Reasoning over Ontologies and Non-monotonic Rules

Vadim Ivanov, Matthias Knorr, and João Leite

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

Abstract. Ontology languages and non-monotonic rule languages are both well-known formalisms in knowledge representation and reasoning, each with its own distinct benefits and features which are quite orthogonal to each other. Both appear in the Semantic Web stack in distinct standards – OWL and RIF – and over the last decade a considerable research effort has been put into trying to provide a framework that combines the two. Yet, the considerable number of theoretical approaches resulted, so far, in very few practical reasoners, while realistic use-cases are scarce. In fact, there is little evidence that developing applications with combinations of ontologies and rules is actually viable. In this paper, we present a tool called NoHR that allows one to reason over ontologies and non-monotonic rules, illustrate its use in a realistic application, and provide tests of scalability of the tool, thereby showing that this research effort can be turned into practice.

1 Introduction

Ontology languages in the form of 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) [18] and the Rule Interchange Format (RIF) [7] 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 and, since DLs are (usually) decidable fragments of first-order logic, are monotonic by nature which means that once drawn conclusions persist when adopting new additional information. They also allow reasoning on abstract information, such as relations between classes of objects even without knowing any concrete instances and a main theme inherited from DLs is the balance between expressiveness and complexity of reasoning. In fact, the very expressive general language OWL 2 with its high worst-case complexity includes three tractable (polynomial) profiles [27] each with a different application purpose in mind.

On the other hand, non-monotonic rules are focused on reasoning over instances and commonly apply 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 the term non-monotonic. This permits to declaratively model defaults and exceptions, in the sense that the absence of an exceptional feature can be used to

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

derive that the (more) common case applies, and also integrity constraints, which can be used to ensure that the considered data is conform with desired specifications.

Combining both formalisms has been frequently requested by applications [1]. 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.

Finding such a combination is a non-trivial problem due to the considerable differences as to how decidability is ensured in each of the two formalisms and a naive combination is easily undecidable. In recent years, there has been a considerable amount of effort devoted to combining DLs with non-monotonic rules as known from Logic Programming – see, e.g., related work in [12,28]) – but this has not been accompanied by similar variety of reasoners and applications. In fact, only very few reasoners for combining ontologies and non-monotonic rules exist and realistic use-cases are scarce. In other words, there is little evidence so far that developing applications in combinations of ontologies and rules is actually viable.

In this paper, we want to contribute to showing that this paradigm is viable by describing a tool called NoHR and show how it can be used to handle a real use-case efficiently as well as its scalability. NoHR is theoretically founded in the formalism of Hybrid MKNF under the well-founded semantics [22] which comes with two main arguments in its favor. First, the overall approach, which was introduced in [28] and is based on the logic of minimal knowledge and negation as failure (MKNF) [26], provides a very general and flexible framework for combining DL ontologies and non-monotonic rules (see [28]). Second, [22], which is a variant of [28] based on the well-founded semantics [13] 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 is realized as a plug-in for the ontology editor Protégé 4.X,[3] that allows the user to query combinations of $\mathcal{EL}_{\perp}^{+}$ ontologies and non-monotonic rules in a top-down manner. To the best of our knowledge, it is the first Protégé plug-in to integrate non-monotonic rules and top-down queries. We describe its features including the possibility to load and edit rule bases, and define predicates with arbitrary arity; guaranteed termination of query answering, with a choice between one/many answers; robustness w.r.t. inconsistencies between the ontology and the rule part and demonstrate its effective usage on the application use-case combining $\mathcal{EL}_{\perp}^{+}$ ontologies and non-monotonic rules outlined in the following and adapted from [29], as well as an evaluation for real ontology SNOMED CT with over 300,000 concepts.

Example 1. The customs service for any developed country assesses imported cargo for a variety of risk factors including terrorism, narcotics, food and consumer safety, pest

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

[3] http://protege.stanford.edu

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

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

infestation, tariff violations, and intellectual property rights. Assessing this risk, even at a preliminary level, involves extensive knowledge about commodities, business entities, trade patterns, government policies and trade agreements. Parts of this knowledge is ontological information and taxonomic, such as the classification of commodities, while other parts require the CWA and thus non-monotonic rules, such as the policies involving, e.g., already known suspects. 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.

The remainder of the paper is structured as follows. In Sect. 2, we briefly recall the DL $\mathcal{EL}_{\bot}^{+}$ and MKNF knowledge bases as a tight combination of the former DL and non-monotonic rules. Then, in Sect. 3, we present the Protégé plug-in NoHR, and, in Sect. 4, we discuss the cargo shipment use case and its realization using NoHR. We present some evaluation data in Sect. 5, before we conclude in Sect. 6.[4]

2 Preliminaries

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

We start by recalling the syntax and semantics of $\mathcal{EL}_{\bot}^{+}$, a large fragment of $\mathcal{EL}^{++}$ [5], the DL underlying the tractable profile OWL 2 EL [27], following the presentation in [21]. For a more general and thorough introduction to DLs we refer to [4].

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

[4] Details on the translation of $\mathcal{EL}$ ontologies into rules used in NoHR can be found in [19].

$C \equiv D$ is an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. Concept and role assertions are *ABox axioms* and all other axioms *TBox axioms*, and an *ontology* is a finite set of axioms.

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

2.2 MKNF Knowledge Bases

MKNF knowledge bases (KBs) build on the logic of minimal knowledge and negation as failure (MKNF) [26]. Two main different semantics have been defined [28,22], and we focus on the well-founded version [22], due to its lower computational complexity and amenability to top-down querying without computing the entire model. Here, we only point out important notions, and refer to [22] and [2] for the details.

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

Definition 2. *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. The* ground instantiation *of $\mathcal{K}$ is the KB $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ where $\mathcal{P}_G$ is obtained from $\mathcal{P}$ by replacing each rule r of $\mathcal{P}$ with a set of rules substituting each variable in r with constants from $\mathcal{K}$ in all possible ways.*

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.

The semantics of $\mathcal{K}$ is based on a transformation of $\mathcal{K}$ into an MKNF formula to which the MKNF semantics can be applied (see [22,26,28] for details). Instead of spelling out the technical details of the original MKNF semantics [28] or its three-valued counterpart [22], we focus on a compact representation of models for which the computation of the well-founded MKNF model is defined.[5] This representation is based on a set of $\mathbf{K}$-atoms and $\pi(\mathcal{O})$, the translation of $\mathcal{O}$ into first-order logic.

[5] Strictly speaking, this computation yields the so-called well-founded partition from which the well-founded MKNF model is defined (see [22] for details).

Definition 3. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base. The* set of $\mathbf{K}$-atoms of $\mathcal{K}_G$*, written* $\mathsf{KA}(\mathcal{K}_G)$*, is the smallest set that contains (i) all ground atoms occurring in $\mathcal{P}_G$, and (ii) an atom ξ for each ground $\mathbf{not}$-atom $\mathbf{not}\xi$ occurring in $\mathcal{P}_G$. For a subset S of $\mathsf{KA}(\mathcal{K}_G)$, the* objective knowledge *of S w.r.t. $\mathcal{K}_G$ is the set of first-order formulas $\mathsf{OB}_{\mathcal{O},S} = \{\pi(\mathcal{O})\} \cup S$.*

The set $\mathsf{KA}(\mathcal{K}_G)$ contains all atoms occurring in $\mathcal{K}_G$, only with $\mathbf{not}$-atoms substituted by corresponding atoms, while $\mathsf{OB}_{\mathcal{O},S}$ provides a first-order representation of $\mathcal{O}$ together with a set of known/derived facts. In the three-valued MKNF semantics, this set of $\mathbf{K}$-atoms can be divided into true, undefined and false atoms. Next, we recall operators from [22] that derive consequences based on $\mathcal{K}_G$ and a set of $\mathbf{K}$-atoms that is considered to hold.

Definition 4. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a positive, ground hybrid MKNF knowledge base. The operators $R_{\mathcal{K}_G}$, $D_{\mathcal{K}_G}$, and $T_{\mathcal{K}_G}$ are defined on subsets of $\mathsf{KA}(\mathcal{K}_G)$:*

$$\begin{aligned}
R_{\mathcal{K}_G}(S) =& \{H \mid \mathcal{P}_G \textit{ contains a rule of the form } H \leftarrow A_1, \ldots A_n \\
& \textit{such that, for all i, } 1 \leq i \leq n, A_i \in S\} \\
D_{\mathcal{K}_G}(S) =& \{\xi \mid \xi \in \mathsf{KA}(\mathcal{K}_G) \textit{ and } \mathsf{OB}_{\mathcal{O},S} \models \xi\} \\
T_{\mathcal{K}_G}(S) =& R_{\mathcal{K}_G}(S) \cup D_{\mathcal{K}_G}(S)
\end{aligned}$$

The operator $T_{\mathcal{K}_G}$ is monotonic, and thus has a least fixpoint $T_{\mathcal{K}_G} \uparrow \omega$. Transformations can be defined that turn an arbitrary hybrid MKNF KB $\mathcal{K}_G$ into a positive one (respecting the given set S) to which $T_{\mathcal{K}_G}$ can be applied. To ensure coherence, i.e., that classical negation in the DL enforces default negation in the rules, two slightly different transformations are defined (see [22] for details).

Definition 5. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and $S \subseteq \mathsf{KA}(\mathcal{K}_G)$. The* MKNF transform $\mathcal{K}_G/S$ *is defined as $\mathcal{K}_G/S = (\mathcal{O}, \mathcal{P}_G/S)$, where $\mathcal{P}_G/S$ contains all rules $H \leftarrow A_1, \ldots, A_n$ for which there exists a rule of the form (1) in $\mathcal{P}_G$ with $B_j \notin S$ for all $1 \leq j \leq m$. The* MKNF-coherent transform $\mathcal{K}_G//S$ *is defined as $\mathcal{K}_G//S = (\mathcal{O}, \mathcal{P}_G//S)$, where $\mathcal{P}_G//S$ contains all rules $H \leftarrow A_1, \ldots, A_n$ for which there exists a rule of the form (1) with $B_j \notin S$ for all $1 \leq j \leq m$ and $\mathsf{OB}_{\mathcal{O},S} \not\models \neg H$. We define $\Gamma_{\mathcal{K}_G}(S) = T_{\mathcal{K}_G/S} \uparrow \omega$ and $\Gamma'_{\mathcal{K}_G}(S) = T_{\mathcal{K}_G//S} \uparrow \omega$.*

Based on these two antitonic operators [22], two sequences $\mathbf{P}_i$ and $\mathbf{N}_i$ are defined, which correspond to the true and non-false derivations.

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

The fixpoints yield the well-founded MKNF model [22] (in polynomial time).

Definition 6. *The* well-founded MKNF model *of an MKNF-consistent ground hybrid MKNF knowledge base $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ is defined as $(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$.*

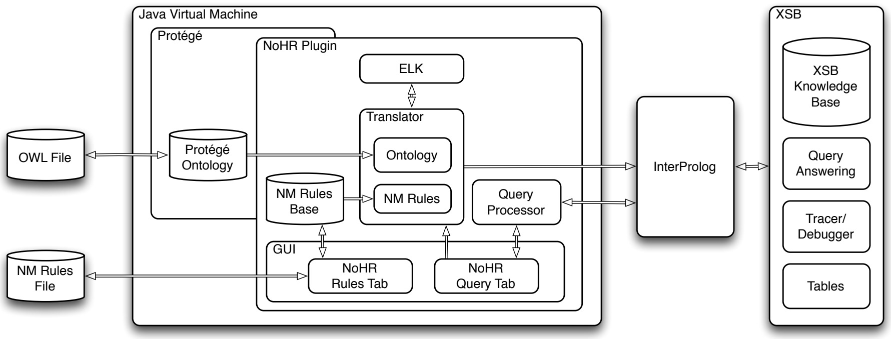


Fig. 1. System Architecture of NoHR

If $\mathcal{K}$ is *MKNF-consistent*, then this partition does correspond to the unique model of $\mathcal{K}$ [22], and, like in [2], we call the partition the *well-founded MKNF model* $\mathsf{M}_{\mathsf{wf}}(\mathcal{K})$. Here, $\mathcal{K}$ may indeed not be MKNF-consistent if the $\mathcal{EL}_{\bot}^{+}$ ontology alone is inconsistent, which is possible if $\bot$ occurs, or by the combination of appropriate axioms in $\mathcal{O}$ and $\mathcal{P}$, e.g., $A \sqsubseteq \bot$ and $A(a) \leftarrow$. In the former case, we argue that the ontology alone should be consistent and be repaired if necessary before combining it with non-monotonic rules. Thus, we assume in the following that $\mathcal{O}$ occurring in $\mathcal{K}$ is consistent.

3 System Description

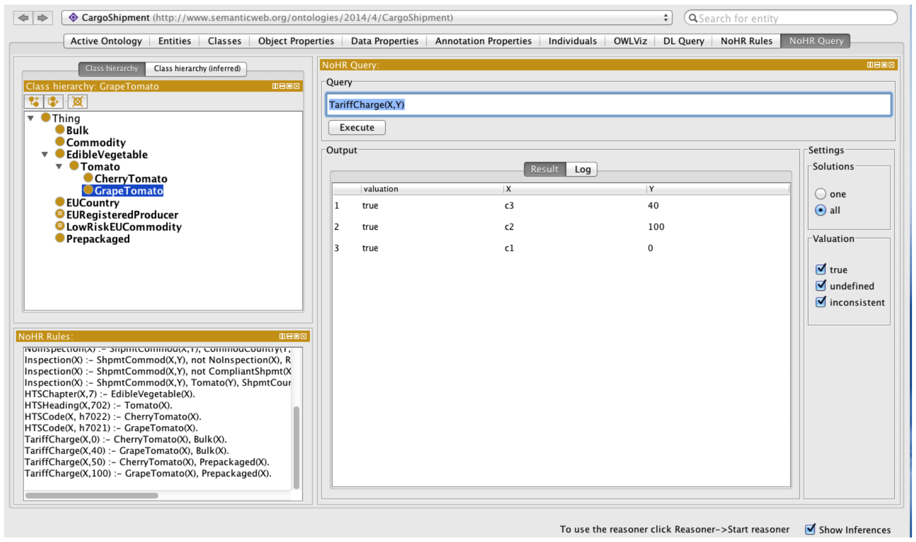
In this section, we briefly describe the architecture of the plug-in for Protégé as shown in Fig. 1 and discuss some features of the implementation and how querying is realized.

The input for the plug-in consists of an OWL file in the DL $\mathcal{EL}_{\bot}^{+}$ as described in Sect. 2.1, which can be manipulated as usual in Protégé, and a rule file. For the latter, we provide a tab called NoHR Rules that allows us to load, save and edit rule files in a text panel following standard Prolog conventions.

The NoHR Query tab (see Fig. 2) also allows for the visualization of the rules, but its main purpose is to provide an interface for querying the combined KB. Whenever the first query is posed by pushing "Execute", a translator is started, initiating the ontology reasoner ELK [21] tailored for $\mathcal{EL}_{\bot}^{+}$ and considerably faster than other reasoners when comparing classification time [21]. ELK is used to classify the ontology $\mathcal{O}$ and then return the inferred axioms to the translator. It is also verified whether $DisjointWith$ axioms appear in $\mathcal{O}$, i.e., in $\mathcal{EL}_{\bot}^{+}$ notation, axioms of the form $C \sqcap D \sqsubseteq \bot$ for arbitrary classes C and D, which determines whether inconsistencies may occur in the combined hybrid knowledge base. Then the result of the classification is translated into rules and joined with the already given non-monotonic rules in $\mathcal{P}$, and the result is conditionally further transformed if inconsistency detection is required.

The result is used as input for the top-down query engine XSB Prolog [6] which realizes the well-founded semantics for logic programs [13]. To guarantee full compatibility with XSB Prolog's more restrictive admitted input syntax, the joint resulting rule set is

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



```
NoInspection(X) :- ShpmtCommod(X,Y), CommodCountry(Y,
Inspection(X) :- ShpmtCommod(X,Y), not NoInspection(X), R
Inspection(X) :- ShpmtCommod(X,Y), not CompliantShpmt(X
Inspection(X) :- ShpmtCommod(X,Y), Tomato(Y), ShpmtCour
HTSChapter(X,7) :- EdibleVegetable(X).
HTSHeading(X,702) :- Tomato(X).
HTSCode(X, h7022) :- CherryTomato(X).
HTSCode(X, h7021) :- GrapeTomato(X).
TariffCharge(X,0) :- CherryTomato(X), Bulk(X).
TariffCharge(X,40) :- GrapeTomato(X), Bulk(X).
TariffCharge(X,50) :- CherryTomato(X), Prepackaged(X).
TariffCharge(X,100) :- GrapeTomato(X), Prepackaged(X).
```

Query: TariffCharge(X,Y)

	valuation	X	Y
1	true	c3	40
2	true	c2	100
3	true	c1	0

Solutions: ( ) one [✓ all]

Valuation: [✓ true] [✓ undefined] [✓ inconsistent]

Fig. 2. NoHR Query tab with a query for $\mathsf{TariffCharge}(\mathbf{x}, \mathbf{y})$ (see Sect. 4)

further transformed such that all predicates and constants are encoded using MD5. The result is transfered to XSB via InterProlog [9],[7] which is an open-source Java front-end allowing the communication between Java and a Prolog engine.

Next, the query is sent via InterProlog to XSB, and answers are returned to the query processor, which collects them and sets up a table showing for which variable substitutions we obtain true, undefined, or inconsistent valuations (or just shows the truth value for a ground query). The table itself is shown in the Result tab (see Fig. 2) of the Output panel, while the Log tab shows measured times of pre-processing the knowledge base and answering the query. XSB itself not only answers queries very efficiently in a top-down manner, with tabling, it also avoids infinite loops.

Once the query has been answered, the user may pose other queries, and the system will simply send them directly without any repeated preprocessing. If the user changes data in the ontology or in the rules, then the system offers the option to recompile, but always restricted to the part that actually changed.

4 Cargo Shipment 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.[8] 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

[7] http://www.declarativa.com/interprolog/

[8] The system described here is not intended to reflect the policies of any country or agency.

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

$\mathsf{Commodity} \equiv (\exists \mathsf{HTSCode}.\top)$	$\mathsf{Tomato} \sqsubseteq \mathsf{EdibleVegetable}$
$\mathsf{CherryTomato} \sqsubseteq \mathsf{Tomato}$	$\mathsf{GrapeTomato} \sqsubseteq \mathsf{Tomato}$
$\mathsf{CherryTomato} \sqcap \mathsf{GrapeTomato} \sqsubseteq \bot$	$\mathsf{Bulk} \sqcap \mathsf{Prepackaged} \sqsubseteq \bot$

$\mathsf{EURegisteredProducer} \equiv (\exists \mathsf{RegisteredProducer}.\mathsf{EUCountry})$

$\mathsf{LowRiskEUCommodity} \equiv (\exists \mathsf{ExpeditableImporter}.\top) \sqcap (\exists \mathsf{CommodCountry}.\mathsf{EUCountry})$

$\mathsf{ShpmtCommod}(s_1, c_1)$	$\mathsf{ShpmtDeclHTSCode}(s_1, \mathrm{h7022})$
$\mathsf{ShpmtImporter}(s_1, i_1)$	$\mathsf{CherryTomato}(c_1)$ $\mathsf{Bulk}(c_1)$
$\mathsf{ShpmtCommod}(s_2, c_2)$	$\mathsf{ShpmtDeclHTSCode}(s_2, \mathrm{h7022})$
$\mathsf{ShpmtImporter}(s_2, i_2)$	$\mathsf{GrapeTomato}(c_2)$ $\mathsf{Prepackaged}(c_2)$
$\mathsf{ShpmtCountry}(s_2, portugal)$	
$\mathsf{ShpmtCommod}(s_3, c_3)$	$\mathsf{ShpmtDeclHTSCode}(s_3, \mathrm{h7021})$
$\mathsf{ShpmtImporter}(s_3, i_3)$	$\mathsf{GrapeTomato}(c_3)$ $\mathsf{Bulk}(c_3)$
$\mathsf{ShpmtCountry}(s_3, portugal)$	$\mathsf{ShpmtProducer}(s_3, p_1)$
$\mathsf{RegisteredProducer}(p_1, portugal)$	$\mathsf{EUCountry}(portugal)$
$\mathsf{RegisteredProducer}(p_2, slovakia)$	$\mathsf{EUCountry}(slovakia)$

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

$\mathsf{AdmissibleImporter}(\mathbf{x}) \leftarrow \mathsf{ShpmtImporter}(\mathbf{y}, \mathbf{x}), \mathbf{not}\mathsf{SuspectedBadGuy}(\mathbf{x}).$
$\mathsf{SuspectedBadGuy}(i_1).$
$\mathsf{ApprovedImporterOf}(i_2, \mathbf{x}) \leftarrow \mathsf{EdibleVegetable}(\mathbf{x}).$
$\mathsf{ApprovedImporterOf}(i_3, \mathbf{x}) \leftarrow \mathsf{GrapeTomato}(\mathbf{x}).$
$\mathsf{CommodCountry}(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{z}, \mathbf{x}), \mathsf{ShpmtCountry}(\mathbf{z}, \mathbf{y}).$
$\mathsf{ExpeditableImporter}(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{z}, \mathbf{x}), \mathsf{ShpmtImporter}(\mathbf{z}, \mathbf{y}),$
$\qquad \mathsf{AdmissibleImporter}(\mathbf{y}), \mathsf{ApprovedImporterOf}(\mathbf{y}, \mathbf{x}).$
$\mathsf{CompliantShpmt}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathsf{HTSCode}(\mathbf{y}, \mathbf{z}), \mathsf{ShpmtDeclHTSCode}(\mathbf{x}, \mathbf{z}).$
$\mathsf{Random}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathbf{not}\mathsf{Random}(\mathbf{x}).$
$\mathsf{NoInspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathsf{CommodCountry}(\mathbf{y}, \mathbf{z}), \mathsf{EUCountry}(\mathbf{z}).$
$\mathsf{Inspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathbf{not}\mathsf{NoInspection}(\mathbf{x}), \mathsf{Random}(\mathbf{x}).$
$\mathsf{Inspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathbf{not}\mathsf{CompliantShpmt}(\mathbf{x}).$
$\mathsf{Inspection}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathsf{Tomato}(\mathbf{y}), \mathsf{ShpmtCountry}(\mathbf{x}, slovakia).$
$\mathsf{HTSChapter}(\mathbf{x}, 7) \leftarrow \mathsf{EdibleVegetable}(\mathbf{x}).$
$\mathsf{HTSHeading}(\mathbf{x}, 702) \leftarrow \mathsf{Tomato}(\mathbf{x}).$
$\mathsf{HTSCode}(\mathbf{x}, \mathrm{h7022}) \leftarrow \mathsf{CherryTomato}(\mathbf{x}).$
$\mathsf{HTSCode}(\mathbf{x}, \mathrm{h7021}) \leftarrow \mathsf{GrapeTomato}(\mathbf{x}).$
$\mathsf{TariffCharge}(\mathbf{x}, 0) \leftarrow \mathsf{CherryTomato}(\mathbf{x}), \mathsf{Bulk}(\mathbf{x}).$
$\mathsf{TariffCharge}(\mathbf{x}, 40) \leftarrow \mathsf{GrapeTomato}(\mathbf{x}), \mathsf{Bulk}(\mathbf{x}).$
$\mathsf{TariffCharge}(\mathbf{x}, 50) \leftarrow \mathsf{CherryTomato}(\mathbf{x}), \mathsf{Prepackaged}(\mathbf{x}).$
$\mathsf{TariffCharge}(\mathbf{x}, 100) \leftarrow \mathsf{GrapeTomato}(\mathbf{x}), \mathsf{Prepackaged}(\mathbf{x}).$

Fig. 3. MKNF knowledge base for Cargo Imports

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 3 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 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 $\mathsf{Random}(\mathbf{x}) \leftarrow \mathsf{ShpmtCommod}(\mathbf{x}, \mathbf{y}), \mathbf{not}\mathsf{Random}(\mathbf{x})$.

The result of querying this knowledge base for $\mathsf{Inspection}(\mathbf{x})$ reveals that of the three shipments, s_2 requires an inspection (due to mislabeling) while s_1 may be subject to a random inspection as it does not knowingly originate from the EU. It can also be verified using the tool that preprocessing the knowledge base can be handled within $300ms$ and the query only takes $12ms$, which certainly suffices as interactive response. Please also 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.

5 Evaluation

In this section, we present some tests showing that a) the huge $\mathcal{EL}^+$ ontology SNOMED CT can be preprocessed for querying in a short period of time, b) adding rules increases the time of the translation only linearly, and c) querying time is in comparison to a) and b) in general completely neglectable. We performed the tests on a Mac book air 13 under Mac OS X 10.8.4 with a 1.8 GHz Intel Core i5 processor and 8 GB 1600 MHz DDR3 of memory. We ran all tests in a terminal version and Java with the "-XX:+AggressiveHeap" option, and test results are averages over 5 runs.

We considered SNOMED CT, freely available for research and evaluation,[9] and added a varying number of non-monotonic rules. These rules were generated arbitrarily,

[9] `http://www.ihtsdo.org/licensing/`

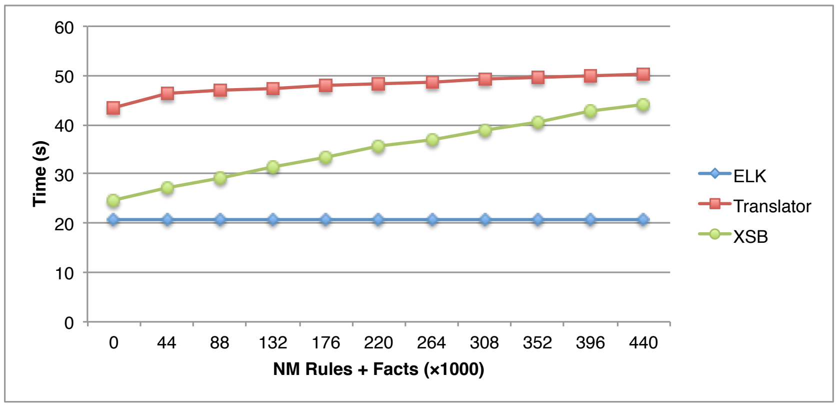


Fig. 4. Preprocessing time for SNOMED with a varying number of Rules

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

The results are shown in Fig. 4 (containing also a constant line for classification of ELK alone and starting with the values for the case without additional rules), and clearly show that a) preprocessing an ontology with over 300,000 concepts takes less than 70 sec. (time for translator+loading in XSB), b) the time of translator and loading the file in XSB only grows linearly on the number of rules with a small degree, in particular in the case of translator, and c) even with up to 500,000 added rules the time for translating does not surpass ELK classification, which itself is really fast [21], by more than a factor 2.5. All this data indicates that even on with a very large ontology, preprocessing can be handled very efficiently.

Finally, we also tested the querying time. To this purpose, we randomly generated and handcrafted several queries of different sizes and shapes using SNOMED with a varying number of non-monotonic rules as described before. In all cases, we observed that the query response time is interactive, observing longer reply times only if the number of replies is very high because either the queried class contains many subclasses in the hierarchy or if the arbitrarily generated rules create too many meaningless links, thus in the worst case requiring to compute the entire model. Requesting only one solution avoids this problem. Still, the question of realistic randomly generated rule bodies for testing querying time remain an issue of future work.

6 Conclusions

We have presented NoHR, the first plug-in for the ontology editor Protégé that integrates non-monotonic rules and top-down queries with ontologies in the OWL 2 profile

OWL 2 EL. We have discussed how this procedure is implemented as a tool and shown how it can be used to implement a real use case on cargo shipment inspections. We have also presented an evaluation which shows that the tool is applicable to really huge ontologies, here SNOMED CT.

There are several relevant approaches discussed in the literature. Most closely related are probably [15,23], because both build on the well-founded MKNF semantics [22]. In fact, [15] is maybe closest in spirit to the original idea of **SLG**($\mathcal{O}$) oracles represented in [2] on which the implementation of NoHR is theoretically founded. It utilizes the CDF framework already integrated in XSB, but its non-standard language is a drawback if we want to achieve compatibility with standard OWL tools based on the OWL API. On the other hand, [23], presents an OWL 2 QL oracle based on common rewritings in the underlying DL DL-Lite [3]. Less closely related is the work pursued in [8,14] that investigates direct non-monotonic extensions of $\mathcal{EL}$, so that the main reasoning task focuses on finding default subset inclusions, unlike this query-centered approach.

Two other related tools are DReW [30] and HD Rules [10], but both are based on different underlying formalisms to combine ontologies and non-monotonic rules. The former builds on dl-programs [12] and focuses on datalog-rewritable DLs [17], and the latter builds on Hybrid Rules [11]. While a more detailed comparison is surely of interest, the main problem is that both underlying formalisms differ from MKNF knowledge bases in the way information can flow between its two components and how flexible the language is [12,28].

We conclude with pointing out that given the successful application of the tool to the use-case as well as its evaluation, an obvious next step will be to try applying it to other use-case domains. This will allow gathering data, which may then be used for a) further dissemination in particular of query processing, which would b) stimulate application-driven optimizations and enhancements of the tool NoHR. Other future directions are extensions to paraconsistency [20] or more general formalisms [16,24,25].

Acknowledgments We would like to thank the referees for their comments. We acknowledge partial support by FCT under project ERRO (PTDC/EIA-CCO/121823/2010) and under strategic project NOVA LINCS (PEst/UID/CEC/04516/2013). V. Ivanov was partially supported by a MULTIC – Erasmus Mundus Action 2 grant and M. Knorr was also partially supported by FCT grant SFRH/BPD/86970/2012.

References

1. Alberti, M., Knorr, M., Gomes, A.S., Leite, J., Gonçalves, R., Slota, M.: Normative systems require hybrid knowledge bases. In: Procs. of AAMAS. pp. 1425–1426. IFAAMAS (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: Procs. of IJCAI (2005)

6. Baral, C., Gelfond, M.: Logic programming and knowledge representation. J. Log. Program. 19/20, 73–148 (1994)
7. Boley, H., Kifer, M. (eds.): RIF Overview. W3C Recommendation 5 February 2013 (2013), available at http://www.w3.org/TR/rif-overview/
8. Bonatti, P.A., Faella, M., Sauro, L.: $\mathcal{EL}$ with default attributes and overriding. In: Procs. of ISWC. LNCS, vol. 6496, pp. 64–79. Springer (2010)
9. Calejo, M.: Interprolog: Towards a declarative embedding of logic programming in java. In: Procs. of JELIA. LNCS, vol. 3229, pp. 714–717. Springer (2004)
10. Drabent, W., Henriksson, J., Maluszynski, J.: Hd-rules: A hybrid system interfacing prolog with dl-reasoners. In: Procs. of ALPSWS. vol. 287 (2007)
11. Drabent, W., Maluszynski, J.: Hybrid rules with well-founded semantics. Knowl. Inf. Syst. 25(1), 137–168 (2010)
12. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artif. Intell. 172(12-13), 1495–1539 (2008)
13. Gelder, A.V., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. J. ACM 38(3), 620–650 (1991)
14. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Reasoning about typicality in low complexity dls: The logics $\mathcal{EL}^{\perp}\mathrm{T}_{min}$ and $DL\text{-}Lite_c\mathrm{T}_{min}$. In: Procs. of IJCAI (2011)
15. Gomes, A.S., Alferes, J.J., Swift, T.: Implementing query answering for hybrid MKNF knowledge bases. In: Procs. of PADL. pp. 25–39. Springer (2010)
16. Gonçalves, R., Alferes, J.: Parametrized logic programming. In: Procs. of JELIA. LNCS, vol. 6341, pp. 182–194. Springer (2010)
17. Heymans, S., Eiter, T., Xiao, G.: Tractable reasoning with dl-programs over datalog-rewritable description logics. In: Procs of ECAI. pp. 35–40. IOS Press (2010)
18. 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/
19. Ivanov, V., Knorr, M., Leite, J.: A query tool for *EL* with non-monotonic rules. In: Procs. of ISWC. LNCS, vol. 8218, pp. 216–231. Springer (2013)
20. Kaminski, T., Knorr, M., Leite, J.: Efficient paraconsistent reasoning with ontologies and rules. In: Procs. of IJCAI. IJCAI/AAAI (2015)
21. 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)
22. 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)
23. Knorr, M., Alferes, J.J.: Querying OWL 2 QL and non-monotonic rules. In: Procs. of ISWC. LNCS, vol. 7031, pp. 338–353. Springer (2011)
24. Knorr, M., Hitzler, P., Maier, F.: Reconciling OWL and non-monotonic rules for the semantic web. In: Procs. of ECAI. pp. 474–479. IOS Press (2012)
25. Knorr, M., Slota, M., Leite, J., Homola, M.: What if no hybrid reasoner is available? hybrid MKNF in multi-context systems. J. Log. Comput. 24(6), 1279–1311 (2014)
26. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Procs. of IJCAI (1991)
27. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (5 February 2013), available at http://www.w3.org/TR/owl2-profiles/
28. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)
29. Slota, M., Leite, J., Swift, T.: Splitting and updating hybrid knowledge bases. TPLP 11(4-5), 801–819 (2011)
30. Xiao, G., Eiter, T., Heymans, S.: The DReW system for nonmonotonic dl-programs. In: Procs. of SWWS 2012. Springer Proceedings in Complexity, Springer (2013)