The DReW System for Nonmonotonic DL-Programs*

Guohui Xiao[1], Thomas Eiter[1], and Stijn Heymans[2]

[1] Institute of Information Systems 184/3
Vienna University of Technology
Favoritenstraße 9–11, A–1040 Vienna, Austria
{xiao,eiter}@kr.tuwien.ac.at

[2] Artificial Intelligence Center, SRI International
Menlo Park, CA 94025, United States
stijn.heymans@sri.com

Abstract. Nonmonotonic DL-programs provide a loose integration of Description Logic (DL) ontologies and Logic Programming (LP) rules with negation, where a rule engine can query an ontology with a native DL reasoner. However, in most systems for DL-programs, the overhead of an external DL reasoner might be considerable. Datalog-*rewritable* DL ontologies, such as most fragments of OWL 2 RL, OWL 2 EL, and OWL 2 QL, can be rewritten to Datalog programs, so that DL-programs can be reduced to Datalog$^{\neg}$, i.e, Datalog with negation, under both well-founded and answer set semantics. We developed the reasoner DReW that uses the Datalog-rewriting technique. In addition to DL-programs, DReW can also answer conjunctive queries under DL-safeness conditions over Datalog-rewritable ontologies, as well as reason on terminological default logics over such ontologies.

1 Introduction

Nonmonotonic DL-programs [5] provide a loose integration of Description Logic (DL) ontologies and Logic Programming (LP) rules with negation, where a rule engine can query an ontology using a native DL reasoner. For DL-programs over tractable DL ontologies under well-founded semantics, the reasoning problem is tractable [4]. However, even for tractable DL-programs, the overhead of an external DL reasoner might be considerable.

To remedy the overload of calling external DL reasoners, we proposed the notion of Datalog-rewritability in [8]. Intuitively, a Datalog-rewritable ontology can be rewritten to a Datalog program in a modular way with respect to data access. Moreover, DL-programs over such Datalog-rewritable ontologies can then be reduced to Datalog$^{\neg}$ programs, i.e., to Datalog with negation. A particular DL that is polynomially Datalog-rewritable is $\mathcal{LDL}^{+}$, which is essentially an extension of OWL 2 RL and was also proposed in [8]. Reasoning in $\mathcal{LDL}^{+}$ is tractable, under both data and combined complexity. Based on [9], it was easily established that OWL 2 EL ontologies (modulo data

* This work has been partially supported by the Austrian Science Fund (FWF) project P20840 and EU Project OntoRule (FP7 231875).

types) are also polynomial Datalog-rewritable [7]. OWL 2 QL is even FO rewritable[1], and thus Datalog-rewritable.

Based on the concept of Datalog-rewriting, we developed a reasoner DReW (Datalog ReWriter)[1] [12, 7], which rewrites DL-programs over Datalog-rewritable ontologies to Datalog$^\neg$ programs, and calls an underlying rule-based reasoner, currently DLV, to perform the actual reasoning. DL-programs are a very expressive language. Several formalisms, e.g., conjunctive query (CQ) answering under DL-safeness restriction [10] and terminological default reasoning [5], can be rewritten to DL-programs. We support these two reasoning services directly in the DReW system.

2 DL-Programs

Informally, a DL-program (Σ, P) consists of a DL knowledge base (or ontology) Σ over predicates $\mathcal{P}_o$ and a Datalog$^\neg$ program P over predicates $\mathcal{P}_p$ distinct from $\mathcal{P}_o$, where P may contain queries to Σ via so called DL-atoms. Due to space constraints, we refer to [5, 4] for the formal syntax and semantics of dl-programs and confine here to illustrate the intuition behind on an example from [3].

Example 1. Suppose that an existing network must be extended by new nodes. The knowledge base Σ contains information about existing nodes $(n_1, \dots, n_5)$ and their interconnections as well as a definition of "overloaded" nodes (concept $\mathit{HighTrafficNode}$), which are nodes with more than three connections:

$$\begin{aligned}
\geq 1.\mathit{wired} \sqsubseteq \mathit{Node}; \quad \top \sqsubseteq \forall \mathit{wired}.\mathit{Node}; \quad \mathit{wired} = \mathit{wired}^-;\\
\geq 4.\mathit{wired} \sqsubseteq \mathit{HighTrafficNode}; \quad n_1 \neq n_2 \neq n_3 \neq n_4 \neq n_5;\\
\mathit{Node}(n_1); \quad \mathit{Node}(n_2); \quad \mathit{Node}(n_3); \quad \mathit{Node}(n_4); \quad \mathit{Node}(n_5);\\
\mathit{wired}(n_1, n_2); \quad \mathit{wired}(n_2, n_3); \quad \mathit{wired}(n_2, n_4);\\
\mathit{wired}(n_2, n_5); \quad \mathit{wired}(n_3, n_4); \quad \mathit{wired}(n_3, n_5).
\end{aligned}$$

The following program P evaluates possible combinations of connecting the new nodes:

$$\begin{aligned}
&\mathit{newnode}(x_1). && (1)\\
&\mathit{newnode}(x_2). && (2)\\
&\mathit{overloaded}(X) \leftarrow \mathrm{DL}[\mathit{wired} \uplus \mathit{connect}; \mathit{HighTrafficNode}](X). && (3)\\
&\mathit{connect}(X, Y) \leftarrow \mathit{newnode}(X), \mathrm{DL}[\mathit{Node}](Y), \mathit{not\ overloaded}(Y),\\
&\qquad\qquad \mathit{not\ excl}(X, Y). && (4)\\
&\mathit{excl}(X, Y) \leftarrow \mathit{connect}(X, Z), \mathrm{DL}[\mathit{Node}](Y), Y \neq Z. && (5)\\
&\mathit{excl}(X, Y) \leftarrow \mathit{connect}(Z, Y), \mathit{newnode}(Z), \mathit{newnode}(X), Z \neq X. && (6)\\
&\mathit{excl}(x_1, n_4). && (7)
\end{aligned}$$

The facts (1)-(2) (bodyless rules) define the new nodes to be added. Rule (3) imports knowledge about overloaded nodes in the existing network, taking new connections

[1] http://www.kr.tuwien.ac.at/research/systems/drew

already into account. Rule (4) connects a new node to an existing one, provided the latter is not overloaded and the connection is not to be disallowed, which is specified by Rule (5) (there must not be more than one connection for each new node) and Rule (6) (two new nodes cannot be connected to the same existing one). Rule (7) states a specific condition: node x_1 must not be connected with n_4.

The meaning of DL-programs is given by formal semantics, among which, answer set semantics [5] and well-founded semantics [4] are widely used (see [11] for a survey). The DL-program (Σ, P) in Example 1 has four strong answer sets : $M_1 = \{\mathit{connect}(x_1, n_1),\ \mathit{connect}(x_2, n_4), \ldots\}$, $M_2 = \{\mathit{connect}(x_1, n_1),\ \mathit{connect}(x_2, n_5), \ldots\}$, $M_3 = \{\mathit{connect}(x_1, n_5),\ \mathit{connect}(x_2, n_1),\ \ldots\}$, and $M_4 = \{\mathit{connect}(x_1, n_5),\ \mathit{connect}(x_2, n_4), \ldots\}$. Note that the ground DL-atom

$$\mathrm{DL}[\mathit{wired} \uplus \mathit{connect}; \mathit{HighTrafficNode}](n_2)$$

from rule (3) is true in any partial interpretation of P. According to the proposed well-founded semantics for DL-programs in [4], the atom $\mathit{overloaded}(n_2)$ is thus true in the well-founded model.

3 Reasoning with DL-Programs by Datalog$^\neg$ Rewriting

We present the rewriting approach in DReW by means of Example 1. This is achieved by carefully rewriting different components of DL-programs into Datalog ($^\neg$) rules.

1. Rewriting Ontology into Datalog. For Datalog-rewritable DLs, the instance query problem can be reduced to the query in Datalog. The DL component Σ in Example 1 is in OWL 2 RL, which is Datalog-rewritable. We transform Σ to Datalog program $\Phi_{RL}(\Sigma)$:

For TBox axiom $\geq 1.\mathit{wired} \sqsubseteq \mathit{Node}$, we add the following rule to $\Phi_{RL}(\Sigma)$:

$$\mathit{Node}(X) \leftarrow \mathit{wired}(X, Y).$$

For TBox axiom $\top \sqsubseteq \forall \mathit{wired}.\mathit{Node}$, we add the following rule:

$$\mathit{Node}(Y) \leftarrow \mathit{wired}(X, Y).$$

For TBox axiom $\mathit{wired} = \mathit{wired}^-$, we have

$$\mathit{wired}(X, Y) \leftarrow \mathit{wired}(Y, X).$$

For TBox axiom $\geq 4\mathit{wired} \sqsubseteq \mathit{HighTrafficNode}$, we have

$$\begin{aligned}\mathit{HighTrafficNode}(X) \leftarrow\ & \mathit{wired}(X, Y_1), \mathit{wired}(X, Y_2), \mathit{wired}(X, Y_3), \mathit{wired}(X, Y_4),\\ & Y_1 \neq Y_2, Y_1 \neq Y_3, Y_1 \neq Y_4, Y_2 \neq Y_3, Y_2 \neq Y_4, Y_3 \neq Y_4.\end{aligned}$$

Finally, the ABox assertions in Σ (e.g., $\mathit{Node}(n_1)$) are transformed to Datalog facts directly. Note that after transformation, $n_i \neq n_j$, $1 \leq i < j \leq 5$, is dropped because of the Unique Name Assumption (UNA) adopted by Datalog.

2. Duplicating Rewritten Ontologies according to the DL-Inputs. Note that each DL-atom sends up a different input to Σ and that entailments for each different input might be different. To this purpose, we copy $\Phi_{RL}(\Sigma)$ to new disjoint equivalent versions for each DL-input, i.e., for each distinct DL-input λ, we define a new program $\Phi_{RL,\lambda}(\Sigma)$ that results from replacing all the predicates by a λ-subscripted version.

Thus, for the set $\Lambda_P = \{\lambda_1 = \emptyset, \lambda_2 = \mathit{wired} \uplus \mathit{connect}\}$ of DL-atoms, we have $\Phi_{RL,\lambda_1}(\Sigma) = \{\mathit{Node}_{\lambda_1}(X) \leftarrow \mathit{wired}_{\lambda_1}(X, Y), \ldots\}$ and $\Phi_{RL,\lambda_2}(\Sigma) = \{\mathit{Node}_{\lambda_2}(X) \leftarrow \mathit{wired}_{\lambda_2}(X, Y), \ldots\}$

3. Rewriting DL-Rules to Normal Rules. To rewrite DL-rules P into normal rules P^{ord}, we simply replace each DL-atom $DL[\lambda; Q](\mathbf{t})$ by a new atom $Q_\lambda(\mathbf{t})$. For example, rule (3) is replaced by

$$\mathit{overloaded}(X) \leftarrow \mathit{HighTrafficNode}_{\lambda_2}(X).$$

4. Rewriting DL-Atoms to Datalog **Rules.** The inputs in the DL-atoms Λ_P can then be encoded as rules $\rho(\Lambda_P)$:

$$\mathit{wired}_{\lambda_2}(X, Y) \leftarrow \mathit{connect}(X, Y).$$

5. Calling Datalog Reasoner. Now we have transformed all the components into a Datalog$^\neg$ program $\Psi(\Sigma, P) = \Phi_{RL,\lambda_1}(\Sigma) \cup \Phi_{RL,\lambda_2}(\Sigma) \cup P^{ord} \cup \rho(\Lambda_P)$. We can send it to a datalog engine, e.g. DLV, and compute the answer set or well-founded models.

4 Reasoning with Conjunctive Queries and Terminological Default Logics

DL-programs are a very expressive language. Several formalisms, e.g., conjunctive query (CQ) answering under DL-safeness restriction [10] and terminological default reasoning [5], can be captured by DL-programs. We support these two reasoning tasks directly in the DReW system.

4.1 Conjunctive Query Answering under DL-safeness Condition

A conjunctive query q is a rule of the following form:

$$\mathit{ans}(X_1, \ldots, X_n) \leftarrow C_1(Y_1), \ldots, C_m(Y_m), r_1(Z_{11}, Z_{12}), \ldots, r_k(Z_{k1}, Z_{k2}),$$

where C_i's and r_i's are concepts and roles in the ontology, respectively, and ans is a fresh predicate name.

When applying the DL-safe condition [10], every such query can be equivalently converted to a DL-rule by replacing every atom $Q(\mathrm{X})$ with the DL-atom $\mathrm{DL}[Q](\mathrm{X})$ having empty input list.

$$\begin{aligned} \mathit{ans}(X_1, \ldots, X_n) \leftarrow\ & \mathrm{DL}[C_1](Y_1), \ldots, \mathrm{DL}[C_m](Y_m), \\ & \mathrm{DL}[r_1](Z_{11}, Z_{12}), \ldots, \mathrm{DL}[r_k](Z_{k1}, Z_{k2}), \end{aligned}$$

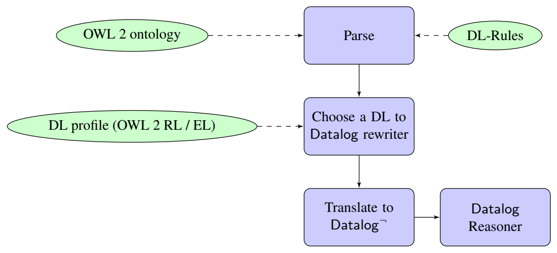
OWL 2 ontology
Parse
DL-Rules
DL profile (OWL 2 RL / EL)
Choose a DL to Datalog rewriter
Translate to Datalog$^\neg$
Datalog Reasoner

Fig. 1. DReW Control Flow of DReW with DL-programs

Example 2. The following CQ retrieves pairs of wired $\mathit{HighTrafficNode}\ X$ and Y:

$$ans(X, Y) \leftarrow \mathit{HighTrafficNode}(X), \mathit{wired}(X, Y), \mathit{HighTrafficNode}(Y)$$

It can be converted to a DL-rule:

$$ans(X, Y) \leftarrow \mathrm{DL}[\mathit{HighTrafficNode}](X), \mathrm{DL}[\mathit{wired}](X, Y), \mathrm{DL}[\mathit{HighTrafficNode}](Y)$$

4.2 Reasoning with Terminological Default

The bidirectional flow of knowledge between a DL ontology and a logic program enables a variety of possibilities. One application for DL-programs is terminological default theory [2, 5]. Here, Reiter-style default rules are applied to named individuals explicitly occurring in the knowledge base; the classic birds&penguins example can be captured by rule $Bird(X) : Flier(X)/Flier(X)$ (informally, birds fly by default).

The intuition of reduction of terminological default logics to DL-programs is to use in/out predicates to guess the default extension and to check the default extension by different DL-inputs; see [7] for more details.

5 System Architecture and Usage

Fig. 1 shows a schematic overview of the components of DReW in charge of reasoning with DL-programs by Datalog rewriting. DReW is written in Java using OWL API[2] for parsing ontologies. The underlying Datalog engine we use is DLV[3] which supports both answer set semantics and well-founded semantics. For the ontology component, the current version DReW supports OWL 2 RL and OWL 2 EL (modulo data types). At present, DReW implements dl-programs under well-founded semantics; support for answer set semantics will be implemented soon.

DReW is distributed as several jars and scripts. It can be used both as a java library and from the command line. Some of the command line options are listed as follows:

```
% drew [Rew] <Ontology.owl> [Rule] [-dlv /path/to/dlv]
```

Option `[Rew]` is the datalog rewriter for DL:

[2] http://owlapi.sourceforge.net/
[3] http://www.dlvsystem.com/dlvsystem/index.php/DLV

```
-rl      Using OWL 2 RL rewriting (default)
-el      Using OWL 2 EL rewriting
```

Option `[Rule]` specifies which rule formalism is used:

```
-dlp     <rule.dlp>      Reasoning with DL-programs
-sparql  <query.sparql>  Conjunctive query answering
-df      <default.df>    Terminological defaults
```

6 Conclusions and Outlook

Interesting classes of DLs are Datalog-rewritable, and reasoning with DL-programs over such DLs can be reduced to Datalog$^{\neg}$ under well-founded semantics and well-founded semantics. This reduction is implemented in DReW system which avoids calling external DL reasoner and runs (sometimes significantly) faster than the traditional hybrid reasoner over Datalog-rewritable ontologies (cf. [12, 7]). Conjunctive query answering under (DL-safeness) and terminological default reasoning can be reduced to DL-programs, and both tasks are directly supported by DReW system.

Future work is planned in two directions. One direction is to support further DLs, e.g., OWL 2 QL and Horn-$\mathcal{SHIQ}$ [6]; the other is to support tailored non-monotonic reasoning modalities, e.g. closed world reasoning [5].

References

1. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *Journal of Artificial Intelligence Research*, 36:1–69, 2009.
2. F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. *J. Autom. Reasoning*, 14(1):149–180, 1995.
3. W. Drabent, T. Eiter, G. Ianni, T. Krennwallner, T. Lukasiewicz, and J. Maluszynski. Hybrid reasoning with rules and ontologies. In F. Bry and J. Maluszynski, editors, *REWERSE*, volume 5500 of *Lecture Notes in Computer Science*, pages 1–49. Springer, 2009.
4. T. Eiter, G. Ianni, T. Lukasiewicz, and R. Schindlauer. Well-founded semantics for description logic programs in the Semantic Web. *ACM Trans. Comput. Log.*, 12(2):11, 2011.
5. T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence*, 172(12-13):1495–1539, 2008.
6. T. Eiter, M. Ortiz, M. Simkus, T. Tran, and G. Xiao. Query rewriting for Horn-SHIQ plus rules. In *Proc. of AAAI 2012*. AAAI.
7. T. Eiter, T. Krennwallner, P. Schneider, and G. Xiao. Uniform Evaluation of Nonmonotonic DL-Programs. In *FoIKS'12*, pages 1–22. Springer.
8. S. Heymans, T. Eiter, and G. Xiao. Tractable reasoning with DL-programs over datalog-rewritable description logics. In *Proc. of ECAI 2010*. IOS Press.
9. M. Krötzsch. Efficient inferencing for OWL EL. In *JELIA*, LNCS 6341:234-246, 2010.
10. B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *Journal of Web Semantics*, 3(1):41–60, July 2005.
11. Y. Wang, J.-H. You, L.-Y. Yuan, Y.-D. Shen, and M. Zhang. The loop formula based semantics of description logic programs. *Theor. Comput. Sci.*, 415:60–85, 2012.
12. G. Xiao, S. Heymans, and T. Eiter. DReW: a reasoner for datalog-rewritable description logics and dl-programs. In *Informal Proc. 1st Int'l Workshop on Business Models, Business Rules and Ontologies (BuRO 2010)*, 2010.