Inconsistency Management for Description Logic Programs and Beyond[⋆]

Thomas Eiter, Michael Fink, and Daria Stepanova

Institute of Information Systems
Vienna University of Technology
Favoritenstraße 9-11, A-1040 Vienna, Austria
{dasha,eiter,fink}@kr.tuwien.ac.at

Abstract. Description logic programs are a declarative approach to access ontological knowledge bases through a query interface, and to combine the query results using rules that can be nonmonotonic. Noticeably, a bidirectional information flow between the rules and the ontology is supported, which opens the possibility of sophisticated data exchange and advanced reasoning tasks on top of ontologies. As it happens, inconsistency may arise from the interplay of the rules and the ontology. We consider this issue and discuss different origins of inconsistency, as well as approaches to deal with it. While recent progress has been made, several issues remain to be explored; among them is inconsistency management for generalizations of description logic programs, and in particular for HEX programs, where this issues is largely unexplored and challenging, the more if distributed or web-based evaluation scenarios are considered.

1 Overview

The need for expressive formalisms that embrace both ontological knowledge bases, and rules has led to a number of proposals that are based on different grounds (see [10] for some overview). Among them are nonmonotonic description logic programs (briefly dl-programs) [4], which provide a declarative approach to access ontologies through a query interface and to combine the query results using rules that can be nonmonotonic. Roughly speaking, a dl-program consists of a pair (O, P) of an ontological knowledge base (briefly, ontology) O and a rule set P, where in the bodies of the rules in P so called dl-atoms of the form $DL[\lambda, Q](\boldsymbol{t})$ may occur. Informally, $Q(\boldsymbol{t})$ is a query to O, and λ is a list of update operations $S\ op\ p$ which specify assertions $S(\boldsymbol{c})$ resp. $\neg S(\boldsymbol{c})$ for O depending on the rules predicate p; these assertions are calculated from the valuations of p and temporally added to O before the query $Q(\boldsymbol{t})$ is evaluated.

Noticeably, this mechanism allows for a bidirectional information flow between the rules and the ontology, which opens the possibility of sophisticated data exchange and advanced reasoning tasks on top of ontologies. It has been been fruitfully generalized to so called HEX programs [5], which provide a view-based access of external information sources from rules beyond ontologies through external atoms of the form $\&e[\boldsymbol{i}](\boldsymbol{t})$, where $\&e$ is an external predicate and $\boldsymbol{i}$ is a list of input parameters (which are terms).

⋆ This work was supported by the Austrian Science Fund (FWF) projects P20480 and P24090.

Different semantics for dl-programs have been defined (cf. e.g. [13]), extending answer set and well-founded semantics. As it happens, the information flow between the rules and the ontology can have unforseen effects and cause inconsistency, such that no answer set or model exists; the dl-program thus yields no information and is unusable.

Possible sources of inconsistency in a dl-program (O, P) are naturally

(1) the rules P,

(2) the ontology O, and

(3) the interface $DL[\lambda, Q](\boldsymbol{t})$ between P and O,

and they may also be intermingled. Different approaches have been pursued to deal with the problem, in particular to tolerate inconsistency and program repair.

Inconsistency tolerance (see e.g., [7; 1]) aims at suppressing or weakening information that leads to inconsistency in model building. In [11], rules with dl-atoms that amount to queries over inconsistent ontologies are suppressed, while [6] extends paraconsistent semantics of ordinary answer set programs to dl-programs, exploiting a logic-based view of dl-programs by introducing assumptions. The semantics pays particular attention to inconsistency due to the lack of stability in models (termed paracoherence), and it is in line with paraconsistent semantics for description logics [9]. However, one aspect in which this approach needs refinement is modularity; layered program evaluation as for ordinary nonmonotonic logic programs is not compatible with unrestricted assumptions, and thus structural information needs to be taken into account.

Repairing dl-programs, i.e., to change formulas to obtain consistency (which is a natural and ubiquitous approach in knowledge representation) was only most recently attacked. Here, due to various sources of inconsistencies, different possibilities exist. The recent work [3] concentrates on the repair of the ontology O, taking the view that the rules are correct (as they are on top of the ontology, this may be separately assessed; moreover, ontology repair is well-researched (cf. [8; 2]), while repair of nonmonotonic rules is less developed (cf. [12]). In particular, [3] restricts changes to the assertional part (the ABox), and formally defines repairs and repair answer sets as changes that enable answer sets; refined notions incorporate criteria to select preferred repairs, with a focus on properties that allow to extend existing evaluation methods for dl-programs. At the heart of the method is a generalized ontology repair problem, which roughly speaking asks for an ABox repair that effects certain positive and negative query answers; while intractable in general, it can be polynomially solved in relevant non-trivial settings.

On the other hand, repair of the rules part and repair of the interface remain to be considered. The former subsumes repair of ordinary nonmonotonic programs, and thus poses a challenge as such, especially if repair goes beyond merely dropping rules. Repair of the interface might be addressed in different ways. One possibility is to modify the update specification λ in a DL-atom $DL[\lambda; Q](\boldsymbol{t})$ and/or the query $Q(\boldsymbol{t})$ to effect a different information flow between the rules and the ontology. However, the search space for changes is large and needs to be limited, and user intervention will most likely be needed. Another possibility is to change query evaluation: rather than simply expanding the ontology O with assertions assembled from the update specification λ, one incorporates them into O in a way such that consistency is preserved, using a revision or update operatord; it remains to identify suitable such operators.

In the context of HEX programs, the issues above are lifted to a more abstract level. Here we have in general multiple (not a single) external sources, which are not necessary ontologies and even if so they may be in different formats; their semantics is defined using Boolean-valued functions $f_{\&}$ associated with external predicates $\&e$ of external atoms $\&e[\boldsymbol{i}](\boldsymbol{t})$, which model membership of a tuple $\boldsymbol{t}$ in the "result" of evaluating $\&e$ with "input" parameters $\boldsymbol{i}$. As for sources of inconsistency in a HEX program, external sources may vanish to appear as such, as in lack of a logical semantics the notion of "inconsistency" is inapplicable; furthermore, repair of an external source may be limited (if not impossible). Hence, paraconsistent semantics and/or rule or interface repair are expected to be more promising. While the abstract nature of HEX programs, in which the external atoms are viewed as black boxes, allows to access any kind of external source, for specific settings domain information about the external sources may be useful. To this end, properties of the evaluation functions $f_{\&e}$ may be taken into account, as well as the possibility to access a source via functions of an API (other than those used in the program) in the style of an abstract data type. Indeed, external functions may serve to provide traditional data structures such as strings, but also more complex ones like trees or routes computed by a route planner. However, work on this is in an embryonic state.

In conclusion, while recent progress has been in inconsistence management of dl-programs, open issues remain to be explored; among them is to consider generalizations of dl-programs, and in particular HEX programs, where this issue is largely unexplored and challenging, the more if distributed or web-based evaluation scenarios are considered.

References

1. Bertossi, L.E., Hunter, A., Schaub, T. (eds.): Inconsistency Tolerance, Lecture Notes in Computer Science (LNCS), vol. 3300. Springer (2005)
2. Bienvenu, M.: On the complexity of consistent query answering in the presence of simple ontologies. In: Proc. 26th Conf. Artificial Intelligence. pp. 705–711. AAAI Press (2012)
3. Eiter, T., Fink, M., Stepanova, D.: Data repair of inconsistent dl-programs. In: Proc. 23nd Int'l Joint Conf. Artificial Intelligence (IJCAI-13). AAAI Press/IJCAI (2013), to appear
4. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. AIJ 172, 1495–1539 (2008)
5. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: Proc. 19th Int'l Joint Conf. Artificial Intelligence (IJCAI-05). pp. 90–96. Professional Book Center (2005)
6. Fink, M.: Paraconsistent hybrid theories. In: Proc. 13th Int'l Conf. Principles of Knowledge Representation and Reasoning (KR 2012). pp. 141–151. AAAI Press (2012)
7. Hunter, A.: Paraconsistent logics. In: Handbook of Defeasible Reasoning and Uncertainty Management Systems, vol. 2, pp. 11–36. Kluwer (1998)
8. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Proc. RR 2010, LNCS 6333, pp. 103–117. Springer (2010)
9. Ma, Y., Hitzler, P., Lin, Z.: Paraconsistent reasoning for expressive and tractable description logics. In: Proc. DL 2008, CEUR Workshop Proc., vol. 353. CEUR-WS.org (2008)
10. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)
11. Pührer, J., Heymans, S., Eiter, T.: Dealing with inconsistency when combining ontologies and rules using DL-programs. In: Proc. ESWC 2010 (I), pp. 183–197. LNCS 6088, (2010)
12. Sakama, C., Inoue, K.: An abductive framework for computing knowledge base updates. Theory and Practice of Logic Programming 3(6), 671–713 (2003)
13. Wang, Y., You, J.H., Yuan, L.Y., Shen, Y.D., Zhang, M.: The loop formula based semantics of description logic programs. Theor. Comput. Sci. 415, 60–85 (2012)