Artificial Intelligence 239 (2016) 7–53

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Data repair of inconsistent nonmonotonic description logic programs ☆

Thomas Eiter, Michael Fink, Daria Stepanova *

Institute of Information Systems, Vienna University of Technology, Vienna, Austria

ARTICLE INFO

Article history:
Received 14 June 2015
Received in revised form 10 June 2016
Accepted 12 June 2016
Available online 15 June 2016

Keywords:
Rules and ontologies
Answer set programming
Description logics
Inconsistency management
Nonmonotonic reasoning

ABSTRACT

Combining Description Logic (DL) ontologies and nonmonotonic rules has gained increasing attention in the past decade, due to the growing range of applications of DLs. A well-known proposal for such a combination are non-monotonic DL-programs, which support rule-based reasoning on top of DL ontologies in a loose coupling, using a well-defined query interface. However, inconsistency may easily arise as a result of the interaction of the rules and the ontology, such that no answer set (i.e., model) of a DL-program exists; this makes the program useless. To overcome this problem, we present a framework for repairing inconsistencies in DL-programs by exchanging formulas of an ontology formulated in *DL-Lite*$_{\mathcal{A}}$, which is a prominent DL that allows for tractable reasoning. Viewing the data part of the ontology as a source of inconsistency, we define program repairs and repair answer sets based on them. We analyze the complexity of the notion, and we extend an algorithm for evaluating DL-programs to compute repair answer sets, under optional selection of preferred repairs that satisfy additional constraints. The algorithm induces a generalized ontology repair problem, in which the entailment respectively non-entailment of queries to the ontology, subject to possible updates, must be achieved by a data change. While this problem is intractable in general, we identify several tractable classes of preferred repairs that are useful in practice. For the class of deletion repairs among them, we optimize the algorithm by reducing query evaluation to constraint matching, based on the novel concept of support set, which roughly speaking is a portion of the data from which entailment of an ontology query follows. Our repair approach is implemented within an answer set program system, using a declarative method for repair computation. An experimental evaluation on a suite of benchmark problems shows the effectiveness of our approach and promising results, both regarding performance and quality of the obtained repairs. While we concentrate on *DL-Lite*$_{\mathcal{A}}$ ontologies, our notions extend to other DLs, for which more general computation approaches may be used.

1. Introduction

Description Logics (DLs) [4], which emerged from semantic networks with the goal to equip respective formalisms with a clear formal semantics based on logic, nowadays play a dominant role among formalisms for Knowledge Representation

☆ Some of the results were presented in preliminary form at IJCAI 2013 [33] and ECAI 2014 [35]. This work has been supported by the Austrian Science Fund (FWF) project P24090.

* Corresponding author.
E-mail addresses: eiter@kr.tuwien.ac.at (T. Eiter), fink@kr.tuwien.ac.at (M. Fink), dasha@kr.tuwien.ac.at (D. Stepanova).

http://dx.doi.org/10.1016/j.artint.2016.06.003
0004-3702/

$$\mathcal{O} = \left\{ \begin{array}{ll} (1)\ \mathit{Child} \sqsubseteq \exists \mathit{hasParent} & (4)\ \mathit{Male}(\mathit{pat}) \\ (2)\ \mathit{Adopted} \sqsubseteq \mathit{Child} & (5)\ \mathit{Male}(\mathit{john}) \\ (3)\ \mathit{Female} \sqsubseteq \neg \mathit{Male} & (6)\ \mathit{hasParent}(\mathit{john}, \mathit{pat}) \end{array} \right\}$$

$$\mathcal{P} = \left\{ \begin{array}{l} (7)\ \mathit{ischildof}(\mathit{john}, \mathit{alex}); \quad (8)\ \mathit{boy}(\mathit{john}); \\ (9)\ \mathit{hasfather}(X, Y) \leftarrow \mathrm{DL}[\mathit{Male} \uplus \mathit{boy}; \mathit{Male}](Y), \mathrm{DL}[; \mathit{hasParent}](X, Y); \\ (10)\ \bot \leftarrow \mathit{not}\ \mathrm{DL}[; \mathit{Adopted}](X), Y_1 \neq Y_2, \mathit{hasfather}(X, Y_1), \\ \qquad\qquad \mathit{ischildof}(X, Y_2), \mathit{not}\ \mathrm{DL}[\mathit{Child} \uplus \mathit{boy}; \neg \mathit{Male}](Y_2); \\ (11)\ \mathit{contact}(X, Y) \leftarrow \mathrm{DL}[; \mathit{hasParent}](X, Y), \mathit{not}\ \mathit{omit}(X, Y); \\ (12)\ \mathit{omit}(X, Y) \leftarrow \mathrm{DL}[; \mathit{Adopted}](X), Y \neq Z, \mathit{hasfather}(X, Y), \mathit{contact}(X, Z) \end{array} \right\}$$

Fig. 1. DL-program Π over a family ontology.

and Reasoning (KRR). As such, DLs are geared towards describing domains in terms of concepts that map to sets of domain objects and their relations, as well as roles that capture relationships among domain objects. This makes DLs well-suited for representing ontologies formally and to reason about them, which has a central role in the Semantic Web vision [9]; indeed, DLs provide the formal underpinning of the Web Ontology Language (OWL), a recommended standard for expressing ontological knowledge on the web. Fueled by the success in this area, DLs have been successfully deployed to many other contexts and applications, among them reasoning about actions [6], data integration and ontology based data access [20,19], spatial reasoning [69], runtime verification and program analysis [2,53], and many others.

Most DL ontologies are fragments of classical first-order logic, and as such lack sufficient expressiveness for the requirements of certain problems; for instance, they cannot model closed-world reasoning, nor can they express nonmonotonicity; these features are often essential in practical application scenarios. Furthermore, DLs do not offer rules, which are popular in practical knowledge representation and serve a complementary aspect: while DLs are focused on specifying and reasoning about conceptual knowledge, logic rules serve well for reasoning about individuals; furthermore they target issues associated with nonmonotonic inference as well as non-determinism. To overcome these shortcomings, several extensions of DLs have been developed, e.g. [80,5,26,27,65,15,23,52,47,14] and various notions of *hybrid knowledge bases* (KBs) have been proposed to get the best out of the DL and rules worlds by combining them (see [66] and references therein). Among them, *Nonmonotonic Description Logic (DL-)programs* [37] are the most prominent approach for a loose coupling between the rules and the ontology via so-called DL-atoms, which serve as query interfaces to the ontology that support information hiding and the use of legacy software (i.e., ontology reasoners). The possibility to add information from the rules part prior to query evaluation allows for adaptive combinations.

Example 1. Consider the DL-program Π in Fig. 1, which captures information about children of a primary school and their parents in simplistic form. It is given as a pair $\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$ of an ontology $\mathcal{O}$ and a set of rules $\mathcal{P}$. The ontology $\mathcal{O}$ contains a taxonomy $\mathcal{T}$ of concepts (i.e., classes) in (1)–(3) and factual data (i.e., assertions) $\mathcal{A}$ about some individuals in (4)–(6). Intuitively, $\mathcal{T}$ states that every child has a parent, adopted child is a child, and male and female are disjoint. The rules $\mathcal{P}$ contain some further facts (7), (8) and proper rules: (9) determines fathers from the ontology, upon feeding information to it; (10) checks, informally, against them for local parent information (*ischildof*) the constraint that a child has for sure at most one father, unless it is adopted (where $\bot$ stands for falsity); finally (11)–(12) single out contact persons for children, which by default are the parents; for adopted children, fathers from the ontology are omitted if some other contact exists. The rules and the ontology interact via DL-atoms, which are the expressions starting with "DL"; e.g., $\mathrm{DL}[\mathit{Male} \uplus \mathit{boy}; \mathit{Male}](X)$ informally selects all individuals c, such that $\mathit{Male}(c)$ is provable from $\mathcal{O}$ after temporarily adding for boys the assertions that they are male in the ontology.

The semantics of DL-programs was given in the seminal paper [37] in terms of answer sets, as a generalization of the answer set semantics of nonmonotonic logic programs [46]. In this way, DL-programs are an extension of answer set programming (ASP) [18] in which the user can evaluate in the rules queries over an ontology via DL-atoms. Notably, DL-atoms enable a bidirectional information flow between the rules and the ontology, which may even be cyclic; this makes DL-programs quite expressive, and allows one to formulate advanced reasoning applications on ontologies, such as extended closed-world or terminological default reasoning [37].

On the other hand, the information flow can lead to inconsistency, i.e., that no answer set of the DL-program exists, even if the ontology and rules are perfectly consistent when considered separately; this happens in the example above, where the DL-program has no answer set. An inconsistent DL-program yields no information and is of no use for constructive problem solving; it may be viewed as broken and in need of an appropriate management of this situation. Systems for evaluating DL-programs, among them dlvhex[1] and DReW,[2] however can not resolve inconsistencies easily; this is clearly a drawback for their deployment to applications.

[1] www.kr.tuwien.ac.at/research/systems/dlvhex.
[2] www.kr.tuwien.ac.at/research/systems/drew.