. However, for defining the meaning of these hybrid knowledge bases, modal operators are also applied to more complex formulas, namely to the whole ontology $\mathcal{O}$ (cf. Definition 7). Intuitively, a hybrid knowledge base specifies that: atom H is known if all the A_i are known and none of the B_j is known for each rule of the form (4) in $\mathcal{P}$; and that the whole ontology $\mathcal{O}$ is known (i.e. $\mathbf{K}\mathcal{O}$). As such, using a more general logic, like MKNF, eases the definition of the semantics of hybrid knowledge bases so, similarly to what is done in (Motik and Rosati 2007) and (Knorr et al. 2008), we resort to the full MKNF logic (Lifschitz 1991).

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 (for more details we refer to (Motik and Rosati 2007)).

The well-founded MKNF semantics, *MKNF*$_{WFS}$, presented in (Knorr et al. 2008), is based on a complete three-valued extension of the original MKNF semantics. However here, as we are only interested in querying for literals and conjunctions of literals, we limit ourselves to the computation of what is called the well-founded partition in (Knorr et al. 2008): basically the atoms that are true or false. For that reason, and in correspondence to logic programming, we name this partition the well-founded model. First, we recall some notions from (Knorr et al. 2008) which will be useful in the definition of the operators for obtaining that well-founded model.

Definition 2
Consider a hybrid MKNF knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$. The *set of* $\mathbf{K}$*-atoms of* $\mathcal{K}$, written $\mathsf{KA}(\mathcal{K})$, is the smallest set that contains (i) all modal $\mathbf{K}$-atoms occurring in $\mathcal{P}$, and (ii) a modal atom $\mathbf{K}\xi$ for each modal atom $\mathbf{not}\xi$ occurring in $\mathcal{K}$.

Furthermore, for a set of modal atoms S, S_{DL} is the subset of DL-atoms of S (Definition 1), and $\widehat{S} = \{\xi \mid \mathbf{K}\xi \in S\}$.

Basically $\mathsf{KA}(\mathcal{K})$ collects all modal atoms of predicates appearing in the rules, and $\widehat{S}$ just removes $\mathbf{K}$ operators from the argument set S.

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 a program transformation making these new modal atoms available for reasoning with the respective rules.

Definition 3
Let $\mathcal{K}$ be a DL-safe hybrid MKNF knowledge base. We obtain $\mathcal{K}^+$ 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.

In $\mathcal{K}^+$, 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 cannot be proved[4]. For example, to guarantee in Example 1 that proving falsity of $Inspect$ for some shipment (in the ontology) enforces default negation of $Inspect$ for that shipment, one would build $\mathcal{K}^+$ by adding to the ontology the axiom $\neg Inspect \sqsubseteq N\_Inspect$ (where $N\_Inspect$ is a new symbol not appearing elsewhere), and in $\mathcal{K}^*$ all original rules with $inspect(X)$ in the head would be transformed by adding $\mathbf{not}$ $n\_inspec(X)$ to the body. In this case, the rule in (2) would be transformed to:

$$\begin{aligned}\mathbf{K}\ inspect(X) \leftarrow\ & \mathbf{K}\ hasShipment(X, Country), \mathbf{not}\ safeCountry(Country),\\ & \mathbf{not}\ n\_inspect(X).\end{aligned}$$

We continue by recalling the definition in (Knorr et al. 2008) of an operator $T_{\mathcal{K}}$ to allow conclusions to be drawn from positive hybrid MKNF knowledge bases (i.e. knowledge bases where rules have no default negation).

Definition 4

[4] 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.

For $\mathcal{K}$ a positive DL-safe 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:

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

$R_{\mathcal{K}}$ derives consequences from the rules in a way similar to the classical T_P operator of definite logic programs, while $D_{\mathcal{K}}$ obtains knowledge from the ontology $\mathcal{O}$, both from non-DL-atoms and the equalities occurring in $\mathcal{O}$, where the $\approx$ operator defines a congruence relation between individuals.

The operator $T_{\mathcal{K}}$ is shown to be monotonic in (Knorr et al. 2008) so, by the Knaster-Tarski theorem, it has a unique least fixed point, denoted $\mathsf{lfp}(T_{\mathcal{K}})$, which is reached after a finite number of iteration steps.

The computation of the well-founded models follows the alternating fixed point construction (van Gelder 1989) of the well-founded semantics for logic programs. This approach requires turning a hybrid MKNF knowledge base into a positive one to make $T_{\mathcal{K}}$ applicable.

Definition 5
Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground DL-safe 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 $\mathbf{K}H \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n$ 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 $1 \leq j \leq m$.

The above transformation resembles that used for answer-sets (Gelfond and Lifschitz 1990) of logic programs and the following two operators are defined.

Definition 6
Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a DL-safe hybrid MKNF knowledge base and $S \subseteq \mathsf{KA}(\mathcal{K}^*)$. We define:

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

Both operators are shown to be antitonic (Knorr et al. 2008), hence their composition is monotonic and form the basis for defining the well-founded MKNF model. Here we present its alternating computation.

$$\begin{array}{rlrl}
\mathbf{T}_0 = & \emptyset & \mathbf{TU}_0 = & \mathsf{KA}(\mathcal{K}^*) \\
\mathbf{T}_{n+1} = & \Gamma_{\mathcal{K}}(\mathbf{TU}_n) & \mathbf{TU}_{n+1} = & \Gamma'_{\mathcal{K}}(\mathbf{T}_n) \\
\mathbf{T}_\omega = & \bigcup \mathbf{T}_n & \mathbf{TU}_\omega = & \bigcap \mathbf{TU}_n
\end{array}$$

Note that by finiteness of the ground knowledge base the iteration stops before reaching ω. It was shown in (Knorr et al. 2008) that the sequences are monotonically increasing, decreasing respectively, and that $\mathbf{T}_\omega$ and $\mathbf{TU}_\omega$ form the well-founded model in the following sense:

Definition 7
Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a DL-safe hybrid MKNF knowledge base and let $\mathbf{T}_\mathcal{K}, \mathbf{TU}_\mathcal{K} \subseteq \mathsf{KA}(\mathcal{K})$, with $\mathbf{T}_\mathcal{K}$ being $\mathbf{T}_\omega$ and $\mathbf{TU}_\mathcal{K}$ being $\mathbf{TU}_\omega$ both restricted to the modal atoms only occurring in $\mathsf{KA}(\mathcal{K})$. Then

$$M_{WF} = \{\mathbf{K}A \mid A \in \mathbf{T}_\mathcal{K}\} \cup \{\mathbf{K}\pi(\mathcal{O})\} \cup \{\mathbf{not}A \mid A \in \mathsf{KA}(\mathcal{K}) \setminus \mathbf{TU}_\mathcal{K}\}$$

is the *well-founded MKNF model* of $\mathcal{K}$, where $\pi(\mathcal{O})$ denotes the first-order logic formula equivalent to the ontology $\mathcal{O}$ (for details on the translation of $\mathcal{O}$ into first-order logic see (Motik and Rosati 2007)) .

All modal **K**-atoms in M_{WF} are true, all modal **not**-atoms are false and all other modal atoms from $\mathsf{KA}(\mathcal{K})$ are undefined.

As shown in (Knorr et al. 2008), the well-founded model is sound with respect to the original semantics of (Motik and Rosati 2007), i.e. all atoms true (resp. false) in the well-founded model are also true (resp. false) according to (Motik and Rosati 2007). In fact, the relation between the semantics of (Knorr et al. 2008) and (Motik and Rosati 2007), is tantamount to that of the well-founded semantics and the answer-sets semantics of logic programs. Moreover, this definition is in fact a generalization of the original definition of the well-founded semantics of normal logic programs, in the sense that if the ontology is empty then this definition exactly yields the well-founded models according to (van Gelder et al. 1991). For more properties, as well as motivation and intuitions on $MKNF_{WFS}$, the reader is referred to (Knorr et al. 2008).

3 XSB Prolog and the Coherent Description Framework

Our implementation of $MKNF_{WFS}$ makes use of XSB Prolog (`xsb.sourceforge.net`) for two reasons. First, XSB's tabling engine evaluates rules according to WFS, and ensures rule termination for programs and goals with the *bounded term-size property* [5]. Second, the implementation uses the prover from XSB's ontology management system, the Coherent Description Framework (CDF) (Swift and Warren 2003).

CDF has been used in numerous commercial projects, and was originally developed as a proprietary tool by the company XSB, Inc [6]. Since 2003, CDF has been used to support extraction of information about aircraft parts from free-text data fields, about medical supplies and electronic parts from web-sites and electronic catalogs, and about the specifics of mechanical parts from scanned technical drawings. Also, CDF is used to maintain screen models for graphical user interfaces that are driven by XSB and its graphics package, XJ (`www.xsb.com/xj.aspx`). We discuss features of CDF that are relevant to the implementation described in Section 4.

[5] Intuitively, a program P and goal G have the bounded term-size property if there is a finite number n such that all subgoals and answers created in the evaluation of the goal G to P have a size less than n.

[6] Most of CDF is open-source, including all features used in this paper. CDF is distributed as a package in XSB's standard release, and full details can be found in its accompanying manual.

$Man \sqsubseteq Person \sqcap Male$

```
isa(cid(man),cid(person))
isa(cid(man),cid(male))
```

$Husband \sqsubseteq Man \sqcap \exists Spouse.Person$

```
isa(cid(husband),cid(man)).
hasAttr(cid(husband),rid(spouse),cid(person)
```

$adam : Husband$

```
isa(oid(adam),cid(husband)
```

Fig. 1. Some DL Axioms and their Type-0 Counterparts

Type-0 and Type-1 Ontologies All classes in CDF are represented by terms of the form *cid(Identifier, Namespace)*, instances by terms of the form *oid(Identifier, Namespace)*, and relations by terms of the form *rid(Identifier, Namespace)*, where *Identifier* and *Namespace* can themselves be any ground Prolog term.

Commercial use has driven CDF to support efficient query answering from Prolog for very large knowledge bases. A key to this is that ontologies in CDF can have a restricted, tractable form. *Type-0* ontologies do not allow representation of negation or disjunction within the ontology itself, and implicitly use the closed-world assumption. As such, Type-0 ontologies resemble a frame-based representation more than a description logic, and do not add any complexity to query evaluation beyond that of WFS. Support for query answering motivates the representation of Type-0 ontologies. The predicate $isa/2$ is used to state inclusion: whether the inclusion is a subclass, element of, or subrelation depends on the type of the term (not all combinations of types of terms are allowed in a CDF program). Relational atoms in CDF have the form:

- $hasAttr(Term_1, Rel_1, Term_2)$ which has the meaning $Term_1 \sqsubseteq \exists Rel_1.Term_2$;
- $allAttr(Term_1, Rel_1, Term_2)$ with the meaning $Term_1 \sqsubseteq \forall Rel_1.Term_2$;
- along with other forms that designate cardinality constraints on relations.

Figure 1 presents some DL Axioms and their Type-0 counterparts, where namespace information has been omitted for readability. The fact that Type-0 ontologies cannot express negation is crucial to their ability to be directly queried. In other words, `isa/2`, `hasAttr/3` and other atoms can be called as Prolog goals, with any instantiation pattern for the call. Query answering for Type-0 goals checks inheritance hierarchies and does not rely on unification alone. Type-0 ontologies use tabling to implement inheritance and use tabled negation so that only the most specific attribute types for a `hasAttr/3` or other query are returned to a user.

Besides the constructs of Type-0 ontologies, Type-1 ontologies further allow atoms of the form

$$necessCond(Term_1, CE)$$

where CE can be any $\mathcal{ALCQ}$ class expression over CDF terms. For instance, the axiom

$$Woman \sqsubseteq Person \sqcap \neg Man$$

would be represented by the atom

```
necessCond(cid(woman),(cid(person),not(cid(man))))
```

where the comma represents conjunction, as in Prolog. Because they use open-world negation, atoms for Type-1 ontologies cannot be directly queried; rather they are queried through goals such as $allModelsEntails(Term, ClassExpr)$, succeeding if $Term \sqsubseteq ClassExpr$ is provable in the current state of the ontology. Type-1 ontologies deduce entailment using a tableau prover written in Prolog.

System Features of CDF Regardless of the type of the ontology, atoms such as $isa/2$, $hasAttr/2$, etc. can be defined extensionaly via Prolog atoms, or intensionaly via Prolog rules. For instance, evaluation of the goal

```
hasAttr(Class1,Rel,Class2)
```

would directly check extensional Prolog facts through a subgoal

```
hasAttr_ext(Class1,Rel,Class2)
```

and would also check intensional rules through a subgoal

```
hasAttr_int(Class1,Rel,Class2)
```

Intensional definitions are used so that atoms can be lazily defined by querying a database or analyzing a graphical model: their semantics is outside that of CDF. In fact, using combinations of rules and facts, Type-0 ontologies are commonly used comprising tens of thousands of classes and relations, and tens of millions of individualss. At the same time, intensional definitions in a Type-1 ontology provide a basis for the tableau prover to call rules, as is required to support the interdependencies of $MKNF_{WFS}$, and will be further discussed in Section 4.2.

Despite their limitations, the vast majority of knowledge used by XSB, Inc. is maintained in Type-0 ontologies. Although Type-0 ontologies have supported numerous commercial projects, their limitations of course preclude the full use of information in ontologies. Support of a uniform querying mechanism for individualss in $MKNF_{WFS}$ as described below is intended as a means to allow commercial projects to use a more powerful form of knowledge representation.

Related Work

CDF was originally developed in 2002-2003, and Type-0 ontologies were envisioned as a means to represent object-oriented knowledge in Prolog. Unlike Flora-2 (Yang et al. 2003) CDF was intended to be a Prolog library, and its inheritance was designed to be entirely monotonic for compatibility with description logics. Type-1 ontologies were originally evaluated through a translation into ASP (Swift 2004): this approach pre-dated that of KAON2 (Motik 2006) and was abandoned due to the difficulty of dynamically pruning search in ASP; afterwards the current tableau approach was developed which attempts to examine as small a portion of the ontology as possible when proving entailment (cf. e.g. (Horrocks and Patel-Schneider 1999) for a discussion of how search may be pruned in tableau provers for ontologies). CDF's approach may be distinguished from (Lukacsy et al. 2008), which also combines ontological deduction with Prolog. Type-0 ontologies rely on WFS reasoning and so achieve good scalability under a weak semantics; theorem proving for Type-1 ontologies is used only when needed; (Lukacsy et al. 2008) takes a more uniform approach to deduction which relies on WAM-level extensions for efficiency; to our knowledge this approach is research-oriented, and has not been used commercially.

4 Goal-Driven MKNF Implementation

In Section 2 we presented a bottom-up computation that constructs the complete well-founded model for a given hybrid knowledge base. However, in practical cases, especially when considering the context of the Semantic Web, this is not what is intended. In fact, it would make little sense to compute the whole model of anything that is related to the *World Wide* Web. Instead, one would like to query the knowledge base for a given predicate (or propositional atom) and determine its truth value. As an illustration, recall Example 1 where we wanted to know if a given shipment should be inspected or not when it arrived, or in the case study of (Patel et al. 2007) where one usually wants to know whether a given patient matches the criteria for a given trial. Deriving all the consequences of the knowledge base to answer a query about a given shipment, or about a given patient, would be impractical.

In this section we describe the algorithms and the design of *CDF-Rules*, a goal-driven implementation for Hybrid MKNF Knowledge Bases under the Well-Founded Semantics that minimizes the computation to the set of individuals that are relevant to a query. *CDF-Rules* makes use of XSB's SLG Resolution (Chen and Warren 1996) for the evaluation of a query, together with tableaux mechanisms supported by CDF theorem prover to check entailment on the ontology. *CDF-Rules* is tuned for Type-1 ontologies, and thereby is also compatible with Type-0 ontologies. For the description of the solution, we assume that the reader has a general knowledge of tabled logic programs (cf. e.g. (Swift and Warren 2011)).

4.1 A Query-Driven Iterative Fixed Point

At an intuitive level, a query to *CDF-Rules* is evaluated in a relevant (top-down like) manner with tabling, through SLG resolution (Chen and Warren 1996), until the selected goal is a literal l formed over a DL-atom. At that point, in addition to further resolution, the ontology also uses tableau mechanisms to derive l. However, as a tableau proof of l may require propositions (literals) inferred by other rules, considerable care must be taken to integrate the tableau proving with rule-based query evaluation.

In its essence, a tableau algorithm decides the entailment of a formula f w.r.t. an ontology $\mathcal{O}$ by trying to construct a common *model* for $\neg f$ and $\mathcal{O}$, sometimes called a *completion graph* (cf. e.g. (Schmidt-Strauss and Smolka 1990)). If such a model can not be constructed, $\mathcal{O} \models f$; otherwise $\mathcal{O}$ does not entail f. Similar to other description logic provers, the CDF theorem prover attempts to traverse as little of an ontology as possible when proving f. As a result, when the prover is invoked on an atom A, the prover attempts to build a model for the underlying individual(s) to which A refers, and explores additional individuals only as necessary.

For our purposes, given the particular interdependence between the rules and the ontology in $MKNF_{WFS}$, the prover must consider the knowledge inferred by the rules in the program for the entailment proof, as a DL-atom can be derived by rules, which in turn may rely on other DL-atoms proven by the ontology. Thus, a query to a DL-atom $p(o)$, iteratively computes a (sub-)model for o, deriving at each iteration new information about the roles and classes of o, along with information about other individuals related to o either in the ontology (via CDF's tableau algorithm) or in the rules (via SLG procedures) until a fixed point is reached.

We start by illustrating the special case of positive knowledge bases without default negation in the rules.

Example 3
Consider the following KB (with the program on the left and the ontology on the right[7]) and the query $third(X)$:

$$\begin{array}{ll}
\mathbf{K}\ third(X) \leftarrow p(X), \mathbf{K}\ second(X). & \\
\mathbf{K}\ first(callback). & First \sqsubseteq Second \\
p(callback). &
\end{array}$$

The query resolves against the rule for $third(X)$, leading to the goals $p(X)$ and $\mathbf{K}\ second(X)$. The predicate p, although not a DL-atom, assures DL-safety, restricting the application of the rules to known individuals. The call $p(X)$ returns true for $X = callback$. Accordingly, the next subgoal is $third(callback)$ which depends on the DL-atom $second(callback)$, corresponding in the ontology to the proposition *Second*. At this point, the computation calls the CDF theorem prover which starts to derive a model for all the properties of the individual *callback*. Yet in this computation, the proposition *Second* itself depends on a predicate defined in the rules – $First$. It can thus be seen that the evaluation of the query $third(callback)$ must be done iteratively – the (instantiated) goal $third(callback)$ should suspend (using tabling) until $second(callback)$ is resolved. Furthermore, $second(callback)$ needs first to prove $first(callback)$ from the rules. In general, goals to DL-atoms may need to suspend in order to compute an iterative fixed point, after which they may either succeed or fail.

To formalize the actions in Example 3 on the special case of definite programs,

[7] To simplify reading, for rules we omit the **K** before non-DL atoms. In fact, in the implementation the ontology must be written according to CDF syntax, and in the rules the modal operators **K** and **not** are replaced by (meta-)predicates *known*/1 and *dlnot*/1, respectively (see Section 4.2).

we start by considering the computation for all individuals (i.e. temporarily disregarding the relevance of individuals, as discussed above).

Definition 8
Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a DL-safe hybrid MKNF knowledge base, where $\mathcal{P}$ does not contain default negation. Let $\mathcal{I}$ be a fixed set of individuals. The function *Tableau*($\mathcal{O}$) computes for a theory $\mathcal{O}$ the entailments of $\mathcal{O}$ for $\mathcal{I}$, disregarding the rules component. The function $SLG(\mathcal{P})$ computes via tabling the set of DL-atoms true in the minimal model of $\mathcal{P}$ for a set of individuals, $\mathcal{I}$, disregarding the ontology component. The model is obtained as the union of the least fixed point of the sequences:

$$\begin{array}{ll} D_0 = Tableau(\mathcal{O}) & R_0 = SLG(\mathcal{P}) \\ D_1 = Tableau(\mathcal{O} \cup R_0) & R_1 = SLG(\mathcal{P} \cup D_0) \\ \dots & \dots \\ D_n = Tableau(\mathcal{O} \cup R_{n-1}) & R_n = SLG(\mathcal{P} \cup D_{n-1}) \end{array}$$

Definition 8 resembles Definition 4 of the operator $T_{\mathcal{K}}$ in Section 2. In fact, the R_i sequence is similar to the $R_{\mathcal{K}}(S)$ operator which collects new conclusions from the rules, whereas the D_i sequence is similar to the $D_{\mathcal{K}}(S)$ operator which collects new conclusions from the ontology. Rather than starting with the empty set of conclusions from the rules, as is the case for $T_{\mathcal{K}}$ of Definition 4, here the R_i sequence starts with all conclusions that can be drawn from the program alone. However, since $T_{\mathcal{K}}$ is monotonic, it is clear that starting the iteration of this operator with all conclusions drawn from the rules alone would yield exactly the same (least) fixed point. Given these minor differences, taking the function *Tableau*($\mathcal{O}$) as correct w.r.t. the consequence relation of the description logic in use, and taking the function $SLG(\mathcal{P})$ as correct w.r.t. the least model semantics of definite logic programs, it is easy to see that if $\mathcal{I}$ is the set of all individuals in the knowledge base then the union of the D_n and R_n at the fixed point exactly coincides with the least fixed point of $T_{\mathcal{K}}$. Furthermore, as long as the program P respects DL-safety, MKNF rules are lazily grounded with respect to the set of individuals $\mathcal{I}$. In fact, given a DL-safe set of MKNF rules, and a set of queries grounded with the individuals in $\mathcal{I}$, the evaluation of the queries results in a complete grounding of the rules, and so the fixed point is guaranteed in a finite number of steps.

Definition 8 captures certain aspects of how the rules and the ontology use each other as a way to derive new knowledge in *CDF-Rules*, via an alternating computation between the rules and the ontology. However it does not capture cases in which the relevant set of individuals changes (i.e. it does not deal with changes in the set $\mathcal{I}$), or the presence of default negation in rule bodies. With regard to relevant individuals, since it is possible to define n-ary predicates in rules along with roles in the ontology, the query may depend on a set of several individuals. Therefore, the fixed point computation must take into account the entire set of individuals that the query depends on. This is done by tabling information about each individual in the set of individuals relevant to the query. This set may increase throughout the fixed point iteration as new dependency relations between individuals (including equality) are discovered. The iteration stops when it is not possible to derive anything else about these individuals, i.e., when both the set of individuals and the classes and roles of those individuals have reached a fixed point. The details of this iterative increase in the set of considered individuals can be found in the algorithm of Figure 2, which also deals with default negation and its interplay with first-order negation.

The following example considers the presence of default negation in rule bodies.

Example 4
Consider the following knowledge base:

$$\begin{array}{ll} \mathbf{K}\ third(X) \leftarrow p(X), \mathbf{K}\ second(X). & First \sqsubseteq Second \\ \mathbf{K}\ fourth(X) \leftarrow p(X), \mathbf{not}\ third(X). & Fourth \sqsubseteq Fifth \\ \mathbf{K}\ first(callback). & \\ p(callback). & \end{array}$$

In this example a predicate $fourth(X)$ is defined at the expense of the negation of $third(X)$. Since $fourth(X)$ is defined in the rules, the negation is *closed world*, that is, $fourth(X)$ should only succeed if it is not possible to prove $third(X)$. Consequently, if we employed SLG resolution blindly, an iteration where the truth of $second(callback)$ had not been made available to the rules from the ontology might mistakenly fail the derivation of $third(callback)$ and so succeed $fourth(callback)$. Likewise, the rules may pass to the ontology knowledge, that after some iterations, no longer applies — in this case if the ontology were told that $fourth(callback)$ was true, it would mistakenly derive $Fifth$.

Example 4 illustrates the need to treat default negation carefully, as the truth of default literals requires re-evaluation when new knowledge is inferred. Recall the manner in which the operators $\Gamma_{\mathcal{K}}$ and $\Gamma'_{\mathcal{K}}$ of Definition 6 address the problem of closed-world negation. Roughly, one step in $\Gamma_{\mathcal{K}}$ (or $\Gamma'_{\mathcal{K}}$) is defined as the application of $T_{\mathcal{K}}$ until reaching a fixed point. Applying $\Gamma'_{\mathcal{K}}$ followed by $\Gamma_{\mathcal{K}}$ is a monotonic operation and thus is guaranteed to have a least fixed point. In each dual application of $\Gamma_{\mathcal{K}}$ and $\Gamma'_{\mathcal{K}}$ two different models follow – a monotonically increasing model of true atoms (i.e. true predicates and propositions), and a monotonically decreasing model of non-false atoms.

In a similar way, the implementation of *CDF-Rules* makes use of two fixed points: an *inner* fixed point where we apply Definition 8 corresponding to $T_{\mathcal{K}}$; and an *outer* fixed point for the evaluation of **not**s, corresponding to $\Gamma_{\mathcal{K}}$ (and $\Gamma'_{\mathcal{K}}$). In the outer iteration, the evaluation of closed-world negation is made by a reference to the previous model obtained by $\Gamma_{\mathcal{K}}$. Thus in *CDF-Rules*, $\mathbf{not}(A)$ succeeds if, in the previous *outer* iteration, A was not proven.

Example 5
As an illustration of the need for the application of the two fixed points, consider the knowledge base below and the query $c(X)$:

$$\begin{array}{ll} \mathbf{K}\ c(X) \leftarrow p(X), \mathbf{K}\ a(X), \mathbf{not}\ b(X). & A \sqsubseteq B \\ \mathbf{K}\ a(object). & \\ p(object). & \end{array}$$

When evaluating the query $c(X)$, X is first bound to *object* by p, and then the nested iteration begins. The inner iteration follows the steps of Definition 8, and since these operators are defined only for definite rules, each negative body literal in a rule such as $p(X)$ is evaluated according to its value in the previous *outer* fixed point, or is simply evaluated as true in the first outer iteration. (As we will see, this is done lazily by *CDF-Rules*). The first stage of the inner iteration is to compute via the rules R_0, which contains $a(object)$, $p(object)$ and $c(object)$; and to compute via the ontology D_0, which is empty as $\mathcal{O} \not\models A$. In the second inner stage the rules achieve the same fixed point as in the first, so $R_1 = R_0$, but the ontology derives B for *object* in D_1. After sharing this knowledge, there is nothing else to infer by either components, and we achieve the first inner fixed point with:

$$T_1 = \{a(object), b(object), c(object), p(object)\}$$

So now, the second outer iteration will start the computation of the inner iteration again and, in this iteration, **not**s are evaluated w.r.t. T_1. As a consequence, $c(object)$ fails, since $b(object) \in T_1$. The fixed point of the second inner iteration contains $p(object)$, $a(object)$ and $b(object)$, which is in fact the correct model for *object*. Afterwards, a final outer iteration is needed to to determine that an outer fixed point has in fact been reached. Since $c(object)$ is in the model of the final iteration, the query $c(X)$ succeeds for $X = object$.

The procedure for a lazily invoked iterative fixed point described above is summarized in Figure 2 using predicates that are described in detail in Section 4.2. The tabled predicate *known*/3 is used in each inner iteration to derive knowledge from the rules component, while $allModelsEntails/3$ infers knowledge from the ontology via a tableau proof. Within rules evaluated by *known*/3, the default negation of a DL-atom A is obtained by the predicate $dlnot(A)$, which succeeds if A was not proven in the last outer iteration. Whenever a role is encountered for an individual, a check is made to determine whether the related individual i_r is already in the list of individuals in the fixed point, and i_r is added if not. The predicates $definedClass/2$ and $definedRole/3$ are used to obtain the relevant classes and roles defined for a given individual over a DL-safe MKNF Hybrid Knowledge Base. We assume that these predicates are defined explicitly by the compiler or programmer, but they can also be inferred via the DL-safe restriction. In fact, by bounding our program to DL-Safe rules, every rule in the hybrid knowledge base must contain a positive predicate that is only defined in the rules. This predicate limits the evaluation of the rules to known individuals, so that *CDF-Rules* can infer the set of individuals that are applicable to a given rule, that is, its *domain*.

The algorithm shown in Figure 2 creates two different sets corresponding to the application of the operator Γ of t $MKNF_{WFS}$ (Definition 6): a credulous set, TU, containing the atoms that are true or undefined; and a skeptical set, T, of the atoms that are true (cf. Definition 7). As in the application of Γ, the T set is monotonically increasing, whereas the TU set is monotonically decreasing. The predicate *known*/3 captures the construction of these sets. Particular to our implementation, the first iteration of *OutIter* in $known(Term, OutIter, InIter)$ cor-

```
Input: A query Query to a DL-Atom
Output: Value of the input query in MKNF_WFS
addIndividuals(Query,IndividualList);
foreach Individual in IndividualList do
    OutIter, InIter = 0;
    S = S1 = {};
    P = P1 = {};
    repeat
        P = P1;
        repeat
            S = S1;
            foreach Class in definedClass(Individual,Class) do
                Term = Class(Individual);
                S1 = S1∪ known(Term, OutIter, InIter);
                S1 = S1∪ allModelsEntails(Term, OutIter, InIter);
                S1 = S1∪ allModelsEntails(not Term, OutIter, InIter);
            end
            foreach Role in definedRole(Individual,Individual1,Role) do
                Term = Role(Individual,Individual1);
                add Individual1 to IndividualList if necessary
                S1 = S1∪ known(Term, OutIter, InIter);
                S1 = S1∪ allModelsEntails(Term, OutIter, InIter);
                S1 = S1∪ allModelsEntails(not Term, OutIter, InIter);
            end
            InIter++;
        until S = S1 ;
        P = S;
        OutIter++;
    until P = P1 ;
end
if known(Query,Outer_Final-1,Inner_Final) then
    return true
else
    if known(Query,Outer_Final,Inner'_Final) then
        return undefined
    else
        return false
    end
end
```

Fig. 2. The Top-Level Algorithm: *ComputeFixedPoint(Query)*

responds to the first step of computation of the set TU, whereas the second iteration corresponds to the second step of computation of the set T. As a consequence, odd iterations of the second argument of $known/3$ are monotonically decreasing TU iterations, whilst even iterations are monotonically increasing T iterations. Since we check for fixed points only in odd iterations the fixed point will only be achieved in TU iterations. This way, we say that $Query$ is true if $known(Query, Outer_{Final} - 1, Inner_{Final})$ holds. If this is not the case, $Query$ is said to be undefined if $known(Query, Outer_{Final}, Inner_{Final})$, and is false otherwise.

4.2 Implementing MKNF$_{WFS}$ *Components*

We now provide a description of the various predicates in the algorithm of Figure 2, discussing the manner in which the rule and ontology components exchange knowledge, and how the fixed point is checked.

4.2.1 Rules Component

As mentioned, inferences from rules are obtained using the predicates $known/1$ corresponding to **K** and $dlnot/1$ corresponding to **not**. As shown in Figure 3, given the goal $known(A)$ with $A = p(O)$, the code first calls $computeFixedPoint(p(O))$ which begins the fixed point computation for the object instance O. $computeFixedPoint/1$ was summarized in Figure 2 and calls the lower-level $known/3$ and $dlnot/3$ to determine the truth of a goal in previous iterations. Once the fixed point has been reached, the final iteration indices for O are obtained from a global store using *get_object_iter(p(O),Outer,Inner)*, and $known/3$ is called again to determine whether $p(O)$ in fact true in the current iteration. This post-fixed point call to $known/3$ simply checks the table, and so is not computationally expensive. $known/3$ is always called with the iteration indices (arguments 2 and 3) bound, and if $p(O)$ is true in the current iteration, the table entry will contain the iteration indices. Within a given iteration, $p(O)$ may be known if it can be directly derived from the rules; alternately, or if $O \in P$ was entailed by the the ontology in the previous inner iteration step, as determined by the call $allModelsEntails/3$, which checks for the entailment of $O \in P$ in the previous inner iteration. In both cases, care must be taken so that it is guaranteed that if $\neg A$ holds, then *not* A holds as well. In Definition 3 this is guaranteed by considering the addition of $\mathbf{not} NP$ in bodies of rules with head P in one of the alternating operators. Analogously, when we try to derive $known(A, OutIter, InIter)$ and the iteration $OutIter$ is even (i.e. corresponding in Definition 7 to a TU step where $\Gamma'_{\mathcal{K}}$, rather than $\Gamma_{\mathcal{K}}$, is being applied), we further check if the ontology derived $\neg A$ in the previous inner iteration via the predicate $no\_prev\_neg/3$. If $\neg A$ is derived, then $no\_prev\_neg/3$ fails via the call to $tnot/1$, which is XSB's operator for tabled negation, and the top-level goal also fails.

On the other hand, the predicate $dlnot(A)$ which uses closed world assumption, succeeds if A fails (Figure 4). As discussed in Example 5, the evaluation of $dlnot/2$ must take into account the result of the previous *outer* iteration. Accordingly, in Figure 4 the call $dlnot(A)$ with $A = p(O)$ gets the current outer iteration for O, and immediately calls $dlnot/2$. If the outer iteration number is greater than 1, the second clause of $dlnot/2$ simply finds the index of the fixed point of the previous outer iteration, and determines whether A was true in that fixed point. Since the call to $known/3$ in $tnot/1$ is tabled, none of these predicates for **not** need to be tabled themselves. As described before, each outer iteration represents an iteration in T and TU sets of Definition 7 for $MKNF_{WFS}$. As a result, T sets are monotonically

```
known(A):-
     computeFixedPoint(A),
     get_object_iter(A,OutIter,InIter),
     known(A,OutIter,InIter).

:- table known/3.
known(A,OutIter,InIter):-
  PrevIter is InIter - 1,
  (   call(A),
    ;
      InIter > 0,
      allModelsEntails(A,OutIter,PrevIter)
  ),
  ( OutIter mod 2 =:= 1 ->
        true
      ;
        no_prev_neg(A,OutIter, PrevIter)
  ).

/* Enforce coherence of default negation with first-order
    negation */
no_prev_neg(_A,_OutIter, PrevIter) :-
  PrevIter < 0,!.
no_prev_neg(A,OutIter, PrevIter) :-
  tnot(allModelsEntails(neg(A),OutIter,PrevIter)).
```

Fig. 3. Prolog Implementation of **K** for Class Properties

```
dlnot(A):-
     computeFixedPoint(A),
     get_object_iter(A,OutIter,_InIter),
     dlnot(A,OutIter).

/* In first iteration, ensure that TU = KA(K*) */
dlnot(_A,0):- !.
/* In subsequent iterations, check previous outer iteration */
dlnot(A,OutIter):-
        PrevIter is OutIter - 1,
        get_final_iter(A,PrevIter,FinIter),
        tnot(known(A,PrevIter,FinIter)).
```

Fig. 4. Prolog Implementation of **not** for Class Properties

increasing while TU sets are monotonically decreasing. To assure that the first TU set is the largest set ($\mathsf{KA}(\mathcal{K}^*)$ following Definition 6), we compel all calls to *dlnot*/1 to succeed in the first outer iteration, as represented by the first clause of *dlnot*/2.

```
:- table allModelsEntails/3.
allModelsEntails(not(Atom),_OutIter,_InIter):- !,
        /* transform Atom to CDF object identifier and
           class expression*/
        /* add individuals to current fixed point list */
        (rec_allModelsEntails(Id,CE) -> fail ; true).
allModelsEntails(Atom,_OutIter,_InIter):-
        /* transform Atom to CDF object identifier and
           class expression*/
        /* add individuals to current fixed point list */
        (rec_allModelsEntails(Id,not(CE)) -> fail ; true).
```

Fig. 5. Prolog Pseudo-code for $allmodelsEntails/3$

4.2.2 Ontology Component

The tabled predicate *allModelsEntails/3* provides the interface to CDF's tableau theorem prover (Figure 5). It is called with a DL-literal L and with the indices of its outer and inner iterations both bound. Although the iterations are not used in the body of *allModelsEntails/3*, representing the iteration information in the head ensures that tabled answers will contain the iteration information. This information is used when *allModelsEntails/3* is called by *known/1* (through *known/3*). *allModelsEntails/3* first converts the atomic form of a proposition to one used by CDF. That is, it translates a 1-ary DL-atom representing an individual's class membership to the CDF predicate $isa/2$, and a 2-ary DL-atom representing an individual's role to the CDF predicate $hasAttr/3$ (see Section 3). In addition, if *Atom* is a 2-ary role, the target individual may be added to the fixed point set of individuals. As is usual with tableau provers, entailment of L by an ontology $\mathcal{O}$ is shown if the classical negation of L is inconsistent with $\mathcal{O}$. Thus, *rec_alModelsEntails/2* immediately fails if the classical negation of L is consistent with $\mathcal{O}$ in the present iteration; otherwise, L is entailed.

The tableau prover, called by $rec\_allModelsEntails/2$, ensures that it obtains all information inferred by the rules during the previous inner iteration, in accordance with Definition 8. This is addressed via the CDF intensional rules. As discussed in Section 3, the architecture of a CDF instance can be divided into two parts – extensional facts and intensional rules. Extensional facts define CDF classes and roles as simple Prolog facts; intensional rules allow classes and roles to be defined by Prolog rules which are outside of the $MKNF_{WFS}$ semantics. In our case, the intensional rules support a programming trick to check rule results from a previous iteration. As shown in Figure 6 they convert the CDF form of an ontology axiom into a 1-ary or 2-ary predicate, and then check the $known/3$ table for a previous iteration using the predicate *lastKnown/1* (not shown). If roles or classes are uninstantiated in the call from the tableau prover, all defined roles and classes for the individual are instantiated using *definedClass/3* or *definedRole/4*, and then called using *lastKnown/1* against the previous (or final) iteration of the rules.

```
isa_int(oid(Obj,NS),cid(Class,NS1)):-
    ground(Obj),ground(Class),!,
    Call =.. [Class,Obj],          /* Call = Class(Obj) */
    lastKnown(Call).
/* Find all possible classes for Obj if called with superclass
    argument uninstantiated */
isa_int(oid(Obj,NS),cid(Class,NS)):-
    ground(Obj),var(Class),!,
    definedClass(Call,Class,Obj),
    lastKnown(Call).

hasAttr_int(oid(Obj1,NS),rid(Role,NS1),oid(Obj2,NS2)):-
    ground(Obj1), ground(Obj2), ground(Role),!,
    Call =.. [Role,Obj1,Obj2],     /* Call = Role(Obj1,Obj2) */
    last_known(Call).
/* Find all possible rules for Obj if called with role argument
     uninstantiated */
hasAttr_int(oid(Obj1,NS),rid(Role,NS1),oid(Obj2,NS2)):-
    ground(Obj1), ground(Obj2), var(Role),!,
    definedRole(Call,Role,Obj1,Obj2),
    last_known(Call).
```

Fig. 6. Callbacks from the ontology component to the rules component

4.2.3 Usage

An MKNF Hybrid Knowledge base is defined over a XSB-Prolog knowledge base together with an ontology specified over CDF. In *CDF-Rules* such a knowledge base is written into two files as follows:

- *rules.P* – containing the set of MKNF rules and facts. A rule is defined as standard Prolog rules as follows:

  $$\texttt{Head :- } A_1\texttt{, ... } A_k\texttt{,known(}B_1\texttt{),...,known(}B_n\texttt{),dlnot(}C_1\texttt{),...,dlnot(}C_m\texttt{).}$$

  where $k, n, m \geq 0$, and the A_is are all non-DL predicates (i.e. predicates that are not defined in the ontology), and the B_is and C_is are predicates that can be both defined in the rules and in the ontology. If $k = n = m = 0$ then the rule is a fact, and it is written as usual in Prolog, omitting the ':-' operator. Note that the transformation to include the negation *N_Head* in the body of a rule for *Head* as specified in Definition 3 is not needed: such a check is done by the call to `no_prev_neg/3` in `known/3`.
  To guarantee correctness, each rule must respect DL-safety. However, in the current implementation it is the programmer's responsibility to check for this condition. The current implementation also does not check that the A_i predicates (i.e. the ones not under *known*/1 or *dlnot*/1) are not defined in the ontology. If a programmer opts to not precede the predicate by *known*/1 or *dlnot*/1, any definition for the predicate in the ontology is simply ignored.

- *cdf_extensional.P* – comprising ordinary ontology facts and concepts defined over the CDF syntax.
- *cdf_intensional.P* – containing predicates allowing the ontology to access information in the rules as in Figure 6. In addition, the file may contain other intensional rules to lazily access information from a database, off of the semanitc web, or from other sources external to *CDF-Rules*.

Example 6
The knowledge base of Example 1 can be easily coded in *CDF-Rules* as:

```
%rules
inspect(X) :- hasShipment(X,C), dlnot(safeCountry(C)).

%cdf_extensional
isa_ext(cid(scandinavianCtr,ont),cid(safeCountry,ont)).
isa_ext(cid(scandinavianCtr,ont),cid(EuropeanCtr,ont)).
isa_ext(oid(norway,ont),cid(scandinavianCtr,ont)).
necesscond_ext(cid(DiplomaticShipt,ont),not(cid(inspect,ont))).
```

Note that the ontological portion is Type-1, due to the use of `necessCond/2`.

4.2.4 Discussion

As described, *CDF-Rules* implements query answering to hybrid MKNF knowledge bases, and tries to reduce the amount of relevance required in the fixed point operation. Relevance is a critical concept for query answering in practical systems, however a poorly designed ontology or rules component can work against one another if numerous individuals depend on one another through DL roles. In such a case the relevance properties of our approach will be less powerful; however in such a case, a simple query to an ontology about an individual will be inefficient in itself. The approach of *CDF-Rules* cannot solve such problems; but it can make query answering as relevant as the underlying ontology allows.

We do not present here a formal proof of soundness and completeness for our algorithm, since this would require the full presentation of the formal derivation procedures on which both XSB-Prolog and CDF implementations rely. However, we have given an informal argument along with our description by referring to complementarity between the implementation and the bottom-up definition of $MKNF_{WFS}$. In particular, there is a close correspondence between the core of our computation represented in Definition 8 and the Definition 4 from the 3-valued MKNF definition, as well as between the operators Γ / Γ' and our inner / outer fixed points. As a result, one can view our goal-driven implementation as an optimization of the bottom-up approach where the computation is limited to the set of *relevant* objects, and where the evaluation of positive predicates and the handling of iterations is performed by the use of SLG resolution.

Further optimizations of the described approach are possible. First is to designate a set of atoms whose value is defined *only* in the ontology: such atoms would require tableau proving, but could avoid the fixed point check of $computeFixedPoint/1$.

Within $computeFixedPoint/1$ another optimization would be to maintain dependencies among individuals. Intuitively, if individual I_1 depended on individual I_2 but not the reverse, a fixed point for I_2 could be determined before that of I_1. However, these optimizations are fairly straightforward elaborations of *CDF-Rules* as presented.

5 Conclusions

In this paper we have described the implementation of a query-driven system, *CDF-Rules*, for hybrid knowledge bases combining both (non-monotonic) rules and a (monotonic) ontology. The system answers queries according to $MKNF_{WFS}$ (Knorr et al. 2008) and, as such, is also sound w.r.t. the semantics defined in (Motik and Rosati 2007) for Hybrid MKNF knowledge bases. The definition of $MKNF_{WFS}$ is parametric on a decidable description logic (in which the ontology is written), and it is worth noting that, as shown in (Knorr et al. 2008), the complexity of reasoning in $MKNF_{WFS}$ is in the same class as that in the decidable description logic; a complexity result that is extended to a query-driven approach in (Alferes et al. 2009). In particular, if the description logic is tractable then reasoning in $MKNF_{WFS}$ is also tractable. Our implementation fixes the description logic part to CDF ontologies that, in its Type-1 version, supports $\mathcal{ALCQ}$ description logic. CDF Type-0 ontologies are simpler, and tractable and, when using Type-0 ontologies only, our implementation exhibits a polynomial complexity behavior. This fact derives from the usage of tabling mechanisms, as defined in SLG resolution and implemented in XSB Prolog[8] In fact, one of the reasons that highly influenced the choice of CDF as the parameter ontology logic in our query-driven implementation for Hybrid MKNF knowledge bases, was the very existence of an implementation of CDF relying on tabling, that could be coupled together with the tabling we needed for $MKNF_{WFS}$. But the algorithms presented here do not rely on particularities of CDF, and we believe that, for other choices of parameter logics, implementations could be made in a way similar to the one described in this paper. Of course, such an implementation would require first an implementation in XSB-Prolog of a prover for the other description logic of choice, providing at least a predicate $allModelsEntails/3$ with the meaning as described above.

Though our choice for the implementation was the Well-Founded Semantics for Hybrid MKNF knowledge bases, $MKNF_{WFS}$, (Knorr et al. 2008), there were other formalisms concerned with combining ontologies with WFS rules (Eiter et al. 2004b; Drabent and Małuszynski 2007). The approach of (Eiter et al. 2004b) combines ontologies and rules in a modular way, i.e. keeps both parts and their semantics separate, thus having similarities with $MKNF_{WFS}$. The interface for this approach is done by the `dlv-hex` system (Schindlauer 2006). Though with identical data complexity to $MKNF_{WFS}$ for a tractable DL, it has a less strong integration, having limitations in the way the ontology can call back program atoms (see (Eiter

[8] The proof of tractability of the implementation of *CDF-Rules* with CDF Type-0 ontologies is beyond the scope of this paper.

et al. 2004b) for details). Hybrid programs of (Drabent and Małuszynski 2007) 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 (both the one of (Motik and Rosati 2007) and $MKNF_{WFS}$) and also(Eiter et al. 2004b) in that if an ontology expresses $B_1 \vee B_2$ then the semantics in (Drabent and Małuszynski 2007) derives p from rules $p \leftarrow B_1$ and $p \leftarrow B_2$, p while MKNF and (Eiter et al. 2004b) do not. For further comparisons of MKNF with other proposals, including those not based on WFS rules, see (Motik and Rosati 2007; Knorr et al. 2008), and for a survey on other proposals for combining rules and ontologies see (Hitzler and Parsia 2009)

CDF-Rules serves as a proof-of-concept for querying $MKNF_{WFS}$ knowledge bases. As discussed, XSB and tractable CDF ontologies have been used extensively in commercial semantic web applications; the creation of *CDF-Rules* is a step towards understanding whether and how $MKNF_{WFS}$ can be used in such applications. As XSB is multi-threaded, *CDF-Rules* can be extended to a $MKNF_{WFS}$ server in a fairly straightforward manner. Since XSB supports CLP, further experiments involve representing temporal or spatial information in a hybrid of ontology, rules, and rule-based constraints. In addition, since the implementation of Flora-2 (Yang et al. 2003) and Silk (Grosof 2009) are both based on XSB, *CDF-Rules* forms a basis for experimenting with $MKNF_{WFS}$ on these systems.

References

Alferes, J. J., Knorr, M., and Swift, T. 2009. Queries to hybrid MKNF knowledge bases through oracular tabling. In *The Semantic Web - ISWC 2009*. LNCS, vol. 5823. Springer, 1–16.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., Eds. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2 ed. Cambridge University Press.

Baral, C. 2002. *Knowledge Representation, Reasoning, and Declarative Problem Solving.* Cambridge University Press.

Chen, W. and Warren, D. S. 1996. Tabled Evaluation with Delaying for General Logic Programs. *Journal of the ACM 43,* 1 (January), 20–74.

Drabent, W. and Małuszynski, J. 2007. Well-founded semantics for hybrid rules. In *RR2007.* Springer, 1–15.

Eiter, T., Lukasiewicz, T., Schindlauer, R., and Tompits, H. 2004a. Combining answer set programming with description logics for the semantic web. In *KR.* 141–151.

Eiter, T., Lukasiewicz, T., Schindlauer, R., and Tompits, H. 2004b. Well-founded semantics for description logic programs in the semantic web. In *RuleML'04.* 81–97.

Gelfond, M. and Lifschitz, V. 1990. Logic programs with classical negation. In *International Conference on Logic Programming*, Warren and Szeredi, Eds. MIT Press, 579–597.

Grimm, S. and Hitzler, P. Winter 2007–08. Semantic matchmaking of web resources with local closed-world reasoning. *International Journal of e-Commerce 12,* 2, 89–126.

Grosof, B. 2009. SILK: Semantic rules take the next big step in power. http://silk.semwebcentral.org.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S., Eds. 2009. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation 27 October 2009. Available from `http://www.w3.org/TR/owl2-primer/`.

Hitzler, P. and Parsia, B. 2009. Ontologies and rules. In *Handbook on Ontologies*, 2 ed., S. Staab and R. Studer, Eds. Springer.

Horrocks, I., Motik, B., Rosati, R., and Sattler, U. 2006. Can OWL and logic programming live together happily ever after? In *Proc. of the 5th Int. Semantic Web Conference (ISWC 2006), volume 4273 of LNCS*, I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, Eds. Springer, 501–514.

Horrocks, I. and Patel-Schneider, P. 1999. Optimizing description logic subsumption. *Journal of Logic and Computation 9,* 3.

Knorr, M., Alferes, J. J., and Hitzler, P. 2008. A coherent well-founded model for hybrid MKNF knowledge bases. In *Europ. Conf. on Artificial Intelligence*. IOS Press, 99–103.

Lifschitz, V. 1991. Nonmonotonic databases and epistemic queries. In *International Joint Conference on Artificial Intelligence*. 381–386.

Lukacsy, G., Szeredi, P., and Kadàr, B. 2008. Prolog based description logic reasoning. In *International Conference on Logic Programming*. 455–469.

Morgenstern, L., Welty, C., and Boley, H., Eds. 2010. *RIF Primer*. W3C Recommendation 22 June 2010. Available from `http://www.w3.org/2005/rules/wiki/Primer`.

Motik, B. 2006. Reasoning in description logics using resolution and deductive databases. Ph.D. thesis, University of Karlsruhe.

Motik, B. and Rosati, R. 2007. A faithful integration of description logics with logic programming. In *International Joint Conference on Artificial Intelligence*. 477–482.

Patel, C., Cimino, J. J., Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E., and Srinivas, K. 2007. Matching patient records to clinical trials using ontologies. In *ISWC/ASWC*. Springer, 816–829.

Schindlauer, R. 2006. Answer-set programming for the semantic web. Ph.D. thesis, Technischen Universitaet Wien Fakultaet fuer Informatik, Glockengasse 6/19, A-1020 Wien.

Schmidt-Strauss, M. and Smolka, G. 1990. Attributive concept descriptions with complements. *Artificial Intelligence 48*, 1–26.

Swift, T. 2004. Deduction in ontologies via answer set programming. In *International Conference on Logic Programming and Non-Monotonic Reasoning*. 275–289.

Swift, T. and Warren, D. 2011. XSB: Extending the power of Prolog using tabling. *Theory and Practice of Logic Programming*. To appear.

Swift, T. and Warren, D. S. 2003. *Cold Dead Fish: A System for Managing Ontologies.* Available via `http://xsb.sourceforge.net`.

van Gelder, A. 1989. The alternating fixpoint of logic programs with negation. In *Principles of Database Systems*. ACM Press, 1–10.

van Gelder, A., Ross, K. A., and Schlipf, J. S. 1991. Unfounded sets and well-founded semantics for general logic programs. *Journal of the ACM 38,* 3, 620–650.

Yang, G., Kifer, M., and Zhao, C. 2003. Flora-2: A rule-based knowledge representation and inference infrastructure for the semantic web. In *CoopIS/DOA/ODBASE*. 671–688.