A Query Tool for $\mathcal{EL}$ with Non-Monotonic Rules

Vadim Ivanov[1,2], Matthias Knorr[1], and João Leite[1]

[1] CENTRIA & Departamento de Informática, Universidade Nova de Lisboa, Portugal
[2] Department of Computing Mathematics and Cybernetics, Ufa State Aviation Technical University, Russia

Abstract. We present the Protégé plug-in NoHR that allows the user to take an $\mathcal{EL}^+_\bot$ ontology, add a set of non-monotonic (logic programming) rules – suitable e.g. to express defaults and exceptions – and query the combined knowledge base. Our approach uses the well-founded semantics for MKNF knowledge bases as underlying formalism, so no restriction other than DL-safety is imposed on the rules that can be written. The tool itself builds on the procedure **SLG($\mathcal{O}$)** and, with the help of OWL 2 EL reasoner ELK, pre-processes the ontology into rules, whose result together with the non-monotonic rules serve as input for the top-down querying engine XSB Prolog. With the resulting plug-in, even queries to very large ontologies, such as SNOMED CT, augmented with a large number of rules, can be processed at an interactive response time after one initial brief pre-processing period. At the same time, our system is able to deal with possible inconsistencies between the rules and an ontology that alone is consistent.

1 Introduction

Ontology languages have become widely used to represent and reason over taxonomic knowledge, and often such knowledge bases are expressed within the language of the OWL 2 profile OWL 2 EL [23]. For example, the clinical health care terminology SNOMED CT,[3] arguably the most prominent example in the area of medicine and currently used for electronic health record systems, clinical decision support systems, or remote intensive care monitoring, to name only a few, builds on a fragment of OWL 2 EL and its underlying description logic (DL) $\mathcal{EL}^{++}$ [5].

Whereas the OWL ontology languages based on DLs [4] are monotonic by nature, which means that once drawn conclusions persist when adopting new additional information, the ability to model defaults and exceptions with a closed-world view is frequently requested as a missing feature. For example, in [25], modeling pharmacy data of patients with the closed-world assumption would have been preferred in the study to match patient records with clinical trials criteria, because usually it can be assumed that a patient is not under a specific medication unless explicitly known. Similarly, in clinical health care terminology, it would be advantageous to be able to express that normally the heart is on the left side of the body unless the person is a dextrocardiac.

In recent years, there has been a considerable amount of effort devoted to extending DLs with non-monotonic features – see, e.g., related work in [12,24]) – and many of the existing approaches focus on combining DLs and non-monotonic rules as known

[3] http://www.ihtsdo.org/snomed-ct/

from Logic Programming. The latter are one of the most well-studied formalisms that admit expressing defaults, exceptions, and also integrity constraints in a declarative way and are part of RIF [6], the other expressive language for the Semantic Web whose standardization is driven by the W3C.[4]

Here we focus on one such combination – Hybrid MKNF under the well-founded semantics [20] – for two reasons. First, the overall approach, which was introduced in [24] and is based on the logic of minimal knowledge and negation as failure (MKNF) [22], provides a very general and flexible framework for combining DL ontologies and non-monotonic rules (see [24]). Second, [20], which is a variant of [24] based on the well-founded semantics [14] for logic programs, has a (lower) polynomial data complexity and is amenable for applying top-down query procedures, such as **SLG**($\mathcal{O}$) [2], to answer queries based only on the information relevant for the query, and without computing the entire model – no doubt a crucial feature when dealing with large ontologies such as SNOMED with over 300,000 classes or [25] with millions of assertions.

In this paper, we describe the system NoHR, realized as a plug-in for the ontology editor Protégé 4.X,[5] that allows the user to query combinations of $\mathcal{EL}^+_\bot$ ontologies and non-monotonic rules in a top-down manner. To the best of our knowledge, it is the first Protégé plug-in to integrate non-monotonic rules and top-down queries. Our approach is theoretically founded in the abstract procedure **SLG**($\mathcal{O}$) and based on utilizing the consequence-driven, concurrent $\mathcal{EL}$ reasoner ELK, which is considerably faster than other $\mathcal{EL}$ reasoners [18], to classify the ontology part and then translate the result into rules which, together with the non-monotonic rules, serve as input for the top-down query engine XSB Prolog.[6] Additional features of the plug-in include: the possibility to load and edit rule bases, and define predicates with arbitrary arity; guaranteed termination of query answering, with a choice between one/many answers; robustness w.r.t. inconsistencies between the ontology and the rule part. Our main contributions are:

- We generalize the procedure presented in [2] to avoid the normalization of $\mathcal{EL}^+_\bot$ knowledge bases, which is not necessary for ELK, also reducing the size of the XSB file and of the tables in XSB. At the same time we significantly improve the formalization in [2] – including the correct handling of complex concept assertions – in order to show that our procedure is correct.
- We describe an implementation of this revised procedure in Java including an ELK reasoner for preprocessing the ontology, whose translated output, together with the rules, can be queried interactively under XSB via the Java front-end Interprolog.[7]
- We evaluate the performance of our tool, showing that even SNOMED augmented with a large number of rules can be preprocessed in a brief period of time and then query answering is possible at interactive response time.

The remainder of the paper is structured as follows. In Sect. 2, we briefly recall the DL $\mathcal{EL}^+_\bot$ and MKNF knowledge bases as a tight combination of the former DL and non-monotonic rules. Then, we present the revised reasoning algorithm that allows us

[4] http://www.w3.org
[5] http://protege.stanford.edu
[6] http://xsb.sourceforge.net
[7] http://www.declarativa.com/interprolog/

Table 1. Syntax and semantics of $\mathcal{EL}^+_\bot$.

	Syntax	Semantics
atomic concept	$A \in \mathsf{N_C}$	$A^\mathcal{I} \subseteq \Delta^\mathcal{I}$
atomic role	$R \in \mathsf{N_R}$	$R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
individual	$a \in \mathsf{N_I}$	$a^\mathcal{I} \in \Delta^\mathcal{I}$
top	$\top$	$\Delta^\mathcal{I}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
existential restriction	$\exists R.C$	$\{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I} : (x,y) \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}$
concept inclusion	$C \sqsubseteq D$	$C^\mathcal{I} \subseteq D^\mathcal{I}$
role inclusion	$R \sqsubseteq S$	$R^\mathcal{I} \subseteq S^\mathcal{I}$
role composition	$R_1 \circ \cdots \circ R_k \sqsubseteq S$	$(x_1,x_2) \in R_1^\mathcal{I} \wedge \ldots \wedge (x_k,y) \in R_k^\mathcal{I} \rightarrow (x_1,y) \in S^\mathcal{I}$
concept assertion	$C(a)$	$a^\mathcal{I} \in C^\mathcal{I}$
role assertion	$R(a,b)$	$(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$

to query such MKNF knowledge bases in Sect. 3. In Sect. 4, we introduce our implementation of the plug-in and evaluate it in Sect. 5, before we conclude in Sect. 6.

2 Preliminaries

2.1 Description Logic $\mathcal{EL}^+_\bot$

We start by recalling the syntax and semantics of $\mathcal{EL}^+_\bot$, a large fragment of $\mathcal{EL}^{++}$ [5], the DL underlying the tractable profile OWL 2 EL [23], following the presentation in [18,19]. For a more general and thorough introduction to DLs we refer to [4].

The language of $\mathcal{EL}^+_\bot$ is defined over countably infinite sets of *concept names* $\mathsf{N_C}$, *role names* $\mathsf{N_R}$, and *individual names* $\mathsf{N_I}$ as shown in the upper part of Table 1. Building on these, *complex concepts* are introduced in the middle part of Table 1, which, together with atomic concepts, form the set of *concepts*. We conveniently denote individuals by a and b, (atomic) roles by R and S, atomic concepts by A and B, and concepts by C and D. All expressions in the lower part of Table 1 are *axioms*. A *concept equivalence* $C \equiv D$ is an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. Concept and role assertions are *ABox axioms* and all other axioms *TBox axioms*, and an *ontology* is a finite set of axioms.

The semantics of $\mathcal{EL}^+_\bot$ is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consisting of a non-empty domain $\Delta^\mathcal{I}$ and an *interpretation function* $\cdot^\mathcal{I}$. The latter is defined for (arbitrary) concepts, roles, and individuals as in Table 1. Moreover, an interpretation $\mathcal{I}$ *satisfies* an axiom α, written $\mathcal{I} \models \alpha$, if the corresponding condition in Table 1 holds. If $\mathcal{I}$ satisfies all axioms occurring in an ontology $\mathcal{O}$, then $\mathcal{I}$ is a *model* of $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$. If $\mathcal{O}$ has at least one model, then it is called *consistent*, otherwise *inconsistent*. Also, $\mathcal{O}$ *entails* axiom α, written $\mathcal{O} \models \alpha$, if every model of $\mathcal{O}$ satisfies α. *Classification* requires to compute all concept inclusions between atomic concepts entailed by $\mathcal{O}$.

2.2 MKNF Knowledge Bases

MKNF knowledge bases (KBs) build on the logic of minimal knowledge and negation as failure (MKNF) [22]. Two main different semantics have been defined [24,20], and we focus on the well-founded version [20], due to its lower computational complexity and amenability to top-down querying without computing the entire model. Here, we only point out important notions, and refer to [20] and [2] for the details.

We start by recalling MKNF knowledge bases as presented in [2] to combine an ($\mathcal{EL}^+_\bot$) ontology and a set of non-monotonic rules (similar to a normal logic program).

Definition 1. *Let $\mathcal{O}$ be an ontology. A function-free first-order* atom $P(t_1, \ldots, t_n)$ *such that P occurs in $\mathcal{O}$ is called* DL-atom*; otherwise it is called* non-DL-atom*. A* rule r *is of the form*

$$H \leftarrow A_1, \ldots, A_n, \mathbf{not} B_1, \ldots, \mathbf{not} B_m \tag{1}$$

where the head *of r, H, and all A_i with $1 \leq i \leq n$ and B_j with $1 \leq j \leq m$ in the* body *of r are atoms. A* program $\mathcal{P}$ *is a finite set of rules, and an* MKNF knowledge base $\mathcal{K}$ *is a pair $(\mathcal{O}, \mathcal{P})$. A rule r is* DL-safe *if all its variables occur in at least one non-DL-atom A_i with $1 \leq i \leq n$, and $\mathcal{K}$ is* DL-safe *if all its rules are DL-safe.*

DL-safety ensures decidability of reasoning with MKNF knowledge bases and can be achieved by introducing a new predicate o, adding $o(i)$ to $\mathcal{P}$ for all constants i appearing in $\mathcal{K}$ and, for each rule $r \in \mathcal{P}$, adding $o(X)$ for each variable X appearing in r to the body of r. Therefore, we only consider DL-safe MKNF knowledge bases.

Example 2. Consider an MKNF knowledge base for recommending vacation destinations taken from [24] (with a few modifications). We denote DL-atoms and constants with upper-case names and non-DL-atoms and variables with lower-case names.[8]

$$\begin{aligned}
PortCity(Barcelona) \quad & OnSea(Barcelona, Mediterranean) \\
PortCity(Hamburg) \quad & NonSeaSideCity(Hamburg) \\
RainyCity(Manchester) \quad & Has(Manchester, AquaticsCenter) \\
Recreational(AquaticsCenter) \quad &
\end{aligned}$$

$$\begin{aligned}
SeaSideCity &\sqsubseteq \exists Has.Beach \\
Beach &\sqsubseteq Recreational \\
\exists Has.Recreational &\sqsubseteq RecreationalCity \\
SeaSideCity(x) &\leftarrow PortCity(x), \mathbf{not}\, NonSeaSideCity(x) \\
interestingCity(x) &\leftarrow RecreationalCity(x), \mathbf{not}\, RainyCity(x) \\
hasOnSea(x) &\leftarrow OnSea(x, y) \\
false &\leftarrow SeaSideCity(x), \mathbf{not}\, hasOnSea(x) \\
summerDestination(x) &\leftarrow interestingCity(x), OnSea(x, y)
\end{aligned}$$

[8] To ease readability, we omit the auxiliary atoms that ensure DL-safety and leave them implicit.

This example shows that we can seamlessly express defaults and exceptions, such as every port city normally being a seaside city, integrity constraints, such as requiring to know for every seaside city on which sea it lies, and at the same time taxonomic/ontological knowledge including information over unknown individuals, such as a seaside city being recreational even if we do not know the specific name of the beach. Note that, unlike [24], the rule with head $false$ is not a true integrity constraint in our case. Rather, whenever the keyword $false$ would be derivable, we know that there is at least one seaside city for which we do not know on which sea it lies.

The semantics of MKNF knowledge bases $\mathcal{K}$ is usually given by a translation π into an MKNF formula $\pi(\mathcal{K})$, i.e., a formula over first-order logic extended with two modal operators $\mathbf{K}$ and $\mathbf{not}$. Namely, every rule of the form (1) is translated into $\mathbf{K}H \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m$, $\pi(\mathcal{P})$ is the conjunction of the translations of its rules, and $\pi(\mathcal{K}) = \mathbf{K}\pi(\mathcal{O}) \wedge \pi(\mathcal{P})$ where $\pi(\mathcal{O})$ is the first-order translation of $\mathcal{O}$. Reasoning with such MKNF formulas is then commonly achieved using a partition of *modal atoms*, i.e., all expressions of the form $\mathbf{K}\varphi$ for each $\mathbf{K}\varphi$ or $\mathbf{not}\varphi$ occurring in $\pi(\mathcal{K})$. For [20], such a partition assigns *true*, *false*, or *undefined* to (modal) atoms, and can be effectively computed in polynomial time. If $\mathcal{K}$ is *MKNF-consistent*, then this partition does correspond to the unique model of $\mathcal{K}$ [20], and, like in [2], we call the partition the *well-founded MKNF model* $\mathsf{M}_{\mathsf{wf}}(\mathcal{K})$. Here, $\mathcal{K}$ may indeed not be MKNF-consistent if the $\mathcal{EL}^+_\bot$ ontology alone is inconsistent, which is possible if $\bot$ occurs, or by the combination of appropriate axioms in $\mathcal{O}$ and $\mathcal{P}$, e.g., $A \sqsubseteq \bot$ and $A(a) \leftarrow$. In the former case, we argue that the ontology alone should be consistent and be repaired if necessary before combining it with non-monotonic rules. Thus, we assume in the following that $\mathcal{O}$ occurring in $\mathcal{K}$ is consistent.

2.3 Querying in MKNF Knowledge Bases

In [2], a procedure, called **SLG($\mathcal{O}$)**, is defined for querying MKNF knowledge bases under the well-founded MKNF semantics. This procedure extends SLG resolution with tabling [9] with an *oracle* to $\mathcal{O}$ that handles ground queries to the DL-part of $\mathcal{K}$ by returning (possibly empty) sets of atoms that, together with $\mathcal{O}$ and information already proven true, allows us to derive the queried atom. We refer to [2] for the full account of **SLG($\mathcal{O}$)**, and only recall a few crucial notions necessary in the following.

SLG($\mathcal{O}$) is based on creating top-down derivation trees with the aim of answering *(DL-safe) conjunctive queries* Q of the form $q(\boldsymbol{X}) \leftarrow A_1, \ldots, A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m$ where each variable in Q occurs in at least one non-DL atom in Q, and where $\boldsymbol{X}$ is the (possibly empty) set of requested variables appearing in the body.

In general, the computation of $\mathsf{M}_{\mathsf{wf}}(\mathcal{K})$ uses two different versions of $\mathcal{K}$ in parallel to guarantee that a) coherence is ensured, i.e., if $\neg P(a)$ is derivable, then $\mathbf{not}P(a)$ has to be true as well (cf. also [20]), and b) MKNF-consistency of $\mathcal{K}$ can be verified. For a top-down approach this is impractical, so, instead, a doubled MKNF knowledge base $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ is defined in which a copy of $\mathcal{O}$ with new doubled predicates is added, and two rules occur in $\mathcal{P}^d$ for each rule in $\mathcal{P}$, intertwining original and doubled predicates (see Def. 3.1 in [2]). It is shown that an atom A is true in $\mathsf{M}_{\mathsf{wf}}(\mathcal{K})$ iff A is

true in $\mathsf{M}_{\mathsf{wf}}(\mathcal{K}^d)$ and A is false in $\mathsf{M}_{\mathsf{wf}}(\mathcal{K})$ iff A^d is false in $\mathsf{M}_{\mathsf{wf}}(\mathcal{K}^d)$. Note that $\mathcal{K}^d$ is indeed necessary in general, but if $\mathcal{K}$ does not contain $\bot$, then we can use $\mathcal{K}$ directly.

In [2], the notion of oracle is defined to handle ground queries to the ontology, but before we recall that notion, we use an example to illustrate the idea.

Example 3. Recall $\mathcal{K}$ in Ex. 2. Since $\bot$ does not occur in $\mathcal{K}$, we can restrict ourselves to $\mathcal{K}$ here. First, consider query $q = \mathit{interestingCity}(\mathit{Manchester})$. We find a rule whose head unifies with q, and obtain two new queries, $\mathit{RecreationalCity}(\mathit{Manchester})$ and $\mathbf{not}\,\mathit{RainyCity}(\mathit{Manchester})$. There is no rule whose head matches the former, but we can query the ontology and the answer is yes together with an empty set of atoms, i.e., $\mathit{RecreationalCity}(\mathit{Manchester})$ can be proven from $\mathcal{O}$ alone. Now we handle $\mathbf{not}\,\mathit{RainyCity}(\mathit{Manchester})$, so we query $\mathit{RainyCity}(\mathit{Manchester})$ which can also be proven by $\mathcal{O}$ alone. Therefore $\mathbf{not}\,\mathit{RainyCity}(\mathit{Manchester})$ fails, so q is false.

Now, consider $q_1 = \mathit{interestingCity}(\mathit{Barcelona})$. We obtain again two new queries, $q_2 = \mathit{RecreationalCity}(\mathit{Barcelona})$ and $q_3 = \mathbf{not}\,\mathit{RainyCity}(\mathit{Barcelona})$. In this case, $q_2 = \mathit{RecreationalCity}(\mathit{Barcelona})$ cannot be proven from $\mathcal{O}$ alone, but the oracle could return $\mathit{Has}(\mathit{Barcelona}, X)$ and $\mathit{Recreational}(X)$, which, if we would find a value for X, would allow us to derive q_2. However, neither of the two atoms appear in a rule head in $\mathcal{P}$, so we will never be able to derive it from $\mathcal{P}$. In fact, the only proper answer the oracle may return is $q_4 = \mathit{SeaSideCity}(\mathit{Barcelona})$. From the corresponding rule in $\mathcal{P}$ we obtain two new queries $q_5 = \mathit{PortCity}(\mathit{Barcelona})$ and $q_6 = \mathbf{not}\,\mathit{NonSeaSideCity}(\mathit{Barcelona})$. Then, q_5 can be derived from $\mathcal{O}$ alone, and q_6 succeeds, because $\mathit{NonSeaSideCity}(\mathit{Barcelona})$ fails. So q_4 succeeds, and therefore also q_2. Finally q_3 succeeds since $\mathit{RainyCity}(\mathit{Barcelona})$ fails, so q_1 is true.

We recall the notions of a complete and a (correct) partial oracle from [2].

Definition 4. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB, $\mathcal{I}$ a set of ground atoms (already proven to be true), S a ground query, and $\mathcal{L}$ a set of ground atoms such that each $L \in \mathcal{L}$ is unifiable with at least one rule head in $\mathcal{P}^d$. The* complete oracle *for $\mathcal{O}$, denoted $\mathit{compT}_{\mathcal{O}}$, is defined by $\mathit{compT}_{\mathcal{O}}(\mathcal{I}, S, \mathcal{L})$ iff $\mathcal{O} \cup \mathcal{I} \cup \mathcal{L} \models S$ or $\mathcal{O}^d \cup \mathcal{I} \cup \mathcal{L} \models S$. A* partial oracle *for $\mathcal{O}$, denoted $\mathit{pT}_{\mathcal{O}}$, is a relation $\mathit{pT}_{\mathcal{O}}(\mathcal{I}, S, \mathcal{L})$ such that if $\mathit{pT}_{\mathcal{O}}(\mathcal{I}, S, \mathcal{L})$, then $\mathcal{O} \cup \mathcal{I} \cup \mathcal{L} \models S$ or $\mathcal{O}^d \cup \mathcal{I} \cup \mathcal{L} \models S$ for consistent $\mathcal{O} \cup \mathcal{I} \cup \mathcal{L}$ and $\mathcal{O}^d \cup \mathcal{I} \cup \mathcal{L}$, respectively.*

A partial oracle $\mathit{pT}_{\mathcal{O}}$ is correct *w.r.t. $\mathit{compT}_{\mathcal{O}}$ iff, for all MKNF-consistent $\mathcal{K}^d$, replacing $\mathit{compT}_{\mathcal{O}}$ in* **SLG($\mathcal{O}$)** *with $\mathit{pT}_{\mathcal{O}}$ succeeds for exactly the same set of queries.*

Partial oracles may avoid returning unnecessary answers $\mathcal{L}$, such as non-minimal answers or those that try to derive an MKNF-inconsistency even though $\mathcal{K}^d$ is MKNF-consistent. Moreover, correctness of partial oracles is only defined w.r.t MKNF-consistent $\mathcal{K}$. The rationale is that, when querying top-down, we want to avoid checking whether the entire KB $\mathcal{K}^d$ is MKNF-consistent. This leads to para-consistent derivations if $\mathcal{K}^d$ is not MKNF-consistent, e.g., some atom P is true, yet P^d is false, while other independent atoms are evaluated as if $\mathcal{K}^d$ was MKNF-consistent (see [2]).

3 Pre-processing the Ontology and Querying

In [2], an **SLG($\mathcal{O}$)** oracle for $\mathcal{EL}^+_\bot$ is defined based on the reasoning algorithm for ontology classification presented in [5], which is restricted to normalized ontologies. Even though the process of normalizing $\mathcal{EL}$ ontologies is linear, it introduces auxiliary predicates to achieve the normal form, which, e.g., not only is counter-intuitive to the idea of finding meaningful explanations for information derived by top-down queries but also increases the size of the resulting XSB file and tables used in XSB. Since the reasoning algorithm of ELK [18,19] is defined for the general case, and ELK is the reasoner we want to use for implementing our query-tool, because it is considerably faster than other $\mathcal{EL}$ reasoners, we generalize the algorithm in [2] to non-normalized ontologies. At the same time, we fix a problem in [2] w.r.t. handling non-atomic concept assertions, and we improve the formalization to prove correctness of our oracle.

In the following, we first utilize the algorithm underlying ELK to compute implicit information derivable from a given ontology. After that, we discard certain axioms, because, with the implicit information computed, they are no longer required for the query task used in **SLG($\mathcal{O}$)**. Then, we translate the remaining set of axioms into rules, which can equally be used as an $\mathcal{EL}$ oracle.

3.1 Simplifying the Ontology

The basic idea for an $\mathcal{EL}^+_\bot$ oracle is to translate the ontology into rules. The only obstacle are concept inclusions with $\exists R.C$ on the right-hand side since these cannot be translated straightforwardly. However, such axioms alone can never contribute to oracle derivations. Consider, e.g., querying $\mathcal{K}$ in Ex. 2 for $Beach(i)$ for any constant i: an oracle cannot derive $Beach(i)$ even if $SeaSideCity(c)$ and $has(c, i)$ were already known to be true. Yet, such axioms are useful indirectly, e.g., to obtain that proving $SeaSideCity(Barcelona)$ would suffice to derive $RecreationalCity(Barcelona)$. So, classification is applied first to make such implicit links explicit, and then all concept inclusions with sub-concepts $\exists R.C$ on the right-hand side can be rewritten or removed.

The consequence-based procedure for classification of TBoxes in $\mathcal{EL}^+_\bot$ is described in [19]. Here we only sketch it with a focus on the results important for our purposes.

First, the initial set $\mathbf{input}$ is defined to contain one axiom $\mathbf{init}(A)$ for each atomic concept A in $\mathcal{O}$. Then, a set of $\mathcal{EL}^+_\bot$ *inference rules* for TBox reasoning in $\mathcal{EL}^+_\bot$ (see [19]) is applied exhaustively to $\mathbf{input}$, yielding $\mathbf{Closure}$ as final result, which contains axioms derivable from $\mathcal{O}$ and axioms of the form $\mathbf{init}(C)$ and $C \xrightarrow{\mathsf{R}} D$, where the latter represents that, for two (initialized) concepts C and D, $C \sqsubseteq \exists R.D$ is entailed.

Theorem 5 ([19]). *Let $\mathcal{O}$ be an $\mathcal{EL}^+_\bot$ ontology, $\mathbf{input}$ a set of expressions $\mathbf{init}(C)$, and $\mathbf{Closure}$ the closure of $\mathbf{input}$ under the $\mathcal{EL}^+_\bot$ inference rules w.r.t. $\mathcal{O}$. Then, for each concept C such that $\mathbf{init}(C) \in \mathbf{Closure}$ and each atomic concept A, we have*

1. $\mathcal{O} \models C \sqsubseteq \bot$ *iff* $C \sqsubseteq \bot \in \mathbf{Closure}$,
2. $\mathcal{O} \models C \sqsubseteq A$ *iff* $C \sqsubseteq \bot \in \mathbf{Closure}$ *or* $C \sqsubseteq A \in \mathbf{Closure}$.

Note that one of the inference rules ($\mathsf{R}^-_\exists$) allows $\mathbf{init}(D) \in \mathbf{Closure}$ if $C \sqsubseteq \exists R.D \in \mathbf{Closure}$. Hence, Theorem 5 does not apply only to $\mathbf{init}(C) \in \mathbf{input}$.

One thing missing so far, which is not correctly covered in the translation presented in [2], is that concept assertions also have to be considered since, e.g., $\exists R.C(a)$ together with $\exists R.C \sqsubseteq D$, make $D(a)$ derivable, yet the $\mathcal{EL}^+_\bot$ inference rules are defined for TBox axioms. The solution, adapted from [19], is to apply a transformation N that translates concept assertions $C(a)$ into concept inclusions $N_a \sqsubseteq C$, where the set of atomic concepts contains an atomic concept N_a for every $a \in \mathsf{N_I}$ s.t. N_a does not appear in $\mathcal{O}$, and leaves all other axioms unchanged. Note that we do not translate role assertions $R(a, b)$, because R is just an atomic role without occurrences of concepts of the form $\exists R.C$. Still, it holds for consistent $\mathcal{O}$ and axiom α that do not contain atomic concepts of the form N_a, that $\mathcal{O} \models \alpha$ iff $N(\mathcal{O}) \models N(\alpha)$ ([19], Theorem 3).

We are now ready to present the new definition of a reduced ontology.

Definition 6. *Let $\mathcal{O}$ be an $\mathcal{EL}^+_\bot$ ontology,* $\mathbf{input}$ *a set of expressions* $\mathbf{init}(A)$ *for each atomic concept A, and* $\mathbf{Closure}$ *the closure of* $\mathbf{input}$ *under the $\mathcal{EL}^+_\bot$ inference rules w.r.t. $N(\mathcal{O})$. The* reduced ontology *of $\mathcal{O}$ is obtained from $\mathcal{O}_1 = N(\mathcal{O}) \cup \mathbf{Closure}$ as follows.*

1. *Remove all statements of the forms $\mathbf{init}(C)$, $C \xrightarrow{R} D$, and $C \sqsubseteq \exists R.D$ from $\mathcal{O}_1$;*
2. *Remove all sub-concepts of the form $\exists R.D$ from the right-hand side of any axiom $C \sqsubseteq D_1 \sqcap D_2 \in \mathcal{O}_1$; if no conjunct is left, remove the entire axiom from $\mathcal{O}_1$;*
3. *Substitute all concept inclusions of the form $N_a \sqsubseteq C$ remaining after 1. and 2. by $C(a)$ for all $a \in \mathsf{N_I}$.*

Note that $\mathbf{input}$ does contain $\mathbf{init}(N_a)$ for all $a \in \mathsf{N_I}$. Moreover, steps 1. and 2. already remove all sub-concepts of the form $\exists R.D$ from concept inclusions that represent concept assertions, so indeed the reduced ontology does not contain concepts of the form $\exists R.D$ in concept assertions and the right-hand sides of concept inclusions. We can show that reduced $\mathcal{O}$ contain all atomic concept assertions they entail.

Lemma 7. *Let $\mathcal{O}$ be reduced and A an atomic concept. If $\mathcal{O} \models A(a)$, then $A(a) \in \mathcal{O}$.*

Also $\mathcal{O}$ and the reduced $\mathcal{O}'$ entail the same unary and binary atoms.

Lemma 8. *Let $\mathcal{O}$ be an $\mathcal{EL}^+_\bot$ ontology, $\mathcal{O}'$ the reduced ontology of $\mathcal{O}$, A a unary and R a binary predicate: $\mathcal{O} \models A(a)$ iff $\mathcal{O}' \models A(a)$ and $\mathcal{O} \models R(a, b)$ iff $\mathcal{O}' \models R(a, b)$.*

Now, we show that we can use the reduced ontology of $\mathcal{O}$ instead of $\mathcal{O}$ as an $\mathcal{EL}^+_\bot$ oracle. First, we specify a partial oracle that is necessarily a correct partial oracle for $\mathcal{O}$.

Definition 9. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB, $\mathcal{I}$ a set of ground atoms (already proven to be true), S a ground query, and $\mathcal{L}$ a set of ground atoms such that each $L \in \mathcal{L}$ is unifiable with at least one rule head in $\mathcal{P}^d$. The* abstract partial oracle *for $\mathcal{O}$, denoted $pT^a_\mathcal{O}$, is a relation $pT^a_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ such that $pT^a_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ iff $\mathcal{O} \cup \mathcal{I} \cup \mathcal{L} \models S$ or $\mathcal{O}^d \cup \mathcal{I} \cup \mathcal{L} \models S$ for consistent $\mathcal{O} \cup \mathcal{I} \cup \mathcal{L}$ and $\mathcal{O}^d \cup \mathcal{I} \cup \mathcal{L}$, respectively.*

Such an abstract partial oracle is not necessarily efficient, since it does return all possible consistent tuples $(\mathcal{I}, S, \mathcal{L})$ but it certainly is correct.

Proposition 10. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB. The abstract partial oracle $pT^a_\mathcal{O}$ is correct w.r.t. $compT_\mathcal{O}$.*

Based on this result, we can show that $\mathcal{O}$ can be substituted with the reduced ontology of $\mathcal{O}$ and still yield a correct partial oracle.

Theorem 11. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB and $\mathcal{O}'$ the reduced ontology of $\mathcal{O}$. Then $pT^a_{\mathcal{O}'}$ is a correct partial oracle w.r.t. $compT_{\mathcal{O}}$.*

3.2 Translation into Rules

Now, we can show how to translate a reduced $\mathcal{EL}^+_\bot$ ontology into rules. The only thing missing is pointing out a technical detail on how coherence and detection of MKNF-inconsistencies is achieved in [20,2]. We would like to remind that coherence intuitively ensures that an atom being "classically" false also is false by default in the non-monotonic rules. That allows us, e.g., to derive from $C \sqsubseteq \bot$, $C(a) \leftarrow \mathbf{not} D(a)$, and $D(a) \leftarrow \mathbf{not} C(a)$ that $C(a)$ is false and $D(a)$ is true, and is therefore useful in general, even if $\mathcal{K}^d$ is MKNF-consistent. In **SLG($\mathcal{O}$)**, special atoms $NH(\boldsymbol{t_i})$ are used to represent a query $\neg H(\boldsymbol{t_i})$ to the oracle. Care must be taken when translating an ontology containing $\bot$ to rules, so that these queries still work properly. Of course, if $\bot$ does not appear in $\mathcal{O}$, then considering $\mathcal{K}$ suffices.

To ease the presentation of the following translation, we introduce a few a priori simplifications. First, in a reduced ontology, the concepts C of concept assertions $C(a)$ and the right-hand sides D of concept inclusions $C \sqsubseteq D$ are all of the form $C_1 \sqcap \ldots \sqcap C_n$ with $n > 0$. We can separate these into an equivalent set of n axioms $C_i(a)$ and $C \sqsubseteq C_i$, respectively, for $1 \leq i \leq n$, simplifying in particular where some C_i is $\bot$ and another an atomic concept. Moreover, we assume without loss of generality that $\bot$ does not occur on the left-hand side of concept inclusions. We know that $\bot \sqcap C$ and $\exists R.\bot$ are both equivalent to $\bot$ and $\bot \sqsubseteq C$ is always true, so we do not need to care translating such cases into rules. Finally, for $\top$, only axioms of the form $\top \sqsubseteq C$ and sub-concepts of the form $\exists R.\top$ deserve our attention, all other instances, namely $\top(a)$, $C \sqsubseteq \top$, or $C \sqcap \top \sqsubseteq D$, are irrelevant. Hence, $\top$ is not considered in assertions, the right-hand side of concept inclusions, and in conjunctions on the left-hand side of concept inclusions, all the more, since $\top$ does not appear in $\mathcal{P}$. We also omit the auxiliary DL-safe atoms.

First, we translate arbitrary concepts into a set of correctly connected atoms.

Definition 12. *Let C be an $\mathcal{EL}^+_\bot$ concept without occurrences of $\bot$, x a variable, and X a set of variables with $x \notin X$. We define $tr(C, x)$ as follows:*

$$tr(C,x) = \begin{cases} \{A(x)\} & \text{if } C = A \\ \emptyset & \text{if } C = \top \\ tr(C_1,x) \cup tr(C_2,x) & \text{if } C = C_1 \sqcap C_2 \\ \{R(x,y)\} \cup tr(D,y) & \text{if } C = \exists R.D \end{cases}$$

where $y \in X$ is a new variable that has not been used before. We obtain $tr(C,x)^d$ from $tr(C,x)$ by substituting all predicates P in $tr(C,x)$ with P^d, and, given a set of atoms S, $\overline{S}$ is a sequence of all atoms contained in S separated by ",".

Each y is indeed intended to be a variable that is globally new in the process of translating C. E.g., $\exists R.((\exists S.\top) \sqcap (A \sqcap \exists R.B))$ translates into $S = \{R(x, y_1), S(y_1, y_2), A(y_1), R(y_1, y_3), B(y_3)\}$. Then $\overline{S}$ is simply "$R(x, y_1), S(y_1, y_2), A(y_1), R(y_1, y_3), B(y_3)$" which can appear as such in a rule body.

Definition 13. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be an MKNF KB with consistent and reduced $\mathcal{O}$. We define $\mathcal{P}^d_\mathcal{O}$ from $\mathcal{O}$, where A is an atomic concept, C a concept, D a non-atomic concept, R, R_i, S roles, and a, b individuals, as the smallest set containing:*

(a1) *for each $A(a) \in \mathcal{O}$: $A(a) \leftarrow$ and $A^d(a) \leftarrow \mathbf{not} NA(a)$.*
(a2) *for each $R(a, b) \in \mathcal{O}$: $R(a, b) \leftarrow$ and $R^d(a, b) \leftarrow \mathbf{not} NR(a, b)$.*
(t1) *for each $\top \sqsubseteq A \in \mathcal{O}$: $A(x) \leftarrow$ and $A^d(x) \leftarrow \mathbf{not} NA(x)$.*
(c1) *for each $C \sqsubseteq A \in \mathcal{O}$: $A(x) \leftarrow \overline{tr(C, x)}$ and $A^d(x) \leftarrow \overline{tr(C, x)^d}, \mathbf{not} NA(a)$.*
(r1) *for each $R \sqsubseteq S \in \mathcal{O}$: $S(x, y) \leftarrow R(x, y)$ and $S^d(x, y) \leftarrow R^d(x, y), \mathbf{not} NS(x, y)$.*
(r2) *for each $R_1 \circ \cdots \circ R_k \sqsubseteq S \in \mathcal{O}$: $S(x_1, y) \leftarrow R_1(x_1, x_2), \ldots, R_k(x_k, y)$ and $S^d(x_1, y) \leftarrow R^d_1(x_1, x_2), \ldots, R^d_k(x_k, y), \mathbf{not} NS(x1, y)$.*
(i1) *for each $A \sqsubseteq \bot \in \mathcal{O}$: $NA(x) \leftarrow$.*
(i2) *for each $D \sqsubseteq \bot \in \mathcal{O}$: $\{NA(y) \leftarrow \overline{tr(D, x) \setminus \{A(y)\}} \mid A(y) \in tr(D, x)\} \cup \{NR(y, z) \leftarrow \overline{tr(D, x) \setminus \{R(y, z)\}} \mid R(y, z) \in tr(D, x)\}$.*

We create in $\mathcal{P}^d_\mathcal{O}$ the rule representation for both $\mathcal{O}$ and $\mathcal{O}^d$. Again, if $\bot$ does not occur in $\mathcal{O}$, then we can skip all rules with doubled predicates, and (i1) and (i2) will not contribute anything either. The additional default atoms of the form $\mathbf{not} NA(x)$ and $\mathbf{not} NS(x, y)$ are added to the doubled rules to be in line with the idea of the doubling of rules in [2]: whenever, e.g., $A(x)$ is "classically false" for some x, then we make sure that $A^d(x)$ is derviable as false for that same x from the rules, but not necessarily $A(x)$, thus allowing to detect potential MKNF-inconsistencies. That is also the reason why (i1) and (i2) do not produce the doubled counterparts: atoms based on predicates of the forms NC^d or NR^d are not used anywhere.

We can show that the translation also maintains derivability of atoms.

Lemma 14. *Let $\mathcal{O}$ be a reduced $\mathcal{EL}^+_\bot$ ontology, A a unary and R a binary predicate: $\mathcal{O} \models A(a)$ iff $\mathcal{P}^d_\mathcal{O} \models A(a)$ and $\mathcal{O}^d \models A^d(a)$ iff $\mathcal{P}^d_\mathcal{O} \models A^d(a)$, and, likewise, $\mathcal{O} \models R(a, b)$ iff $\mathcal{P}^d_\mathcal{O} \models R(a, b)$ and $\mathcal{O}^d \models R^d(a, b)$ iff $\mathcal{P}^d_\mathcal{O} \models R^d(a, b)$.*

Thus, we can define a correct partial oracle based on $\mathcal{P}^d_\mathcal{O}$.

Theorem 15. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB, $\mathcal{O}_1$ the reduced ontology of $\mathcal{O}$, and $pT^{\mathcal{EL}}_\mathcal{O}$ a* partial $\mathcal{EL}$ oracle *such that $pT^{\mathcal{EL}}_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ iff $\mathcal{P}^d_{\mathcal{O}_1} \cup \mathcal{I} \cup \mathcal{L} \models S$. Then $pT^{\mathcal{EL}}_\mathcal{O}$ is a correct partial oracle w.r.t. $compT_\mathcal{O}$.*[9]

Instead of coupling two rule reasoners that interact with each other using an oracle, we can simplify the process altogether and integrate both into one rule reasoner. The resulting approach is decidable with data complexity in $\mathbf{P}$.

Theorem 16. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be an MKNF KB with $\mathcal{EL}^+_\bot\mathcal{O}$. An* **SLG($\mathcal{O}$)** *evaluation of a query in $\mathcal{K}_{\mathcal{EL}^+_\bot} = (\emptyset, (\mathcal{P}^d \cup \mathcal{P}^d_\mathcal{O}))$ is decidable with data complexity in $\mathbf{P}$.*

[9] Of course, the partial $\mathcal{EL}$ oracle has to be defined w.r.t. the reduced $\mathcal{O}_1$.

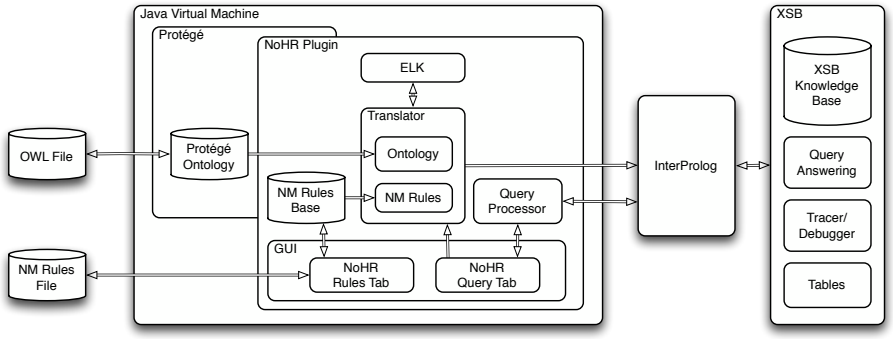

Fig. 1. System Architecture of NoHR

4 System Description

In this Section, we briefly describe the architecture of our plug-in for Protégé as shown in Fig. 1 and discuss some features of our implementation and querying in XSB.

The input for our plug-in consists of an OWL file, which can be manipulated as usual in Protégé, and a rule file. For the latter, we provide a tab called NoHR Rules that allows us to load, save and edit rule files in a text panel. The syntax follows Prolog conventions, so that one rule from Ex. 2 can be represented, e.g., by

```
SeaSideCity(X) :- PortCity(X), not NonSeaSideCity(X).
```

The NoHR Query tab also allows for the visualization of the rules, but its main purpose is to provide an interface for querying the combined KB. Whenever the first query is posed by pushing "Execute", the translator is started, initiating the ELK reasoner to classify the ontology and return the inferred axioms to translator. It is verified whether $DisjointWith$ axioms appear in $\mathcal{O}$ which determines whether the application of the transformations presented in Sect. 3 provides a doubled set of rules or not, and whether the non-monotonic rules have to be doubled as well. Then, accordingly, a joint (non-monotonic) rule set is created in which all predicates and constants are encoded using MD5 to ensure full compatibility with XSB Prolog's more restrictive admitted input syntax. The result is transfered to XSB via InterProlog [8], which is an open-source Java front-end allowing the communication between Java and a Prolog engine.

Next, the query is sent via InterProlog to XSB, and answers are returned to the query processor, which collects them and sets up a table showing for which variable substitutions we obtain true, undefined, or inconsistent valuations (or just shows the truth value for a ground query). The table itself is shown in the Result tab of the Output panel, while the Log tab shows measured times and system messages, including those from XSB via InterProlog. XSB itself not only answers queries very efficiently in a top-down manner, with tabling, it also avoids infinite loops.

Once the query has been answered, the user may pose other queries, and the system will simply send them directly without any repeated preprocessing. If the user changes

	# Axioms	ELK	Translator	XSB	Total
ChEBI	67184	3,92	7,41	1,75	9,16
EMAP	13730	2,94	5,14	0,00	5,14
Fly Anatomy	19211	2,70	4,99	0,83	5,82
FMA	126548	7,56	14,67	3,58	18,25
GALEN-OWL	36547	5,00	8,76	2,46	11,22
GALEN7	44461	5,88	9,67	4,56	14,23
GALEN8	73590	21,90	42,76	28,12	70,88
GO1	28897	3,74	6,06	1,09	7,15
GO2	73590	4,57	8,90	5,02	13,92
Molecule Role	9629	3,41	5,14	0,15	5,29
SNOMED CT	294480	20,61	43,32	24,69	68,01

Fig. 2. Preprocessing time (s) of different $\mathcal{EL}$ ontologies

data in the ontology or in the rules, then the system offers the option to recompile, but always restricted to the part that actually changed.

Our plug-in is under active development and the most recent version is available at https://code.google.com/p/nohr-reasoner/.

5 Evaluation

In this section, we evaluate our system with the aim of showing that a) different $\mathcal{EL}$ ontologies can be preprocessed for querying in a short period of time, b) adding rules increases the time of the translation only linearly, and c) querying time is in comparison to a) and b) in general completely neglectable.

We performed the tests on a Mac book air 13 under Mac OS X 10.8.4 with a 1.8 GHz Intel Core i5 processor and 8 GB 1600 MHz DDR3 of memory. We used OWL API 3.4.2 for managing ontologies, the ELK libraries 0.3.2, directly integrated into our tool, InterProlog 2.3a4 as Java front end between Java and Prolog, and XSB 3.4.0 for querying. We ran all tests in a terminal version and Java with the "-XX:+AggressiveHeap" option, and test results are averages over 5 runs.

First, we preprocessed a number of ontologies without additional rules and measured the time. The results are shown in Fig. 2.

We considered the ontologies, mentioned in [19], that are available online[10] or, in the case of SNOMED CT, freely available for research and evaluation.[11] The second column of Fig. 2 shows the number of axioms in each ontology to give an idea of their size. Very detailed measures of these ontologies are presented in [19].

[10] http://code.google.com/p/elk-reasoner/
[11] http://www.ihtsdo.org/licensing/

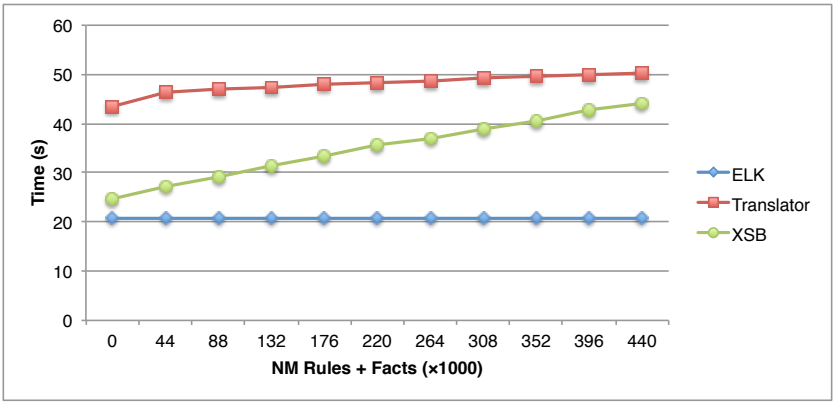

Fig. 3. Preprocessing time for SNOMED with a varying number of Rules

We measured the time it takes to only classify these ontologies with the ELK standalone application for comparison (third column), and also the time to translate the OWL file into a file that can be loaded into XSB (Translator, fourth column). Then, we measured the time to load the resulting file dynamically into XSB (fifth column) and we also show the total time (sixth column, sum of columns four and five).

One can observe that the total time is not proportional to the number of axioms, but rather also influenced by which kinds of axioms appear in each ontology which in particular affects the reasoning time of ELK (see also [19]). We can also see that the translator (including ELK processing) takes approximately twice the amount of time of ELK alone, while loading the file in XSB varies from 0 sec. (EMAP) to approximately the same time as running ELK alone, depending on the number of derived axioms that can be translated into rules. Still, the total preprocessing time varies from 5.14 sec. to 70.88 sec. which we belief is acceptable since this has to be executed only once before querying.

Next, we considered only SNOMED CT and added a varying number of non-monotonic rules. These rules were generated arbitrarily, using predicates from the ontology and additional new predicates (up to arity three), producing rules with a random number of body atoms varying from 1 to 10 and facts (rules without body atoms) with a ratio of 1:10. Note that, due to the translation of the DL part into rules, all atoms literally become non-DL-atoms. So ensuring that each variable appearing in the rule is contained in at least one non-negated body atom suffices to guarantee DL-safety for these rules.

The results are shown in Fig. 3 (containing also a constant line for classification of ELK alone and starting with the values from the first experiment with no additional rules), and clearly show that the time of translator and loading the file in XSB only grows linearly on the number of rules with a small degree, in particular in the case of translator. This indicates that adding non-monotonic rules to ontologies has a rather low impact on preprocessing.

Finally, we tested the querying time. To this purpose, we randomly generated and handcrafted several queries of different sizes and shapes, some satisfiable while others not, using SNOMED CT with a varying number of non-monotonic rules as described in the previous experiment. More concretely, we tested queries ranging from single atoms to complex conjunctive queries. We also varied the depth of classes and properties in the hierarchies, how many sub-elements the considered classes/properties have themselves (the more they have, the more answers are returned w.r.t. the arbitrarily many created rules and facts), and how variables are connected. E.g., we considered sequences of atoms without any (positive variable) connection between them, thus creating all combinations of the answers for each atom, but also queries in which properties (or new predicates of arity greater 1) introduce connections or where the same variable appears in different queried classes. In all cases, we observed that the query response time is interactive, mostly significantly below one second, observing longer reply times only if the number of replies is very high because either the queried class contains many subclasses in the hierarchy or if the arbitrarily generated rules create too many meaningless links, thus in the worst case requiring to compute the entire model. Requesting only one solution avoids this problem. Additionally, the question of realistic randomly generated rule bodies for testing querying time remain an issue of future work.

6 Conclusions

We have presented NoHR, the first plug-in for the ontology editor Protégé that integrates non-monotonic rules and top-down queries with ontologies in the OWL 2 profile OWL 2 EL. Our approach realizes an **SLG**($\mathcal{O}$) oracle for $\mathcal{EL}^+_\bot$, utilizing the $\mathcal{EL}$ reasoner ELK for preprocessing an ontology and then translating it into rules, which can be queried together with the non-monotonic rules in XSB.

We have generalized the procedure presented in [2] to non-normalized $\mathcal{EL}^+_\bot$ knowledge bases and shown that this formalization provides a correct **SLG**($\mathcal{O}$) oracle. We also have discussed how this procedure is implemented in our tool, and that it offers the representation of non-monotonic knowledge such as defaults and exceptions in a seamless way. We have evaluated the performance showing that different $\mathcal{EL}$ ontologies can be preprocessed for querying in a short period of time, adding rules increases this time only linearly, and querying time is in comparison to preprocessing insignificant.

There are several relevant approaches discussed in the literature. Most closely related are probably [16,21], because both build on the well-founded MKNF semantics [20]. In fact, [16] is maybe closest in spirit to the original idea of **SLG**($\mathcal{O}$) oracles presented in [1]. It utilizes the CDF framework already integrated in XSB, but its non-standard language is a drawback if we want to achieve compatibility with standard OWL tools based on the OWL API. On the other hand, [21], presents an OWL 2 QL oracle based on common rewritings in the underlying DL DL-Lite [3]. Less closely related is the work pursued in [7,15] that investigates direct non-monotonic extensions of $\mathcal{EL}$, so that the main reasoning task focuses on finding default subset inclusions, unlike our query-centered approach.

Two related tools are DReW [26] and HD Rules [10], but both are based on different underlying formalisms to combine ontologies and non-monotonic rules. The former

builds on dl-programs [12] and focuses on datalog-rewritable DLs [17], and the latter builds on Hybrid Rules [11]. While a more detailed comparison is surely of interest, the main problem is that both underlying formalisms differ from MKNF knowledge bases in the way information can flow between its two components and how flexible the language is [12,24]. Finally, SWRL-IQ [13] is also interesting because it utilizes expressive features of XSB, but in a monotonic setting and under Protégé 3.X, so OWL 2 is not supported.

In terms of future work, we consider extending our tool so that it is possible to load files in RIF format, thereby achieving a tool to jointly utilize the two language paradigms present in the ongoing standardization of the Semantic Web. We also want to work on extending the language, to allow nominals, or at least safe nominals [19]. Finally, improving the user interface is also in our focus leveraging XSB language features, in particular in the light of [13], which uses an expressive query language and elaborate traces for finding explanations.

Acknowledgments. We would like to thank Miguel Calejo for his help with InterProlog, Pavel Klinov for his help with ELK, Terry Swift for his help with XSB, and Gonca Güllü for her collaboration. Vadim Ivanov was partially supported by a MULTIC – Erasmus Mundus Action 2 grant. Matthias Knorr and João Leite were partially supported by FCT funded project ERRO – Efficient Reasoning with Rules and Ontologies (PTDC/EIA-CCO/121823/2010) and Matthias Knorr also by FCT grant SFRH/BPD/86970/2012.

References

1. Alferes, J.J., Knorr, M., Swift, T.: Queries to hybrid MKNF knowledge bases through oracular tabling. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC. LNCS, vol. 5823, pp. 1–16. Springer (2009)
2. Alferes, J.J., Knorr, M., Swift, T.: Query-driven procedures for hybrid MKNF knowledge bases. ACM Trans. Comput. Log. 14(2), 1–43 (2013)
3. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The $DL\text{-}Lite$ family and relations. J. Artif. Intell. Res. (JAIR) 36, 1–69 (2009)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 3rd edn. (2010)
5. Baader, F., Brandt, S., Lutz, C.: Pushing the el envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) IJCAI. pp. 364–369. Professional Book Center (2005)
6. Boley, H., Kifer, M. (eds.): RIF Overview. W3C Recommendation (5 February 2013), available at http://www.w3.org/TR/rif-overview/
7. Bonatti, P.A., Faella, M., Sauro, L.: $\mathcal{EL}$ with default attributes and overriding. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC (1). LNCS, vol. 6496, pp. 64–79. Springer (2010)
8. Calejo, M.: Interprolog: Towards a declarative embedding of logic programming in java. In: Alferes, J.J., Leite, J.A. (eds.) JELIA. LNCS, vol. 3229, pp. 714–717. Springer (2004)
9. Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. J. ACM 43(1), 20–74 (1996)

10. Drabent, W., Henriksson, J., Maluszynski, J.: Hd-rules: A hybrid system interfacing prolog with dl-reasoners. In: Polleres, A., Pearce, D., Heymans, S., Ruckhaus, E. (eds.) ALPSWS. CEUR Workshop Proceedings, vol. 287. CEUR-WS.org (2007)
11. Drabent, W., Maluszynski, J.: Hybrid rules with well-founded semantics. Knowl. Inf. Syst. 25(1), 137–168 (2010)
12. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artif. Intell. 172(12-13), 1495–1539 (2008)
13. Elenius, D.: Swrl-iq: A prolog-based query tool for owl and swrl. In: Klinov, P., Horridge, M. (eds.) OWLED. CEUR Workshop Proceedings, vol. 849. CEUR-WS.org (2012)
14. Gelder, A.V., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. J. ACM 38(3), 620–650 (1991)
15. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A tableau calculus for a nonmonotonic extension of $\mathcal{EL}^{\perp}$. In: Brünnler, K., Metcalfe, G. (eds.) TABLEAUX. LNCS, vol. 6793, pp. 180–195. Springer (2011)
16. Gomes, A.S., Alferes, J.J., Swift, T.: Implementing query answering for hybrid MKNF knowledge bases. In: Carro, M., Peña, R. (eds.) PADL. LNCS, vol. 5937, pp. 25–39. Springer (2010)
17. Heymans, S., Eiter, T., Xiao, G.: Tractable reasoning with dl-programs over datalog-rewritable description logics. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) ECAI. Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 35–40. IOS Press (2010)
18. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of $\mathcal{EL}$ ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC. LNCS, vol. 7032. Springer (2011)
19. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible elk. Tech. rep., University of Oxford (2013), submitted to a journal, available from http://elk.semanticweb.org/papers/elk-journal-2013.pdf
20. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artif. Intell. 175(9–10), 1528–1554 (2011)
21. Knorr, M., Alferes, J.J.: Querying owl 2 ql and non-monotonic rules. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N.F., Blomqvist, E. (eds.) ISWC (1). LNCS, vol. 7031, pp. 338–353. Springer (2011)
22. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Mylopoulos, J., Reiter, R. (eds.) IJCAI. pp. 381–386. Morgan Kaufmann (1991)
23. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (5 February 2013), available at http://www.w3.org/TR/owl2-profiles/
24. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)
25. Patel, C., Cimino, J.J., Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: Matching patient records to clinical trials using ontologies. In: Aberer, K., Choi, K.S., Noy, N.F., Allemang, D., Lee, K.I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ISWC/ASWC. LNCS, vol. 4825, pp. 816–829. Springer (2007)
26. Xiao, G., Eiter, T., Heymans, S.: The DReW system for nonmonotonic dl-programs. In: SWWS 2012. Springer Proceedings in Complexity, Springer (2013)