NoHR: Integrating XSB Prolog with the OWL 2 Profiles and Beyond

Carlos Lopes, Matthias Knorr, and João Leite

NOVA LINCS, Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal,

Abstract. We present the latest, substantially improved, version of NoHR, a reasoner designed to answer queries over hybrid theories composed of an OWL ontology in Description Logics and a set of non-monotonic rules in Logic Programming. Whereas the need to combine the distinctive features of these two knowledge representation and reasoning approaches stems from real world applications, their integration is nevertheless theoretically challenging due to their substantial semantical differences. NoHR has been developed as a plug-in for the widely used ontology editor Protégé - in fact, the first hybrid reasoner of its kind for Protégé, building on a combination of reasoners dedicated to OWL and rules - but it is also available as a library, allowing for its integration within other environments and applications. Compared to previous versions of NoHR, this is the first that supports all polynomial OWL profiles, and even beyond, allowing for its usage with real-world ontologies that do not fit within a single profile. In addition, NoHR has now an enhanced integration with its rule engine, which provides support for a vast number of standard built-in Prolog predicates that considerably extend its usability.

Keywords: query answering, logic programming, description logic ontologies

1 Introduction

Ontology languages based on Description Logics (DLs) [4] and non-monotonic rule languages as known from Logic Programming (LP) [6] are both well-known formalisms in knowledge representation and reasoning (KRR) each with its own distinct benefits and features. This is also witnessed by the emergence of the Web Ontology Language (OWL) [13] and the Rule Interchange Format (RIF) [17] in the ongoing standardization of the Semantic Web driven by the W3C.[1]

On the one hand, ontology languages have become widely used to represent and reason over taxonomic knowledge. Since DLs are (usually) decidable fragments of first-order logic, they are monotonic by nature, which means that drawn conclusions persist when adopting new additional information. Furthermore, they allow reasoning on abstract information, such as relations between

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

classes of objects, even without knowing any concrete instances. The balance between expressiveness and complexity of reasoning with ontology languages, inherited from DLs, is witnessed by the fact that the very expressive general language OWL 2, with its high worst-case complexity, includes three tractable (polynomial) profiles [20] each with a different application purpose in mind.

On the other hand, non-monotonic rules explicitly represent inference, from premises to conclusions, focusing on reasoning over instances. They commonly employ the Closed World Assumption (CWA), i.e., the absence of a piece of information suffices to derive it being false, until new information to the contrary is provided, hence being non-monotonic. This permits to declaratively model defaults and exceptions, in the sense that the absence of an exceptional feature can be used to derive that the (more) common case applies, and also integrity constraints, which can be used to ensure that the data under consideration is conform to the desired specifications.

Combining both formalisms, though a non-trivial problem due to the mismatch between semantic assumptions of the two formalisms, and the considerable differences as to how decidability is ensured in each of them, has been frequently requested by applications [22, 1, 23]. For example, in clinical health care, large ontologies such as SNOMED CT,[2] that are captured by the OWL 2 profile OWL 2 EL and its underlying description logic (DL) $\mathcal{EL}^{++}$ [5], are used for electronic health record systems, clinical decision support systems, or remote intensive care monitoring, to name only a few. Yet, expressing conditions such as dextrocardia, i.e., that the heart is exceptionally on the right side of the body, is not possible and requires non-monotonic rules. Another example can be found in [22], where 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. Also, in [1] it is shown that in Legal Reasoning, besides the well known need for default reasoning afforded by non-monotonic rules, it is also necessary to reason in the absence of concrete known individuals (instances), hence requiring features found in ontology languages such as DL. Moreover, another application scenario can be found in the risk assessment of cargo shipment, which we will describe in more detail in Sec. 2.3. Notably, ontologies developed for such applications, are often covered by the constructors the polynomial OWL 2 profiles provide, but do not necessarily fall precisely into one of those profiles.

In this paper, we describe the latest version of NoHR[3] (Nova Hybrid Reasoner), a plug-in for the ontology editor Protégé 5.X,[4] that allows the user to query combinations of ontologies and non-monotonic rules in a top-down manner, which substantially extends the usability of NoHR w.r.t. both the ontology and the rule part.

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

NoHR is theoretically founded on the formalism of Hybrid MKNF under the well-founded semantics [18] which comes with two main arguments in its favor (cf. the related work in [9, 21] on combining DLs with non-monotonic rules). First, the overall approach, introduced in [21] and based on the logic of minimal knowledge and negation as failure (MKNF) [19], provides a very general and flexible framework for combining DL ontologies and non-monotonic rules (see [21]). Second, [18], which is a variant of [21] based on the well-founded semantics [10] for logic programs, has a lower data complexity than the former – it is polynomial for polynomial DLs – 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 and huge amounts of data.

NoHR – the first Protégé plug-in to integrate non-monotonic rules and top-down queries – is implemented in a way that combines the capabilities of the DL reasoners ELK [16], HermiT [11], and Konclude [24] with the rule engine XSB Prolog,[5] exhibiting the following additional features:[6]

- Support for ontologies written in any of the three tractable OWL 2 Profiles, and even beyond for those combining all the permitted constructors;
- Support for a vast number of standard built-in Prolog predicates;
- Rule editor within Protégé accompanied with a completely overhauled rule parser to match the novel extensions;
- Possibility to define predicates with arbitrary arity in Protégé;
- Guaranteed termination of query answering;
- Robustness w.r.t. inconsistencies between the ontology and the rules;
- Scalable fast interactive response times.

2 Hybrid MKNF Knowledge Bases

We start with illustrating the kind of hybrid knowledge bases considered by NoHR. First, we present an overview on the kind of ontologies in description logics permitted here, referring to [4] for a more general and thorough introduction to DLs. Then we recall hybrid MKNF knowledge bases, followed by an example scenario and some general considerations on how to perform query answering.

2.1 Description Logics

Description logics (DLs) are usually decidable fragments of first-order logic, commonly defined over disjoint countably infinite sets of *concept names* $\mathsf{N_C}$, *role names* $\mathsf{N_R}$, and *individual names* $\mathsf{N_I}$, matching unary predicates, binary predicates and constants resp. Building on these, *complex concepts* (and sometimes also *complex roles*) are introduced based on the logical constructors a concrete

[5] http://xsb.sourceforge.net
[6] NoHR 3.0 Beta can be downloaded for testing from http://nohr.di.fct.unl.pt/.

DL admits to be used. An ontology $\mathcal{O}$ then is a finite set of *inclusion axioms* of the form $C \sqsubseteq D$ where C and D are both (complex) concepts (or roles) and *assertions* of the form $C(a)$ or $R(a, b)$ for concepts C, roles R, and individuals a, b. The semantics of such ontologies is defined in terms on *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty domain $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ in a standard way for first-order logic. Typical reasoning tasks are model-checking or consistency or classification which requires computing all concept inclusions between atomic concepts entailed by $\mathcal{O}$.

While the description logic $\mathcal{SROIQ}$ underlying the W3C standard OWL 2 is very general and highly expressive as it admits many different constructors, reasoning with it is highly complex, which is why the profiles OWL 2 EL, OWL 2 QL and OWL 2 RL have been defined [20] for which reasoning is tractable. Since a detailed account on the full technical specification of these profiles would be beyond the scope of this paper, we rather give a brief overview on important features as supported in NoHR.

First, $\mathcal{EL}^+_\bot$, a large fragment of $\mathcal{EL}^{++}$ [5], the DL underlying the tractable profile OWL 2 EL [20], only allows conjunction of concepts, existential restriction of concepts ($\exists R.C$ for roles R and concepts C basically corresponding to $\forall x \exists y R(x, y) \wedge C(y)$), hierarchies of roles, and disjoint concepts. $\mathcal{EL}^+_\bot$ is tailored towards reasoning with large conceptual models, i.e., large TBoxes.

DL-Lite$_R$, one language of the *DL-Lite* family [3] and underlying OWL 2 QL, admits in addition role inverses and disjoint roles, but in exchange only simple role hierarchies, no conjunction, and limitations on the use of existential restrictions in particular on the left hand side of inclusion axioms. This profile focuses on answering queries over huge amounts of data, and is amenable to the usage of relational database technology.

Finally, Description Logic Programs [12], which underly OWL 2 RL, have been introduced with the aim to find a fragment that can be directly translated into rules (and vice-versa), and therefore can also be implemented using a rule reasoner. Due to that, this profile allows more constructors than the other two profiles, but usually restricts their usage to one side of the inclusion axioms to ensure that the translation to rules is possible.

Many ontologies matching one of these profiles exist and are used in applications, but there are also those that do use features from more than one of those, such as the bio-ontology Galen[7] or the famous benchmark ontology LUBM.[8] For these, reasoning is no longer polynomial, but highly efficient general purpose OWL reasoners nevertheless make their usage feasible in practice.

2.2 MKNF Knowledge Bases

MKNF knowledge bases (KBs) build on the logic of minimal knowledge and negation as failure (MKNF) [19]. Two main different semantics have been defined [21, 18], and we focus on the well-founded version [18], due to its lower

[7] https://bioportal.bioontology.org/ontologies/GALEN
[8] http://swat.cse.lehigh.edu/projects/lubm/

computational complexity and amenability to top-down querying without computing the entire model. Here, we only point out important notions following [14], and refer to [18] and [2] for the details.

MKNF knowledge bases as presented in [2] combine an ontology and a set of non-monotonic rules (similar to a normal logic program).

A *rule* r is of the form $H \leftarrow A_1, \ldots, A_n, \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_m$ 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, $\mathcal{O}$ is an ontology, and an *MKNF knowledge base* $\mathcal{K}$ is a pair $(\mathcal{O}, \mathcal{P})$. A rule r is *safe* if all its variables occur in at least one A_i with $1 \leq i \leq n$, and $\mathcal{K}$ is *safe* if all its rules are safe.[9]

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 **K** and **not**. It is shown in [18], that if $\mathcal{K}$ is *MKNF-consistent*, then a unique model of $\mathcal{K}$ can be determined. 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 $\mathcal{P}$. 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 that $\mathcal{O}$ occurring in $\mathcal{K}$ is consistent, which does not truly constitute a restriction as we can always make $\mathcal{O}$ by appropriately turning assertions in $\mathcal{O}$ into rules without any effect on the semantics of $\mathcal{K}$.

2.3 Use Case

The customs service for any developed country assesses imported cargo for a variety of risk factors including terrorism, narcotics, food and consumer safety, pest infestation, tariff violations, and intellectual property rights.[10] Assessing this risk, even at a preliminary level, involves extensive knowledge about commodities, business entities, trade patterns, government policies and trade agreements. Some of this knowledge may be external to a given customs agency: for instance the broad classification of commodities according to the international Harmonized Tariff System (HTS), or international trade agreements. Other knowledge may be internal to a customs agency, such as lists of suspected violators or of importers who have a history of good compliance with regulations.

Figure 1 shows a simplified fragment $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ of such a knowledge base. In this fragment, a shipment has several attributes: the country of its origination, the commodity it contains, its importer and producer. The ontology contains a geographic classification, along with information about producers who are located in various countries. It also contains (partial) information about three shipments: s_1, s_2 and s_3. There is also a set of rules indicating information

[9] In general, the notion of DL-safety is used in this context which requires that these variables occur in atoms that do themselves not occur in the ontology, but due to the particular reasoning method employed here, we can relax that restriction.

[10] The system described here, originally presented in [23], is not intended to reflect the policies of any country or agency.

* * * $\mathcal{O}$ * * *

$\mathsf{Commodity} \equiv (\exists \mathsf{HTSCode}.\top)$
$\mathsf{CherryTomato} \sqsubseteq \mathsf{Tomato}$
$\mathsf{CherryTomato} \sqcap \mathsf{GrapeTomato} \sqsubseteq \bot$
$\mathsf{Tomato} \sqsubseteq \mathsf{EdibleVegetable}$
$\mathsf{GrapeTomato} \sqsubseteq \mathsf{Tomato}$
$\mathsf{Bulk} \sqcap \mathsf{Prepackaged} \sqsubseteq \bot$
$\mathsf{EURegisteredProducer} \equiv (\exists \mathsf{RegisteredProducer}.\mathsf{EUCountry})$
$\mathsf{LowRiskEUCommodity} \equiv (\exists \mathsf{ExpeditableImporter}.\top) \sqcap (\exists \mathsf{CommodCountry}.\mathsf{EUCountry})$
$\mathsf{Inspection} \sqcap \mathsf{NoInspection} \sqsubseteq \bot$

$\mathsf{ShpmtCommod}(s_1, c_1)$
$\mathsf{ShpmtImporter}(s_1, i_1)$
$\mathsf{ShpmtCommod}(s_2, c_2)$
$\mathsf{ShpmtImporter}(s_2, i_2)$
$\mathsf{ShpmtCountry}(s_2, turkey)$
$\mathsf{ShpmtCommod}(s_3, c_3)$
$\mathsf{ShpmtImporter}(s_3, i_3)$
$\mathsf{ShpmtCountry}(s_3, portugal)$
$\mathsf{RegisteredProducer}(p_1, portugal)$
$\mathsf{RegisteredProducer}(p_2, slovakia)$
$\mathsf{ShpmtDeclHTSCode}(s_1, \mathrm{h}7022)$
$\mathsf{CherryTomato}(c_1) \quad \mathsf{Bulk}(c_1)$
$\mathsf{ShpmtDeclHTSCode}(s_2, \mathrm{h}7022)$
$\mathsf{GrapeTomato}(c_2) \quad \mathsf{Prepackaged}(c_2)$
$\mathsf{ShpmtDeclHTSCode}(s_3, \mathrm{h}7021)$
$\mathsf{GrapeTomato}(c_3) \quad \mathsf{Bulk}(c_3)$
$\mathsf{ShpmtProducer}(s_3, p_1)$
$\mathsf{EUCountry}(portugal)$
$\mathsf{EUCountry}(slovakia)$

* * * $\mathcal{P}$ * * *

$\mathsf{AdmissibleImporter}(\mathbf{x}) \leftarrow \mathsf{ShpmtImporter}(\mathbf{y}, \mathbf{x}), \mathbf{not}\ \mathsf{SuspectedBadGuy}(\mathbf{x}).$
$\mathsf{SuspectedBadGuy}(i_1).$
$\mathsf{ApprovedImporterOf}(i_2, \mathbf{x}) \leftarrow \mathsf{EdibleVegetable}(\mathbf{x}).$
$\mathsf{ApprovedImporterOf}(i_3, \mathbf{x}) \leftarrow \mathsf{GrapeTomato}(\mathbf{x}).$
$\mathsf{CommodCountry}(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{z}, \mathbf{x}), \mathsf{ShpmtCountry}(\mathbf{z}, \mathbf{y}).$
$\mathsf{ExpeditableImporter}(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{z}, \mathbf{x}), \mathsf{ShpmtImporter}(\mathbf{z}, \mathbf{y}),$
$\qquad \mathsf{AdmissibleImporter}(\mathbf{y}), \mathsf{ApprovedImporterOf}(\mathbf{y}, \mathbf{x}).$
$\mathsf{CompliantShpmt}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathsf{HTSCode}(\mathbf{y}, \mathbf{z}), \mathsf{ShpmtDeclHTSCode}(\mathbf{x}, \mathbf{z}).$
$\mathsf{Random}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathbf{not}\ \mathsf{Random}(\mathbf{x}).$
$\mathsf{NoInspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathsf{CommodCountry}(\mathbf{y}, \mathbf{z}), \mathsf{EUCountry}(\mathbf{z}).$
$\mathsf{Inspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathbf{not}\ \mathsf{NoInspection}(\mathbf{x}), \mathsf{Random}(\mathbf{x}).$
$\mathsf{Inspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathbf{not}\ \mathsf{CompliantShpmt}(\mathbf{x}).$
$\mathsf{Inspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathsf{Tomato}(\mathbf{y}), \mathsf{ShpmtCountry}(\mathbf{x}, slovakia).$
$\mathsf{HTSChapter}(\mathbf{x}, 7) \leftarrow \mathsf{EdibleVegetable}(\mathbf{x}).$
$\mathsf{HTSHeading}(\mathbf{x}, 702) \leftarrow \mathsf{Tomato}(\mathbf{x}).$
$\mathsf{HTSCode}(\mathbf{x}, \mathrm{h}7022) \leftarrow \mathsf{CherryTomato}(\mathbf{x}).$
$\mathsf{HTSCode}(\mathbf{x}, \mathrm{h}7021) \leftarrow \mathsf{GrapeTomato}(\mathbf{x}).$
$\mathsf{TariffCharge}(\mathbf{x}, 0) \leftarrow \mathsf{CherryTomato}(\mathbf{x}), \mathsf{Bulk}(\mathbf{x}).$
$\mathsf{TariffCharge}(\mathbf{x}, 40) \leftarrow \mathsf{GrapeTomato}(\mathbf{x}), \mathsf{Bulk}(\mathbf{x}).$
$\mathsf{TariffCharge}(\mathbf{x}, 50) \leftarrow \mathsf{CherryTomato}(\mathbf{x}), \mathsf{Prepackaged}(\mathbf{x}).$
$\mathsf{TariffCharge}(\mathbf{x}, 100) \leftarrow \mathsf{GrapeTomato}(\mathbf{x}), \mathsf{Prepackaged}(\mathbf{x}).$

Fig. 1. MKNF knowledge base for Cargo Imports

about importers, and about whether to inspect a shipment either to check for compliance of tariff information or for food safety issues. For that purpose, the set of rules also includes a classification of commodities based on their harmonized tariff information (HTS chapters, headings and codes, cf. http://www.usitc.gov/tata/hts), and tariff information, based on the classification of commodities as given by the ontology.

The overall task then is to access all the information and assess whether some shipment should be inspected in full detail, under certain conditions randomly, or not at all. In fact, an inspection is considered if either a random inspection is indicated, or some shipment is not compliant, i.e., there is a mismatch between the filed cargo codes and the actually carried commodities, or some suspicious cargo is observed, in this case tomatoes from slovakia. In the first case, a potential random inspection is indicated whenever certain exclusion conditions do not hold. To ensure that one can distinguish between strictly required and random inspections, a random inspection is assigned the truth value undefined based on the rule $\mathsf{Random}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathbf{not}\ \mathsf{Random}(\mathbf{x})$. Note that the example indeed utilizes the features of rules and ontologies: for example exceptions to the potential random inspections can be expressed, but at the same time, taxonomic and non-closed knowledge is used, e.g., some shipment may in fact originate from the EU, this information is just not available.

Querying MKNF knowledge bases is based on $\mathbf{SLG}(\mathcal{O})$, as defined in [2]. 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 [2] for the full account of $\mathbf{SLG}(\mathcal{O})$. E.g., the result of querying the example knowledge base for $\mathsf{Inspection}(\mathbf{x})$ reveals that of the three shipments, s_2 requires an inspection (due to mislabeling), s_1 may be subject to a random inspection as it does not knowingly originate from the EU, while s_3 is indicated as inconsistent, due to the fact that Inspection and NoInspection hold, which are required to be disjoint.

3 NoHR Plug-in

In this section, we describe the architecture of the plug-in for Protégé as shown in Fig. 2 and discuss some features of the implementation, in particular the new ones that considerably extend the applicability of the tool.

3.1 Architecture

The input for the plug-in consists of an OWL file written in a description logic as described in Sect. 2.1, which can be loaded and 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 dedicated panel.

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

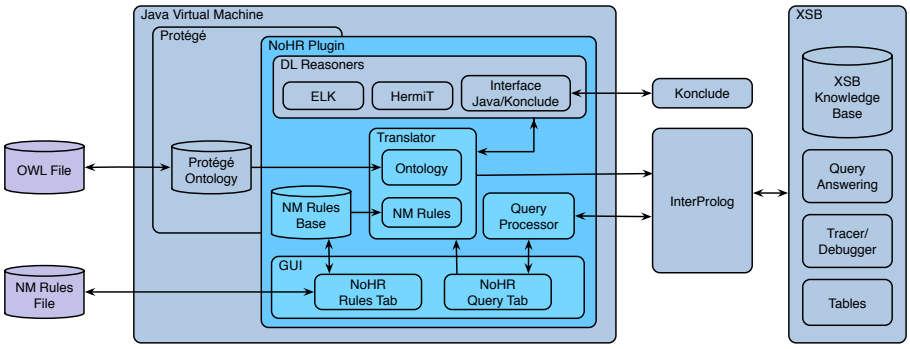

Fig. 2. System architecture of NoHR

the first query is posed by pushing "Execute", the translator is started, which with the help of one of the integrated OWL ontology reasoners ELK [16], HermiT [11], and Konclude [24] transforms the ontology axioms into a set of rules that are equivalent to the ontology axioms in terms of answers to ground queries (see details in Section 3.2). The translation result is joined with the given non-monotonic rules in $\mathcal{P}$, which is further transformed if inconsistency detection is required (in the presence of certain DL constructs in the ontology, such as DisjointWith axioms).

The result is used as input for the top-down query engine XSB Prolog which realizes the well-founded semantics for logic programs [10]. The transfer to XSB is realized via InterProlog,[11] 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). XSB itself not only answers queries very efficiently in a top-down manner, with tabling, it also avoids infinite loops, thus ensuring termination of the query process.

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

3.2 Support for OWL 2 Profiles

The new version of NoHR supports the usage of ontologies written in any of the OWL 2 profiles and even those that combine expressive features from several of them. For the latter, DL reasoning is naturally no longer polynomial, but the usage of highly efficient general purpose DL reasoners, namely HermiT [11], and

[11] http://interprolog.com/java-bridge/

Konclude [24], allow us to compensate for this in practice despite the higher worst-case complexity.

The usage of these DL reasoners depends in fact on the concrete ontology. Namely, a switch determines whether the ontology falls into one of the OWL 2 profiles or not. In the former case, the specific translation module developed for each profile is used, unless the user indicates specifically that one of the general reasoners should be used, which are used anyway in the latter case.

Regarding the OWL 2 profiles, specific translation modules have been developed. For $\mathcal{EL}^+_\bot$, i.e., OWL 2 EL, the ontology reasoner ELK [16], tailored for $\mathcal{EL}^+_\bot$ and considerably faster than other reasoners when comparing classification time, is used to classify the ontology resulting from normalizing the given ontology $\mathcal{O}$ to ensure that no possible derivations w.r.t. answering ground queries are lost during the subsequent translation into rules.[12] The inferred axioms together with $\mathcal{O}$ are translated discarding certain axioms which are irrelevant for answering ground queries. For $DL\text{-}Lite_R$, i.e., OWL 2 QL, a dedicated direct translation without prior classification is used, introducing some auxiliary predicates instead to compensate for the missing inferred axioms (see [14] and [8] for the respective details on both approaches). The new module for DLP, i.e., OWL 2 RL, makes use of a direct translation that does not require any auxiliary predicates as the profile supports the direct translation into rules by design.

Concerning an ontology that does not fall into one single profile, a new general translation module is used that follows the methodology employed for $\mathcal{EL}^+_\bot$, i.e., (partial) normalization, classification, and translation of the (relevant majority of the) inferred axioms, where the translation function itself results from a merge of those for the single profiles. The reason why we apply two different general purpose DL reasoners is as follows. It has been shown that Konclude is in general the most efficient reasoner currently available [24], unfortunately, unlike Protégé, it is not programmed in Java, but in C++, and the necessary effort to interface with Konclude constitutes a considerable drain on efficiency of reasoning, because no native Java interface compatible with the current version of the OWL API exists. HermiT, on the other side, does not suffer from that problem, but has some limitations when reasoning with very large ontologies [11]. Therefore, both reasoners are integrated and the user can choose in the preference panel which of the two should be employed.

Note that if we consider the hybrid knowledge base as presented in Fig. 1 whose ontology is in $\mathcal{EL}^+_\bot$, and query for $\mathsf{Inspection}(x)$, then we would obtain that $s1$ is undefined, s_2 is true, and s_3 is inconsistent. However, if we add an additional property $\mathsf{ImportedBy}$ and declare it to be the inverse of $\mathsf{ShpmtImporter}$, then the resulting ontology would no longer fit any profile and the previous version of NoHR would simply cease to work. In our new version, the ontology change is detected and the general translation module is used instead, that is, NoHR is capable to interactively adjust to changes that alter the DL fragment of the considered ontology.

[12] Similar techniques have been used independently that allow the usage of OWL 2 RL reasoners for answering ground queries for ontologies outside of OWL 2 RL [25].

Finally, note that, while HermiT and Konclude are in fact applicable to arbitrary DL ontologies, the method applied in NoHR is not: it is well known that constructors such as first-order disjunction cannot be captured in non-monotonic rules in an equivalent way, which is why the approach is focused on the constructors occurring in the OWL 2 profiles.

3.3 Extended Prolog Support

The rule editor and parser have also been significantly improved, both in terms of user-friendliness and applicability. To begin with, the rule syntax has been slightly changed so that variables are now always written with a leading "?". While this deviates from standard prolog syntax, it permits the user and the system to differentiate between variables and constant names from the ontology where, in general, there is no restriction to only have names in lowercase. This is also aligned with common rule notation in SWRL,[13] a monotonic first-order rule language that has been used in Protégé.

With the new version, NoHR was also extended with the ability to use a large set of inbuilt XSB prolog predicates. This extension allows the user to specify rules that make use of arithmetic, comparison and list-based predicates for enriching the knowledge base. The novel features include:

- Built-in XSB Prolog predicates prefixed with `#`. For example:

```
A(?X) :- B(?X), C(?Y), #compare("<",?X,?Y).
```

- Inline form of built-in XSB Prolog predicates. For example:

```
A(?X) :- B(?Y), C(?Z), ?X is (?Y + ?Z) * 2.
```

- Numeric and list expressions within rule syntax and integration with numeric datatypes from OWL 2.

In addition, the rule editor allows switching between the labels of ontology names and their Internationalized Resource Identifiers (IRIs), which can be useful in case the labels used in the ontology are not unique.

4 Evaluation

Previous tests on the $\mathcal{EL}^+_\bot$ component have shown that a) different $\mathcal{EL}$ ontologies can be preprocessed for querying in a short period of time (around one minute for SNOMED CT with over 300,000 concepts), b) adding rules increases the time of the translation only linearly, and c) querying time is in general neglectable, in comparison to a) and b) [14]. In subsequent tests on improved versions of both components (for $\mathcal{EL}^+_\bot$ and $DL\text{-}Lite_R$) [8], we have shown that i) NoHR scales reasonably well for OWL QL query answering without non-monotonic

[13] http://www.w3.org/Submission/SWRL/

	Axioms	EL	ELK	HermiT	Konclude
Fly Anatomy	19,211	yes	0.75	1.54	1.63
Full-Galen	37,696	no	–	–	11.21
LUBM1	93	no	–	1.81	3.82
Snomed Anatomy	40,485	yes	1.76	3.08	12.39
Snomed CT	294,479	yes	13.89	–	63.71

Fig. 3. Reasoners average classification run times (in seconds) when called from Java.

rules (only slowing down for memory-intensive cases), ii) preprocessing is even faster when compared to NoHR's previous version using a classifier (for EL), iii) querying scales well, even for over a million facts/assertions in the ABox, despite being slightly slower on average in comparison to EL, and iv) adding rules scales linearly for pre-processing and querying, even for an ontology with many negative inclusions (for $DL\text{-}Lite_R$).

Here, we are additionally interested as to how using the general reasoners compares to applying the dedicated translation module. For that purpose, we first compared how the preprocessing period is affected by the usage of the different reasoners using standard ontologies of different expressiveness (in Fig. 3, we mention the number of inclusion axioms and whether the ontology fits the OWL 2 EL profile). All tests were performed on a 4 GHz Intel i7 processor with 16 GB under Windows 7 (64-bit) with 12 GB of maximum memory for Java. The results are shown in Fig. 3. We can observe that ELK is indeed always fastest, which confirms that by default, whenever the ontology fits $\mathcal{EL}^+_\bot$, the dedicated translation module should be used. Additionally, we observe that HermiT is faster than Konclude in all instances where it does not time out, which justifies using HermiT whenever it is capable of classifying the given ontology. Still, Konclude turns out to be useful, in the cases where HermiT fails to classify an ontology that does not fit the OWL 2 EL profile.

In addition, we also compared HermiT and the translation module for OWL 2 QL that does not use any classifier. This test is actually quite similar to the one in [8] comparing the modules for QL and EL. The difference is that we can use a more expressive ontology that only fits the QL profile, and not EL. Fig. 4 shows the results for ii) where we considered LUBM[14], a standard benchmark for evaluating queries over a large data set, which also includes a given set of standard queries. We created instances of LUBM_n with $n = 1, 5, 10$ using the provided generator, and a restricted version of LUBM which fits OWL QL (thus rendering only one standard query meaningless), with the number of assertions ranging from roughly 100,000 to over 1,400,000. Note that "Initialization" includes loading the ontology and, for HermiT, 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 factor 35 for LUBM_{10}, which is mainly due to the absence of a classification step. When querying, we observed a slight compensation as run-

[14] http://swat.cse.lehigh.edu/projects/lubm/

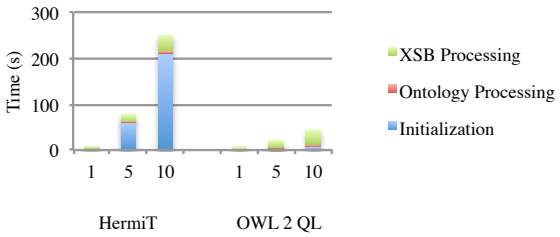

Fig. 4. Preprocessing time for LUBM for the two translation modes

ning queries using the HermiT translation is slightly faster (up to factor 2.5), but faster preprocessing clearly favors the dedicated OWL 2 QL module here.

5 Conclusions

The Protégé plugin NoHR – also distributed as an API – affords us the possibility to query knowledge bases composed of both an ontology in one of the OWL 2 profiles and even a union of their language features and a set of non-monotonic rules, using a top-down reasoning approach, which means that only the part of the ontology and rules that is relevant for the query is actually evaluated. Its sound theoretical foundation together with the fast interactive response times make NoHR a truly one-of-a-kind reasoner and the novel extensions have considerably widened its applicability.

Ongoing work focuses on supporting different sets of rules and allowing the import of other rule file formats such as Prolog. In terms of future work, the integration of database access is of interest as this could considerably reduce the time it takes to load prolog files with huge amounts of data in XSB. Also, adjusting NoHR to the paraconsistent semantics for MKNF knowledge bases of [15] would provide better support to the already observed paraconsistent behavior.

Acknowledgments. We would like to acknowledge the valuable contribution of both Nuno Costa and Vadim Ivanov to the development of NoHR. This work was partially supported by Fundação para a Ciência e a Tecnologia (FCT) under UID/CEC/04516/2013, and grant SFRH/BPD/86970/2012 (M. Knorr).

References

1. Alberti, M., Knorr, M., Gomes, A.S., Leite, J., Gonçalves, R., Slota, M.: Normative systems require hybrid knowledge bases. In: AAMAS. pp. 1425–1426 (2012)
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-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 $\mathcal{EL}$ envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Procs. of IJCAI. pp. 364–369. Professional Book Center (2005)
6. Baral, C., Gelfond, M.: Logic programming and knowledge representation. J. Log. Program. 19/20, 73–148 (1994)
7. Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. J. ACM 43(1), 20–74 (1996)
8. Costa, N., Knorr, M., Leite, J.: Next step for NoHR: OWL 2 QL. In: Arenas, M., Corcho, O., et al. (eds.) Procs. of ISWC. LNCS, vol. 9366, pp. 569–586 (2015)
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. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: An OWL 2 reasoner. J. Autom. Reasoning 53(3), 245–269 (2014)
12. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Hencsey, G., White, B., Chen, Y.R., Kovács, L., Lawrence, S. (eds.) Procs. of WWW. pp. 48–57. ACM (2003)
13. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer (Second Edition). W3C Recommendation 11 December 2012 (2012), available from http://www.w3.org/TR/owl2-primer/
14. 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 (2013)
15. Kaminski, T., Knorr, M., Leite, J.: Efficient paraconsistent reasoning with ontologies and rules. In: Yang, Q., Wooldridge, M. (eds.) Procs. of IJCAI (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. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Mylopoulos, J., Reiter, R. (eds.) Procs. of IJCAI. pp. 381–386. Morgan Kaufmann (1991)
20. 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/
21. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5), 93–154 (2010)
22. Patel, C., et al.: Matching patient records to clinical trials using ontologies. In: Aberer, K., et al. (eds.) Procs. of ISWC. LNCS, vol. 4825, pp. 816–829 (2007)
23. Slota, M., Leite, J., Swift, T.: On updates of hybrid knowledge bases composed of ontologies and rules. Artif. Intell. 229, 33–104 (2015)
24. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. J. Web Sem. 27, 78–85 (2014)
25. Stoilos, G., Grau, B.C., Motik, B., Horrocks, I.: Repairing ontologies for incomplete reasoners. In: Aroyo, L., et al. (eds.) Procs. of ISWC. LNCS, vol. 7031, pp. 681–696. Springer (2011)