Next Step for NoHR: OWL 2 QL

Nuno Costa, Matthias Knorr and João Leite

NOVA LINCS, Departamento de Informática, Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa

Abstract. The Protégé plug-in NoHR allows the user to combine an OWL 2 EL ontology with a set of non-monotonic (logic programming) rules – suitable, e.g., to express defaults and exceptions – and query the combined knowledge base (KB). The formal approach realized in NoHR is polynomial (w.r.t. data complexity) and it has been shown that even very large health care ontologies, such as SNOMED CT, can be handled. As each of the tractable OWL profiles is motivated by different application cases, extending the tool to the other profiles is of particular interest, also because these preserve the polynomial data complexity of the combined formalism. Yet, a straightforward adaptation of the existing approach to OWL 2 QL turns out to not be viable. In this paper, we provide the non-trivial solution for the extension of NoHR to OWL 2 QL by directly translating the ontology into rules without any prior classification. We have implemented our approach and our evaluation shows encouraging results.

1 Introduction

NoHR[1] is a plug-in for the ontology editor Protégé[2] that allows its users to query combinations of $\mathcal{EL}^+_\bot$ ontologies and non-monotonic rules in a top-down manner.

Its motivation stems from the fact that many ontologies, such as the very large health care ontologies widely used in the area of medicine, e.g., SNOMED CT,[3] are expressed in OWL 2 EL, one of the OWL 2 profiles [24], and its underlying description logic (DL) $\mathcal{EL}^{++}$ [4]. Yet, due to their monotonic semantics, i.e., previously drawn conclusions persist when new additional information is adopted, DL-based ontology languages [3] are not suitable to model defaults and exceptions with a closed-world view, a frequently requested feature, e.g., when matching patient records to clinical trial criteria [26].

Among the plethora of approaches for extending DLs with non-monotonic features and deal with this problem (c.f. related work in [9,25]), NoHR builds on (Hybrid) MKNF KBs [25], which are based on the logic of minimal knowledge and negation as failure (MKNF) [23], under their well-founded semantics [18], a formalism that combines DLs and non-monotonic rules as known from Logic Programming.

This choice is motivated, on the one hand, by the fact that non-monotonic logic programming rules 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 [17], the other expressive language for the Semantic Web whose standardization

[1] http://centria.di.fct.unl.pt/nohr/
[2] http://protege.stanford.edu
[3] http://www.ihtsdo.org/snomed-ct/

is driven by the W3C.[4] On the other hand, MKNF KBs provide a very general and flexible framework for combining DL ontologies and non-monotonic rules (see [25]). In addition, [18], which is a variant of [25] based on the well-founded semantics [10] for logic programs, has a (lower) polynomial data complexity and is amenable for applying top-down query procedures, such as **SLG**($\mathcal{O}$) [1], to answer queries based only on the information relevant for the query, i.e., without computing the entire model.

NoHR is thus applicable to combinations of non-monotonic rules and OWL 2 EL ontologies. However, other applications (see, e.g., [6,27]) require ontologies using DL constructors which are not covered by OWL 2 EL, such as concept and role negation or role inverses – adding these to OWL 2 EL would raise its polynomial complexity [4].

OWL 2 QL and the $DL\text{-}Lite$ family [5,2] to which the DL underneath OWL 2 QL belongs, $DL\text{-}Lite_R$, is suitable in these cases and has recently drawn a lot of attention in research and in applications. Even though a simple language at first glance, it is expressive enough to capture basic ontology languages, conceptual data models, e.g., Entity-Relationship, and object-oriented formalisms, e.g., basic UML class diagrams. Reasoning focuses on answering queries by rewriting the initial query, with the help of the ontology, into a set of queries that can be answered using an industry-strength SQL engine over the data. This yields that query answering in OWL 2 QL is in LogSpace (more precisely AC^0), but also links directly to applications in ontology-based data access (OBDA) [6,20]. Altogether, OWL 2 QL is naturally tailored towards huge datasets.

To also provide such OWL 2 QL based applications with the additional expressive power obtained from combining DL ontologies with non-monotonic rules, in this paper, we extend NoHR to deal with the OWL 2 QL profile. Whereas, at first sight, this could seem like a routine exercise, to the best of our knowledge, there is currently no dedicated open-source OWL 2 QL classifier with OWL API available that also classifies negative concepts (similar to the NI-closure in [5], but whose direct adaptation would potentially introduce a huge number of additional axioms). Thus, since we cannot simply replace the reasoner ELK [16], used currently in NoHR for $\mathcal{EL}$, with a correspondent for $DL\text{-}Lite_R$, we translate the ontology directly into rules. This introduces some non-trivial problems such as the need to capture unsatisfiable concepts and roles, and irreflexive roles (covered in [5] also by the NI-closure). We solve this problem by introducing an extension of the graph, used e.g., for classification in OWL QL [22], to negative axioms, which is already a contribution in its own right. The resulting translation is implemented as a module of NoHR, and its performance evaluated. Our main contributions are:

- A procedure for translating $DL\text{-}Lite_R$ ontologies into rules which allows answering queries over MKNF KBs combining such ontologies and non-monotonic rules;
- A substantial extension of the Protégé plug-in NoHR to include OWL 2 QL ontologies, beyond $DL\text{-}Lite_R$ via normalizations, including optimizations on the number of created rules and the use of tabling in the top-down query engine XSB;[5]
- An evaluation of our extension that shows that NoHR for OWL 2 QL maintains all positive evaluation results of the OWL 2 EL version [13], and is even faster during pre-processing, as no classification is necessary, in exchange for a slightly longer average response time during querying.

[4] http://www.w3.org

[5] http://xsb.sourceforge.net

The remainder of the paper is structured as follows. In Sect. 2, we briefly recall $DL\text{-}Lite_R$ and MKNF KBs as a tight combination of the former DL and non-monotonic rules, followed, in Sect. 3, by the translation of $DL\text{-}Lite_R$ ontologies into rules. In Sect. 4, we discuss the changes made in the implementation for OWL 2 QL including optimizations, and evaluate it in Sect. 5, before we conclude in Sect. 6.

2 Preliminaries

2.1 $DL\text{-}Lite_R$

The description logic underlying OWL QL is $DL\text{-}Lite_R$, one language of the $DL\text{-}Lite$ family [5,2], which we recall following the presentation in [19].

The syntax of $DL\text{-}Lite_R$ is based on three disjoint sets of *individual names* $\mathsf{N_I}$, *concept names* $\mathsf{N_C}$, and *role names* $\mathsf{N_R}$. *Complex concepts* and *roles* can be formed according to the following grammar

$$B \rightarrow A \mid \exists Q \qquad C \rightarrow B \mid \neg B \qquad Q \rightarrow P \mid P^- \qquad R \rightarrow Q \mid \neg Q$$

where $A \in \mathsf{N_C}$ is a concept name, $P \in \mathsf{N_R}$ a role name, and P^- its inverse. We also call B a *basic concept*, Q a *basic relation*, C a *general concept* and R a *general role*.

A $DL\text{-}Lite_R$ knowledge base $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The TBox contains *general inclusion axioms (GCI)* of the form $B \sqsubseteq C$ and *role inclusion axioms* (RI) of the form $Q \sqsubseteq R$, with B, C, Q, and R defined as above. We term *positive inclusion axioms* all GCIs and RIs in $\mathcal{O}$ such that C is a basic concept and R is a basic relation, respectively, and all other GCIs and RIs *negative inclusion axioms*. We also assume that Q^- denotes the role P if $Q = P^-$, and P^- if $Q = P$. The ABox contains assertions of the form $A(a)$ and $P(a, b)$ where $A \in \mathsf{N_C}$, $P \in \mathsf{N_R}$, and $a, b \in \mathsf{N_I}$. Assertions $C(a)$ for general concepts C can be included by $A \sqsubseteq C$ and $A(a)$ for a new concept name A.

The semantics of $DL\text{-}Lite_R$ is based on *interpretations* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consisting of a nonempty interpretation domain $\Delta^\mathcal{I}$ and an interpretation function $\cdot^\mathcal{I}$ that assigns to each individual a a distinct[6] element $a^\mathcal{I}$ of $\Delta^\mathcal{I}$, to each concept name A a subset $A^\mathcal{I}$, and to each role name P a binary relation $P^\mathcal{I}$ over $\mathcal{I}$. This can be extended as usual:

$$(P^-)^\mathcal{I} = \{(i_2, i_1) \mid (i_1, i_2) \in P^\mathcal{I}\} \qquad (\neg B)^\mathcal{I} = \Delta^\mathcal{I} \setminus B^\mathcal{I}$$
$$(\exists Q)^\mathcal{I} = \{i \mid (i, i') \in Q^\mathcal{I}\} \qquad (\neg Q)^\mathcal{I} = \Delta^\mathcal{I} \times \Delta^\mathcal{I} \setminus Q^\mathcal{I}$$

An interpretation $\mathcal{I}$ is a *model of GCI* $B \sqsubseteq C$ and *of RI* $Q \sqsubseteq R$ if $B^\mathcal{I} \subseteq C^\mathcal{I}$ and $Q^\mathcal{I} \subseteq R^\mathcal{I}$ respectively. $\mathcal{I}$ is also a *model of an assertion* $A(a)$ ($P(a, b)$) if $a^\mathcal{I} \in A^\mathcal{I}$ ($(a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I}$). Given an axiom/assertion α we denote by $\mathcal{I} \models \alpha$ that $\mathcal{I}$ is a model of α. A *model* of a $DL\text{-}Lite_R$ KB $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \alpha$ holds for all $\alpha \in \mathcal{T} \cup \mathcal{A}$, and $\mathcal{O}$ is *satisfiable* if it has at least one model, and *unsatifiable* otherwise. Also, $\mathcal{O}$ *entails* axiom α, written $\mathcal{O} \models \alpha$, if every model of $\mathcal{O}$ satisfies α.

[6] Hence, the unique name assumption is applied and, as shown in [2], dropping it would increase significantly the computational complexity of $DL\text{-}Lite_\mathcal{R}$.

2.2 MKNF Knowledge Bases

MKNF knowledge bases (KBs) build on the logic of minimal knowledge and negation as failure (MKNF) [23]. Two main different semantics have been defined [25,18], and we focus on the well-founded version [18], due to its lower computational complexity and amenability to top-down querying without computing the entire model. Here, we only point out important notions following [13], and refer to [18] and [1] for the details.

We start by recalling MKNF knowledge bases as presented in [1] to combine an 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 $P(t_1, \ldots, t_n)$ *s.t. P occurs in $\mathcal{O}$ is called* DL-atom*; otherwise* 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 the following MKNF knowledge base $\mathcal{K}$ for recommending CDs, adapted from [18] (with some modifications). We denote DL-atoms and constants with upper-case names and non-DL-atoms and variables with lower-case names.[7]

$$\begin{aligned}
\exists HasArtist^- &\sqsubseteq Artist & Piece &\sqsubseteq \exists HasArtist\\
\exists HasComposed^- &\sqsubseteq Piece & Artist &\sqsubseteq \neg Piece\\
HasComposed^- &\sqsubseteq HasArtist
\end{aligned}$$

$$\begin{aligned}
recommend(x) \leftarrow & Piece(x), \mathbf{not}\ owns(x), \mathbf{not}\ lowEval(x), interesting(x).\\
interesting(x) \leftarrow & Piece(x), \mathbf{not}\ owns(x), Piece(y), owns(y),\\
& Artist(z), HasArtist(y, z), HasArtist(x, z).\\
owns(Summertime). \qquad & HasArtist(Summertime, Gershwin).\\
Piece(Summertime). \qquad & HasComposed(Gershwin, RhapsodyInBlue).
\end{aligned}$$

This example shows that we can seamlessly express defaults and exceptions, such as recommending pieces as long as they are not owned or having a low evaluation, and at the same time taxonomic/ontological knowledge including information over unknown individuals, such as every piece having at least one artist without having to specify whom, but also features of $DL\text{-}Lite_R$, such as domain and range restrictions (of roles).

[7] To ease readability, we omit the auxiliary atoms that ensure DL-safety and leave them implicit. Also, whenever the body of a rule is empty, we dub it a *fact* and omit the $\leftarrow$ occasionally.

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 a rule of the form $\mathbf{K}H \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n, \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_m$, and $\pi(\mathcal{P})$ 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 [18], 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 $\mathcal{K}$ [18], and, like in [1], 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 ontology alone is unsatisfiable, or by the combination of appropriate axioms in $\mathcal{O}$ and rules in $\mathcal{P}$, e.g., axiom $A \sqsubseteq \neg B$ in $\mathcal{O}$, and facts $A(a)$ and $B(a)$ in $\mathcal{P}$. Strictly speaking, unlike [13], we do not have to make assumptions on the satisfiability of $\mathcal{O}$ as we are not going to use a classifier when processing $DL\text{-}Lite_R$ ontologies. Still, for the technical results established in Sec. 3, we will rely on satisfiability since we are able to entail everything from an unsatisfiable $\mathcal{O}$, whereas the translation into rules defined in Sec. 3 would not permit that. This is why, in the following, we assume that $\mathcal{O}$ occurring in $\mathcal{K}$ is satisfiable, which does not truly constitute a restriction as we can always turn the ABox into rules without any effect on $\mathsf{M}_{\mathsf{wf}}(\mathcal{K})$. An alternative approach would be to use one of the paraconsistent semantics for MKNF knowledge bases [15], but this is outside the scope of this paper, and an issue for future work, as no paraconsistent correspondence to the querying procedure **SLG($\mathcal{O}$)** used here currently exists.

2.3 Querying in MKNF Knowledge Bases

In [1], 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 [7] 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 [1] 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* $Q = q(\boldsymbol{X}) \leftarrow A_1, \ldots, A_n, \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_m$, 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 [18]), 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 [1]). 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 necessary in general, but we can use $\mathcal{K}$ here if it contains no negative inclusion axioms.

In [1], 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. As this suffices for our purposes, we omit $\mathcal{K}^d$ and restrict ourselves to $\mathcal{K}$ here. Consider query $q = recommend(Summertime)$. There is a matching rule head in $\mathcal{K}$, and, by instantiating the rule body with $x = Summertime$, we obtain a new set of queries. The first one, $Piece(Summertime)$, can be answered by means of the rule with matching head. The second, $\mathbf{not}\ owns(Summertime)$, is handled by querying for $owns(Summertime)$, for which also exists a corresponding rule, which means that $\mathbf{not}\ owns(Summertime)$ fails, so q is false.

Consider $q_1 = recommend(RhapsodyInBlue)$. We can use the same rule with matching rule head and, again, obtain four new instantiated queries from the rule body. Now, $Piece(RhapsodyInBlue)$ cannot be derived from the rules, but we can query the ontology and the oracle will return, e.g., a query $HasComposed(x_1, RhapsodyInBlue)$ that if proven true can be added to $\mathcal{O}$, which would allow us to derive the queried goal. Because of the fact $HasComposed(Gershwin, RhapsodyInBlue)$, this query succeeds, and so does $Piece(RhapsodyInBlue)$. Subsequently, neither $owns(RhapsodyInBlue)$ nor $lowEval(RhapsodyInBlue)$ can be proven, so both fail, and their (default) negated queries succeed. For the remaining new query $interesting(RhapsodyInBlue)$, the second rule head matches, which creates a further set of subgoals. The first two have just been answered, so have the next two with $y = Summertime$ for q, and it can be verified that the remaining also follow from the interplay of $\mathcal{O}$ and $\mathcal{P}$ in $\mathcal{K}$. Thus, q_1 succeeds.

We recall the notions of a complete and a (correct) partial oracle from [1].

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 $compT_\mathcal{O}$, is defined by $compT_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ 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 $pT_\mathcal{O}$, is a relation $pT_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ such that if $pT_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$, 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 $pT_\mathcal{O}$ is correct *w.r.t. $compT_\mathcal{O}$ iff, for all MKNF-consistent $\mathcal{K}^d$, replacing $compT_\mathcal{O}$ in* **SLG($\mathcal{O}$)** *with $pT_\mathcal{O}$ 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. Also, 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 [1]).

3 Translating the Ontology into Rules

As argued for the case of $\mathcal{EL}^+_\bot$ [13], axioms with $\exists$ on the right-hand side, e.g., $Piece \sqsubseteq \exists HasArtist$, cannot be translated straightforwardly into rules, nor do they directly contribute to the result when querying for ground instances, e.g., of $HasArtist(x, y)$. Still,

such axioms may contribute to derivations within $\mathcal{O}$, which is why, in [13], classification using the dedicated and highly efficient $\mathcal{EL}$ reasoner ELK [16] is first applied to derive implicit consequences. These, together with all axioms in $\mathcal{O}$, are then translated into rules, now discarding certain axioms with $\exists$ on the right-hand side.

Since, to the best of our knowledge, no dedicated and open-source OWL 2 QL classifier with OWL API that also classifies negative concepts is currently available, we translate the ontology directly into rules. This also simplifies and shortens the preprocessing phase and avoids a priori-classification, but requires some non-trivial considerations to ensure that no derivations are lost in the process, which we now explain.

Essentially, axioms, such as $Piece \sqsubseteq \exists HasArtist$, cannot be translated into a rule $HasArtist(x, y) \leftarrow Piece(x)$ using a universal variable y, as this would allow us to derive $HasArtist(x, y)$ for any $Piece(x)$ and y, which is clearly not what the axiom expresses. Using a new constant c instead of y would not be correct either, as querying for $HasArtist(x, y)$ would return $HasArtist(x, c)$ for any $Piece(x)$ for the same c. Therefore, we proceed differently by introducing new auxiliary predicates that intuitively represent the domain and range of roles. For our example, this will yield the rule $DHasArtist(x) \leftarrow Piece(x)$ where $DHasArtist$ stands for the domain of $HasArtist$ (and $RHasArtist$ its range). Using such auxiliary predicates also means that we have to make sure that, e.g., $HasArtist(Summertime, Gershwin)$ allows us to derive $DHasArtist(Summertime)$, which can be achieved via an additional rule $DHasArtist(x) \leftarrow HasArtist(x, y)$. Moreover, for $HasComposed^- \sqsubseteq HasArtist$, it does not suffice to translate the axiom to $HasArtist(x, y) \leftarrow HasComposed(y, x)$, but also link the new auxiliary predicates for both roles, through the addition of the rules $DHasArtist(x) \leftarrow RHasComposed(x)$ and $RHasArtist(x) \leftarrow DHasComposed(x)$.

We now formalize this translation, and start by introducing notation on how to translate general concepts and roles. For that purpose, we formally introduce for each role $P \in \mathsf{N_R}$ auxiliary predicates DP and RP with the intuition of representing the domain and range of P. Also, similar to previous work in [1,13], we use special atoms $NH(\boldsymbol{t_i})$ in **SLG($\mathcal{O}$)** that represent a query $\neg H(\boldsymbol{t_i})$ to the oracle. These are, of course, only relevant if $\mathcal{O}$ contains negative inclusion axioms.

Definition 5. *Let C be a concept, R a role, x and y variables, and v a new (anonymous) variable (disjoint from x and y). We define $tr(C, x)$ and $tr(R, x, y)$ as follows:*

$$tr(C,x) = \begin{cases} A(x) & \text{if } C = A \\ DP(x) & \text{if } C = \exists P \\ RP(x) & \text{if } C = \exists P^- \\ NA(x) & \text{if } C = \neg A \\ tr(\neg Q, x, v) & \text{if } C = \neg\exists Q \end{cases} \quad tr(R,x,y) = \begin{cases} P(x,y) & \text{if } R = P \\ P(y,x) & \text{if } R = P^- \\ NP(x,y) & \text{if } C = \neg P \\ NP(y,x) & \text{if } C = \neg P^- \end{cases}$$

We obtain $tr^d(C, x)$ and $tr^d(Q, x, y)$ from $tr(C, x)$ and $tr(Q, x, y)$ by substituting all predicates P in $tr(C, x)$ and $tr(Q, x, y)$ with P^d, respectively.

This way, $tr(C, x)$ and $tr(R, x, y)$ handle both positive and negative inclusions and no additional case distinction is necessary.

Before we present the actual translation, we need to introduce one central notion, namely a graph to represent the axioms in a given TBox $\mathcal{T}$ as well as the implicitly

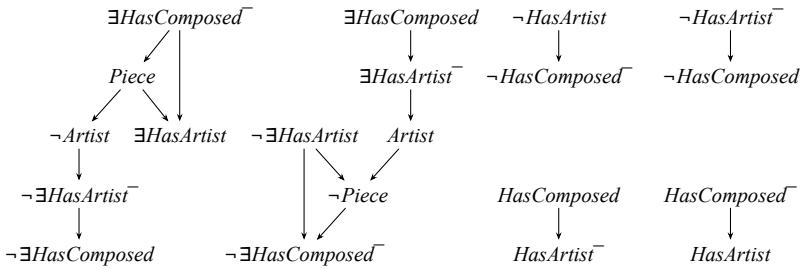

Fig. 1. The digraph $\mathcal{G}_\mathcal{T}$ for Example 2

derivable axioms, which will be necessary for defining the translation itself, but also turn out useful when establishing the correctness of the translation. Graphs have been used for classification in OWL QL (of positive inclusion axioms) [22], and we extend the notion here to also take negative inclusion axioms into account. We thus introduce the digraph (directed graph) of $\mathcal{T}$ as follows.

Definition 6. *Let $\mathcal{T}$ be a $DL\text{-}Lite_R$ TBox. The* digraph *of $\mathcal{T}$, $\mathcal{G}_\mathcal{T} = \langle \mathcal{V}, \mathcal{E} \rangle$, is constructively defined as follows.*

1. *If $A \in \mathsf{N}_C$, then A and $\neg A$ are in $\mathcal{V}$;*
2. *If $R \in \mathsf{N}_R$, then P, $\exists P$, $\exists P^-$, $\neg P$, $\neg\exists P$, and $\neg P^-$ are in $\mathcal{V}$;*
3. *If $B_1 \sqsubseteq B_2 \in \mathcal{T}$, then the edges (B_1, B_2) and $(\neg B_2, \neg B_1)$ are in $\mathcal{E}$;*
4. *If $Q_1 \sqsubseteq Q_2 \in \mathcal{T}$, then the edges (Q_1, Q_2), (Q_1^-, Q_2^-), $(\exists Q_1, \exists Q_2)$, $(\exists Q_1^-, \exists Q_2^-)$, $(\neg Q_2, \neg Q_1)$,$(\neg Q_2^-, \neg Q_1^-)$, $(\neg\exists Q_2, \neg\exists Q_1)$ e $(\neg\exists Q_2^-, \neg\exists Q_1^-)$ are in $\mathcal{E}$;*
5. *If $B_1 \sqsubseteq \neg B_2 \in \mathcal{T}$, then the edges $(B_1, \neg B_2)$ and $(B_2, \neg B_1)$ are in $\mathcal{E}$;*
6. *If $Q_1 \sqsubseteq \neg Q_2 \in \mathcal{T}$, then the edges $(Q_1, \neg Q_2)$, $(Q_2^-, \neg Q_1^-)$, $(\exists Q_1, \neg\exists Q_2)$, $(\exists Q_2, \neg\exists Q_1)$, $(\exists Q_1^-, \neg\exists Q_2^-)$ and $(\exists Q_2^-, \neg\exists Q_1^-)$ are in $\mathcal{E}$.*

Basically, each possible general concept and general role over $\mathsf{N_C}$ and $\mathsf{N_R}$ is a node in $\mathcal{G}_\mathcal{T}$, and the directed edges represent logical implications that follow from the axioms. Namely, for items 3. and 5., the subset inclusion itself and its contrapositive are in $\mathcal{E}$, and this is similar for items 4. and 6., only that the additional combinations due to inverses, $\exists$, and $\neg$ have to be taken into account. In this sense, the graph can be understood as capturing all subset inclusions (explicit and implicit) in $\mathcal{O}$, i.e., whenever there is a path from concept C_1 to concept C_2 and from role R_1 to role R_2, then $C_1 \sqsubseteq C_2$ and $R_1 \sqsubseteq R_2$ hold respectively. An Example of such a digraph is given in Fig. 1 for the TBox $\mathcal{T}$ from Example 2.

One observation w.r.t. Fig. 1, is that $\exists HasComposed \sqsubseteq \neg\exists HasComposed^-$, i.e., $HasComposed$ is irreflexive. Even though this does not entail any assertion, knowing that $\forall x.\neg HasComposed(x, x)$ does hold should be captured in the translation. We introduce $\Psi(\mathcal{T})$, the set of irreflexive roles in $\mathcal{T}$, to be able to ensure exactly that.

Definition 7. *Let $\mathcal{T}$ be a $DL\text{-}Lite_R$ TBox and $\mathcal{G}_\mathcal{T}$ its digraph. We define $\Psi(\mathcal{T})$ as the smallest set of all $P \in \mathsf{N_R}$ that satisfy at least one of the following conditions:*

1. *For some $B_1 \sqsubseteq \neg B_2 \in \mathcal{T}$, there exist paths from $\exists P$ to B_1 and from $\exists P^-$ to B_2;*
2. *For some $B_1 \sqsubseteq \neg B_2 \in \mathcal{T}$, there exist paths from $\exists P^-$ to B_1 and from $\exists P$ to B_2;*
3. *For some $Q_1 \sqsubseteq \neg Q_2 \in \mathcal{T}$, there exist paths from P to Q_1 and from P^- to Q_2;*
4. *For some $Q_1 \sqsubseteq \neg Q_2 \in \mathcal{T}$, there exist paths from P^- to Q_1 and from P to Q_2.*

This notion builds on $\mathcal{G}_\mathcal{T}$, which is also required for detecting a further set of derivations. Imagine we would (wrongfully) add $Artist \sqsubseteq \exists HasComposed^-$ to $\mathcal{O}$ in Example 2. Then there would be a path from $Artist$ to both $Piece$ and $\neg Piece$, i.e., the concept $Artist$ would be unsatisfiable. Note that independently of whether the MKNF KB is MKNF-inconsistent or not, we need to make sure that all unsatisfiable concepts and roles are determined, so we introduce $\Omega(\mathcal{T})$, quite similar in spirit to $\Psi(\mathcal{T})$.

Definition 8. *Let $\mathcal{T}$ be a $DL\text{-}Lite_R$ TBox and $\mathcal{G}_\mathcal{T}$ its digraph. We define $\Omega(\mathcal{T})$ as the smallest set of all $A \in \mathsf{N_C}$ such that, for some $B_1 \sqsubseteq \neg B_2 \in \mathcal{T}$, there exist paths from A to both B_1 and B_2, and all $P \in \mathsf{N_R}$ that satisfy at least one of the following conditions:*

1. *For some $B_1 \sqsubseteq \neg B_2 \in \mathcal{T}$, there exist paths from $\exists P$ to both B_1 and B_2;*
2. *For some $B_1 \sqsubseteq \neg B_2 \in \mathcal{T}$, there exist paths from $\exists P^-$ to both B_1 and B_2;*
3. *For some $Q_1 \sqsubseteq \neg Q_2 \in \mathcal{T}$, there exist paths from P to both Q_1 and Q_2;*
4. *For some $Q_1 \sqsubseteq \neg Q_2 \in \mathcal{T}$, there exist paths from P^- to both Q_1 and Q_2.*

With all pieces in place, we can introduce the translation of a $DL\text{-}Lite_R$ ontology.

Definition 9. *Let $\mathcal{O}$ be a $DL\text{-}Lite_R$ ontology. We define $\mathcal{P}^d_\mathcal{O}$ from $\mathcal{O}$, where B_1, B_2 are basic concepts, Q_1, Q_2 basic roles, x, y variables, and a, b individuals, as the smallest set containing:*

(e) *for every $P \in \mathsf{N_R}$:*

$DP(x) \leftarrow P(x,y).$ $\quad DP^d(x) \leftarrow P^d(x,y).$

$RP(y) \leftarrow P(x,y).$ $\quad RP^d(y) \leftarrow P^d(x,y).$

(a1) *for every $A(a) \in \mathcal{O}$:*

$A(a) \leftarrow .$ $\quad A^d(a) \leftarrow \mathbf{not}\ NA(a).$

(a2) *for every $P(a,b) \in \mathcal{O}$:*

$P(a,b) \leftarrow .$ $\quad P^d(a,b) \leftarrow \mathbf{not}\ NP(a,b).$

(s1) *for every $B_1 \sqsubseteq B_2 \in \mathcal{O}$:*

$tr(B_2,x) \leftarrow tr(B_1,x).$ $\quad tr^d(B_2,x) \leftarrow tr^d(B_1,x), \mathbf{not}\ tr(\neg B_2,x).$

$tr(\neg B_1,x) \leftarrow tr(\neg B_2,x).$

(s2) *for every $Q_1 \sqsubseteq Q_2 \in \mathcal{O}$:*

$tr(Q_2,x,y) \leftarrow tr(Q_1,x,y).$ $\quad tr^d(Q_2,x,y) \leftarrow tr^d(Q_1,x,y), \mathbf{not}\ tr(\neg Q_2,x,y).$

$tr(\exists Q_2,x) \leftarrow tr(\exists Q_1,x).$ $\quad tr^d(\exists Q_2,x) \leftarrow tr^d(\exists Q_1,x), \mathbf{not}\ tr(\neg\exists Q_2,x).$

$tr(\exists Q_2^-,x) \leftarrow tr(\exists Q_1^-,x).$ $\quad tr^d(\exists Q_2^-,x) \leftarrow tr^d(\exists Q_1^-,x), \mathbf{not}\ tr(\neg\exists Q_2^-,x).$

$tr(\neg Q_1,x,y) \leftarrow tr(\neg Q_2,x,y).$

(n1) *for every $B_1 \sqsubseteq \neg B_2 \in \mathcal{O}$:*

$tr(\neg B_1,x) \leftarrow tr(B_2,x).$ $\quad tr(\neg B_2,x) \leftarrow tr(B_1,x).$

(n2) *for every $Q_1 \sqsubseteq \neg Q_2 \in \mathcal{O}$:*

$tr(\neg Q_2,x,y) \leftarrow tr(Q_1,x,y).$ $\quad tr(\neg Q_1,x,y) \leftarrow tr(Q_2,x,y).$

(i1) *for every $A \in \Omega(\mathcal{T})$:* $\quad NA(x) \leftarrow .$

(i2) *for every $P \in \Omega(\mathcal{T})$:* $\quad NP(x,y) \leftarrow .$

(ir) *for every $P \in \Psi(\mathcal{T})$:* $\quad NP(x,x) \leftarrow .$

Item **(e)** ensures that the domain and range of roles is correctly encoded, items **(a1)** and **(a2)** translate the ABox, items **(s1)** and **(s2)** the positive inclusions, items **(n1)** and **(n2)** the negative inclusions, and items **(i1)**, **(i2)**, and **(ir)** introduce the rules representing unsatisfiable concepts and unsatisfiable and irreflexive roles. Note, that $\mathcal{P}^d_\mathcal{O}$ contains the rule representation for both $\mathcal{O}$ and $\mathcal{O}^d$, which is why items **(e)**–**(s2)** contain doubled rules. Of course, if $\mathcal{O}$ does not contain negative inclusion axioms, then we can skip all these, as well as items **(n1)**–**(ir)** which will not contribute anything anyway in this case. The additional default atoms are added to the doubled rules to be in line with the idea of the doubling of rules in [1]: whenever, e.g., $A(x)$ is "classically false" for some x, i.e., $NA(x)$ holds, then we make sure that $A^d(x)$ is derivable 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 neither **(n1)**–**(ir)** nor the contrapositives in **(s1)** and **(s2)** do produce the doubled counterparts: atoms based on predicates of the forms NC^d or NR^d are not used anywhere. Finally, the doubled rules in **(e)** do not contain the default negated atom as this case does really just associate domain and range to a role assertion, either present in the ABox or derived elsewhere. Additionally, predicates NDP or NRP are not used anywhere, so such default negated atoms would be of no impact anyway.

We can establish three correspondences between entailment from satisfiable $\mathcal{O}$ and the program resulting from the translation $\mathcal{P}^d_\mathcal{O}$. First, we consider positive atoms.

Lemma 10. *Let $\mathcal{O}$ be a $DL\text{-}Lite_R$ ontology, A a unary and R a binary predicate:*

- $\mathcal{O} \models A(a)$ *iff* $\mathcal{P}^d_\mathcal{O} \models A(a)$ *and* $\mathcal{O} \models R(a,b)$ *iff* $\mathcal{P}^d_\mathcal{O} \models R(a,b)$.

A similar property holds for (classically) negated atoms.

Lemma 11. *Let $\mathcal{O}$ be a $DL\text{-}Lite_R$ ontology, A a unary and R a binary predicate:*

- $\mathcal{O} \models \neg A(a)$ *iff* $\mathcal{P}^d_\mathcal{O} \models NA(a)$ *and* $\mathcal{O} \models \neg R(a,b)$ *iff* $\mathcal{P}^d_\mathcal{O} \models NR(a,b)$.

We can also show the correspondent to Lemma 10 for the doubled predicates.

Lemma 12. *Let $\mathcal{O}$ be a $DL\text{-}Lite_R$ ontology, A a unary and R a binary predicate:*

- $\mathcal{O}^d \models A^d(a)$ *iff* $\mathcal{P}^d_\mathcal{O} \models A^d(a)$ *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 13. *Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled MKNF KB and $pT^{QL}_\mathcal{O}$ a* partial QL oracle *such that $pT^{QL}_\mathcal{O}(\mathcal{I}, S, \mathcal{L})$ iff $\mathcal{P}^d_\mathcal{O} \cup \mathcal{I} \cup \mathcal{L} \models S$. Then $pT^{QL}_\mathcal{O}$ is a correct partial oracle w.r.t. $compT_\mathcal{O}$.*

Instead of coupling two rule reasoners that interact with each other using an oracle, we can integrate both into one rule reasoner. The resulting approach is polynomial w.r.t. data complexity (as in [1,13], but not in AC^0 any longer as for OWL 2 QL alone).

Theorem 14. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be an MKNF KB with $\mathcal{O}$ in $DL\text{-}Lite_R$. An* **SLG($\mathcal{O}$)** *evaluation of a query in $\mathcal{K}_{QL} = (\emptyset, (\mathcal{P}^d \cup \mathcal{P}^d_\mathcal{O}))$ is decidable with data complexity in* PTIME.

	q2	q3	q4	q5	q6	q7	q8	q9	q10	q13	q14
NoHR 1	5	5	20	200	1543	124	4538	324	4	2	1328
Pellet 1	24	15	17	14	33	13	67	104	14	13	30
NoHR 9	606	14	191	5745	58150	1455	54435	32525	18	8	67516
Pellet 9	148	108	96	96	358	102	123	524	99	97	177
NoHR 20	3602	29	308	5158	109898	3891	119744	90314	43	19	127772
Pellet 20	427	477	246	247	726	249	265	1329	249	249	469

Fig. 2. Query response times for NoHR and Pellet

4 System Description

In this section, we briefly describe the changes to the architecture of our plug-in and discuss some optimizations implemented w.r.t. the translation described in Sec. 3.

To allow the usage of OWL QL ontologies, changes were essentially made in the translator. Since NoHR now supports two OWL profiles a switch was introduced that checks the profile of the loaded/edited ontology. Whenever it belongs to OWL EL, NoHR behaves as described in [13], i.e., the reasoner ELK is used to classify the ontology and return the inferred axioms to translator, which are then translated. Otherwise, we treat $\mathcal{O}$ of the hybrid KB based on the translation described in Sec. 3 for OWL QL.

Notably, in Sec. 3, we only considered $DL\text{-}Lite_R$, while OWL QL includes a number of additional constructs which often can be expressed in $DL\text{-}Lite_R$. To account for that, we first normalize such expressions to axioms in $DL\text{-}Lite_R$. This includes ignoring certain expressions, most of which do not contribute to derivations, e.g., `SubClassOf(B owl:Thing)`, while others make the ontology unsatisfiable, such as `ClassAssertion(owl:Nothing a)`, although, as mentioned before, with no effect when querying the translated rules.

Subsequently, the graph is constructed, for determining unsatisfiable concepts and unsatisfiable and irreflexive roles, after which the translation is performed, which includes a number of optimizations. First, whenever there are no negative inclusions, the doubled rules are omitted in the cases **(e)**–**(s2)** of Def. 9. Additionally, case **(e)** is limited to those rules whose heads appear in the body of another rule. Both steps reduce the overall number of rules created during the translation.

The second group of optimizations is related to tabling in XSB, which contributes to help answering queries very efficiently in a top-down manner, and avoid infinite loops while querying. However, simply declaring all predicates to be tabled is very memory-consuming, so we reduced the number of tabled predicates without affecting loop detection. For example, only predicates that appear in any rule head and under negation in any rule body need to be tabled. In addition, rules with an empty body (facts) can be ignored in the previous criterion, as these will never cause infinite loops.

5 Evaluation

In this section, we evaluate our system and show that a) our system scales reasonably well for OWL query answering (only being considerably slower for memory-intensive

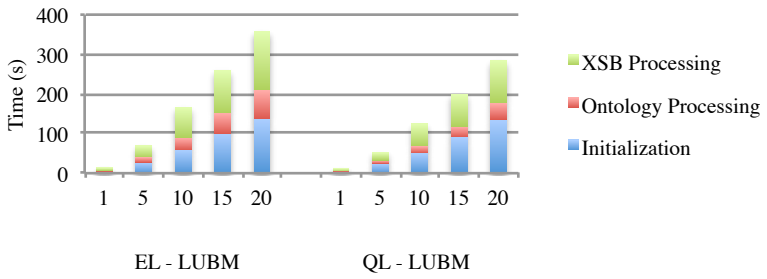

Fig. 3. Preprocessing time for LUBM for the two translation modes

cases), b) preprocessing is even faster when compared to NoHR's previous version using a classifier (for EL), which was already capable of preprocessing large ontologies in a short period of time, c) querying scales well, even for over a million facts/assertions in the ABox, despite being slightly slower on average in comparison to EL, and d) adding rules scales linearly for pre-processing and querying, even for an ontology with many negative inclusions.

All tests were performed on a MacBook Pro (Retina, 13-inch, Early 2015) under OS X Yosemite 10.10.4 with 2.9 GHz Intel Core i5 processor and 16 GB of 1867 MHz DDR 3 memory. We ran all tests with a terminal version of NoHR with max. 8 GB of RAM allocated to Java 8 and we used XSB 3.6.0 for querying with the remaining RAM. Test results are averages over 5 runs.

We considered LUBM[8] [12], a standard benchmark for evaluating queries over a large data set. The benchmark's ontology contains 43 classes, 25 object and 7 data properties and 243 axioms, and it comes with a data generator and 14 queries q_1–q_{14}. First, to test general scalability, we utilized the material[9] in [21], that provides data instances of LUBM$_n$ for $n = 1, 9, 20$, where n specifies the number of universities and where LUBM is slightly simplified to fall completely into the QL profile. For our test, we focused on the provided material for Pellet,[10] as it worked correctly right away. Regarding pre-processing we observe that NoHR is slightly slower than Pellet (with the factor varying between 1.6 and 6.2), mainly due to the time of additionally loading the file in XSB, a step not necessary for Pellet. The results of answering queries q_2–q_{10}, q_{13}, and q_{14} can be found in Fig. 2.[11] We observe that NoHR is faster for some queries (q_3, q_{10}, q_{13} – up to factor 16), and slower for others, either below factor 15 (q_2, q_4, q_7), or with a significant difference (the remainder). The latter occurs due to the huge amount of data being stored in XSB's tables in the query process, ultimately intended for handling non-monotonic rules that are not even part of Pellet. Yet, at the same time,

[8] http://swat.cse.lehigh.edu/projects/lubm/
[9] https://github.com/ontop/iswc2014-benchmark
[10] https://github.com/complexible/pellet
[11] q_1 is flawed for Pellet and the other two queries have been omitted here, as the restriction to QL cancels the OWL reasoning capability intended to be tested (transitivity and realization).

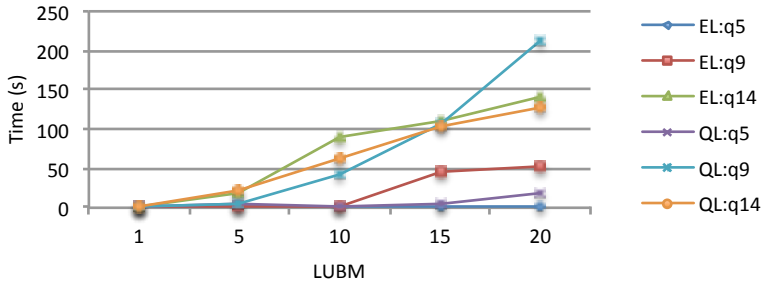

Fig. 4. Query time for three LUBM queries

tabling enables NoHR to be faster, namely, when an already computed result can simply be looked up (see the test below on LIPID for further details).

Next, with the aim of comparing our new approach, based on a direct translation, with the one using a classifier (for OWL EL), we created instances of LUBM_n with $n = 1, 5, 10, 15, 20$ using the provided generator, and a restricted version of LUBM which fits both OWL EL and QL (thus rendering q_{13} meaningless, but now permitting q_1 in exchange), with the number of assertions ranging from roughly 100,000 to over 2,700,000. We performed pre-processing and the results for both kinds of translators (EL and QL) can be found in Fig. 3. Note that "Initialization" includes loading the ontology and for EL also classifying it, "Ontology Processing" includes the actual translation, and "XSB Processing" the writing of the rule file and loading it in XSB. We observe that QL is considerably faster, indeed up to 80s for LUBM_{20}, which is to a considerable extent due to avoiding classification and a smaller rule file being created. Besides that, the preprocessing time increases linearly, and the overall time for preprocessing is acceptable in our opinion as this is only done once before querying.

We also queried in XSB for both versions, EL and QL. Some representative results are shown in Fig. 4. Basically, for queries q_1–q_5, q_7 and q_{10} the response time is below 18s, often strictly below 1s, in general slightly in favor of the EL version (up to factor 8). For the other queries, response time increases more significantly with huge amounts of data, divided into those slightly in favor of QL (q_6, q_8, q_{14}, with a factor below 2, but up to 20s in absolute value), and those in favor of EL (q_9, up to factor 4 and 150s in absolute value). In all cases, the response time grows linearly w.r.t. the increasing size of data, and querying in QL is slightly slower on average. Here, EL compensates for the longer preprocessing, and it thus seems that deciding which of the two forms of translations performs better depends on the kind (and number) of queries we pose.

Finally, with the aim of also testing a more expressive OWL 2 QL ontology, we used the LIPID ontology,[12] which has, besides 749 subclass axioms, 1,486 class disjointness axioms and 20 inverse object properties in combination with non-monotonic

[12] http://bioonto.dcs.aber.ac.uk/ql-ont/

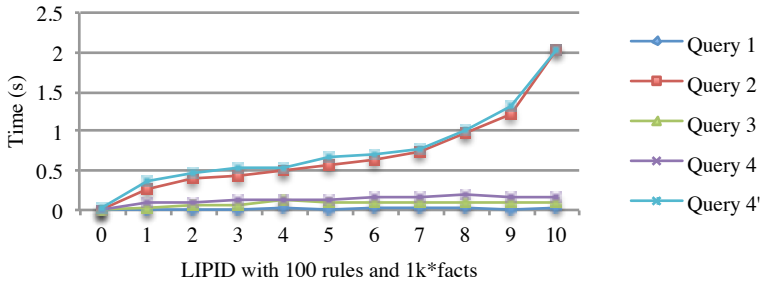

Fig. 5. Query time for LIPID

rules. The latter were created by means of the rule generator previously used in [13], containing a fixed number of 100 rules and a number of facts increasing in steps of 1k, also introducing some new predicates not present in the ontology itself. We performed the preprocessing step and observed only small effects due to the increasing amount of rules. The time for processing the ontology was naturally stable for all steps, and overall processing time was between 1.4 and 3.3s. Notably, the considerable amount of negative inclusions had no significant impact on time, e.g., when constructing the graph. Then, we posed four simple queries (Query1–4), namely Acyl_Ester_Chain(X), Lipid(X), Organic_Group(X), and Entity(X) to the resulting rule sets in XSB, with the position in the concept hierarchy varying from the lowest level (Query 1) to the top-most below $\top$ (Query 4) with 715 subclasses. The results are shown in Fig. 5. As we can see, the response time is very reasonable, from well below 1s to at most 2.2s. We also posed Query 4 without posing the three previous queries beforehand. The result is also included in Fig. 5 as Query 4', and it shows the speed-up that tabling of prior query results for subclasses has on the response time of Query 4 (up to factor 11.5). Overall, the results somehow also show the effect of the arbitrary rules in raising the time for query answering, since they introduce additional non-hierarchical (positive and negative) links within the ontology. We conclude with noting that performance tests of querying (non-monotonic) rules and ontologies would considerably benefit from real datasets, but, unfortunately, to the best of our knowledge, none are currently available.

6 Conclusions

We have extended NoHR, the Protégé plug-in that allows to query non-monotonic rules and ontologies in OWL 2 EL, to also admit ontologies in OWL 2 QL. While the principal architecture of the tool remains the same, the crucial module that translates the ontology into rules with the help of a classifier simply cannot be re-used, which is why we introduced a novel direct translation for OWL 2 QL ontologies to cover this profile. We have implemented this translation and discussed optimizations. The evaluation shows that it maintains all positive evaluation results of the OWL 2 EL version [13],

and is even faster during pre-processing, as no classification is necessary, in exchange for an on average slightly longer response time during querying.

Besides the OWL 2 EL profile supported by NoHR, and compared to in Sect. 5, also [11,19] both build on the well-founded MKNF semantics [18]. While [11] uses the non-standard CDF framework integrated in XSB, which complicates compatibility to standard OWL tools based on the OWL API, [19] presents an OWL 2 QL oracle based on common rewritings in the underlying DL $DL\text{-}Lite_R$ [2], but would require constant interaction between a rule reasoner and a DL reasoner, which is why we believe it is ultimately less efficient than our approach. Two related tools are DReW [29] and HD Rules [8], but both are based on different base formalisms to combine ontologies and non-monotonic rules w.r.t. the way information can flow between its two components and how flexible the language is [9,25], which considerably complicates comparison.

For future work, the extension to OWL 2 RL seems an obvious next step, but developing an alternative for OWL 2 QL using the classifier integrated in ontop [21] or even the general reasoner Konclude [28], could shed more light on whether classification or direct translation fares better for proper OWL 2 QL ontologies. The efficiency of the latter reasoner also motivates looking into non-polynomial DLs, with possible influences from recent work on rewriting disjunctive datalog programs [14]. Using a relational database for the data as in OBDA would also be interesting, yet this would require non-trivial theoretical work on rewriting queries including non-monotonic rules. Finally, we may extend NoHR for OWL 2 QL (and EL) to the paraconsistent semantics [15] that would provide true support to the paraconsistent behavior already observed .

Acknowledgments. This work was partially supported by Fundação para a Ciência e a Tecnologia (FCT) under project PTDC/EIA-CCO/121823/2010, strategic project PEst/UID/CEC/04516/2013, and grant SFRH/BPD/86970/2012 (M. Knorr).

References

1. Alferes, J.J., Knorr, M., Swift, T.: Query-driven procedures for hybrid MKNF knowledge bases. ACM Trans. Comput. Log. 14(2), 1–43 (2013)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The $DL\text{-}Lite$ family and relations. J. Artif. Intell. Res. (JAIR) 36, 1–69 (2009)
3. 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)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Procs. of IJCAI. pp. 364–369. Professional Book Center (2005)
5. Calvanese, D., de Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The $DL\text{-}Lite$ family. Journal of Automated Reasoning 39(3), 385–429 (2007)
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semantic Web 2(1), 43–53 (2011)
7. Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. J. ACM 43(1), 20–74 (1996)

8. Drabent, W., Henriksson, J., Maluszynski, J.: Hd-rules: A hybrid system interfacing prolog with dl-reasoners. In: Polleres, A., et al. (eds.) Procs. of ALPSWS. CEUR-WS.org (2007)
9. 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)
10. Gelder, A.V., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. J. ACM 38(3), 620–650 (1991)
11. Gomes, A.S., Alferes, J.J., Swift, T.: Implementing query answering for hybrid MKNF knowledge bases. In: Carro, M., Peña, R. (eds.) Procs. of PADL. LNCS, vol. 5937, pp. 25–39. Springer (2010)
12. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Sem. 3(2-3), 158–182 (2005)
13. Ivanov, V., Knorr, M., Leite, J.: A query tool for $\mathcal{EL}$ with non-monotonic rules. In: Alani, H., et al. (eds.) Procs. of ISWC. LNCS, vol. 8218, pp. 216–231. Springer (2013)
14. Kaminski, M., Nenov, Y., Grau, B.C.: Datalog rewritability of disjunctive datalog programs and its applications to ontology reasoning. In: Brodley, C.E., Stone, P. (eds.) Procs. of AAAI. pp. 1077–1083. AAAI Press (2014)
15. Kaminski, T., Knorr, M., Leite, J.: Efficient paraconsistent reasoning with ontologies and rules. In: Yang, Q., Wooldridge, M. (eds.) Procs. of IJCAI. IJCAI/AAAI (2015)
16. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK: From polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. Journal of Automated Reasoning 53, 1–61 (2013)
17. Kifer, M., Boley, H. (eds.): RIF Overview (Second Edition). W3C Working Group Note 5 February 2013 (2013), available at http://www.w3.org/TR/rif-overview/
18. 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)
19. Knorr, M., Alferes, J.J.: Querying OWL 2 QL and non-monotonic rules. In: Aroyo, L., et al. (eds.) Procs. of ISWC. LNCS, vol. 7031, pp. 338–353. Springer (2011)
20. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: Walsh, T. (ed.) Procs. of IJCAI. pp. 2656–2661. IJCAI/AAAI (2011)
21. Kontchakov, R., Rezk, M., Rodriguez-Muro, M., Xiao, G., Zakharyaschev, M.: Answering SPARQL queries over databases under OWL 2 QL entailment regime. In: Mika, P., et al. (eds.) Procs. of ISWC. LNCS, vol. 8796, pp. 552–567. Springer (2014)
22. Lembo, D., Santarelli, V., Savo, D.F.: Graph-based ontology classification in OWL 2 QL. In: Cimiano, P., et al. (eds.) Procs. of ESWC. LNCS, vol. 7882, pp. 320–334. Springer (2013)
23. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Mylopoulos, J., Reiter, R. (eds.) Procs. of IJCAI. pp. 381–386. Morgan Kaufmann (1991)
24. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles (Second Edition). W3C Recommendation 11 December 2012 (2012), available at http://www.w3.org/TR/owl2-profiles/
25. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5), 93–154 (2010)
26. 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., et al. (eds.) Procs. of ISWC. LNCS, vol. 4825, pp. 816–829. Springer (2007)
27. Savo, D.F., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Romagnoli, V., Ruzzi, M., Stella, G.: Mastro at work: Experiences on ontology-based data access. In: Haarslev, V., et al. (eds.) Procs. of DL. CEUR Workshop Proceedings, vol. 573. CEUR-WS.org (2010)
28. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. J. Web Sem. 27, 78–85 (2014)
29. Xiao, G., Eiter, T., Heymans, S.: The DReW system for nonmonotonic dl-programs. In: Li, J., et al. (eds.) SWWS 2012. Springer Proceedings in Complexity, Springer (2013)