NLP-DL: A Knowledge-Representation System for Coupling Nonmonotonic Logic Programs with Description Logics*

Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits
Institut für Informationssysteme, Technische Universität Wien,
Favoritenstraße 9–11, A-1040 Vienna, Austria
{eiter, ianni, roman, tompits}@kr.tuwien.ac.at

Abstract

Combining description logic systems with other reasoning systems, possibly over the Web, has become an important research issue and calls for advanced methods and algorithms. Among several approaches in this direction are *nonmonotonic description logic programs*, which couple description logics and nonmonotonic logic programs under generalized versions of the answer-set semantics as well as of the well-founded semantics, which are the predominant semantics for such programs. We briefly report here on the current prototype of the NLP-DL system, implementing these semantics, which couples state-of-the-art engines for description logics and nonmonotonic logic programs.

1 Introduction

The *Web Ontology Language* (*OWL*) is a W3C recommended standard for the *Ontology Layer* of the Semantic Web,[1] whose major sublanguages *OWL Lite* and *OWL DL* and are based on the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, respectively. Current and future efforts in building the Semantic Web are aimed at the *Rules*, *Logic*, and *Proof Layers* on top of the Ontology Layer. As they should offer sophisticated representation and reasoning capabilities, this requests the need for integrating the Rules and the Ontology Layer. Indeed, description logics do not offer rules, and powerful extensions with rich knowledge representation constructs (such as negation as failure) are non-trivial, both from a semantic as well as from a computational point of view, since major reasoning tasks quickly become undecidable.

Several proposals for combining description logics with rule-based languages exist, cf. [1] and references therein. Among them, *nonmonotonic description logic programs* (or *dl-programs*) [3] are a novel method to couple description logics with nonmonotonic logic programs. Roughly, such a program is a pair $KB = (L, P)$, containing a knowledge base L in a description logic, i.e. a finite set of description logic axioms (in $\mathcal{SHIF}(\mathbf{D})$ resp., $\mathcal{SHOIN}(\mathbf{D})$) representing knowledge about concepts, roles, and individuals, and a finite set P of nonmonotonic logic-programming rules (since *negation as failure* is supported) called *dl-rules*. These rules are *extended logic program rules* [9] but may additionally contain *queries to* L in their bodies. Noticeably, such a query may involve input from P to L; hence, a *bidirectional flow of information between* P *and* L is facilitated. Thus, dl-programs allow for building nonmonotonic rules on top of ontologies. Importantly, dl-programs are decidable [3].

Semantically, dl-programs fully support encapsulation and privacy of the components, in the sense that logic programs and description logic reasoning are *technically separated* and *only interfacing details* need to be known. This also fosters the view of dl-programs providing a *rule-based glue for combining inferences from a description logic knowledge base*. Computationally, this encapsulation means that dl-programs can be evaluated by coupling existing reasoners to a hybrid reasoning system.

Here, we briefly describe our operational prototype of the NLP-DL system implementing dl-programs, which has been developed by coupling the two state-of-the-art solvers DLV [6] for nonmonotonic logic programs and RACER [7] for description logics. Due to this combination, NLP-DL is a powerful platform for expressive (yet decidable) knowledge representation and reasoning, featuring (i) ontologies, (ii) rules under negation as failure (a.k.a. *default negation*), (iii) *strong* ("classical") negation besides negation as failure, and (iv) constraints (which can be easily emulated).

2 Description Logic Rules

A *dl-rule* is an expression of the form

$$a \leftarrow b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m\,,\ m \geq k \geq 0\,, \quad (1)$$

where a is a classical literal and each b_i is either a classical literal in a function-free first-order language, or a *dl-atom*, which is of the form $DL[S_1\, op_1 p_1, \ldots, S_m\, op_m\ p_m; Q](\mathbf{t})$, $m \geq 0$, where each S_i is either a concept or a role name, $op_i \in \{\uplus, \cup\!\!\!\!-, \cap\!\!\!\!-\}$, p_i is a unary resp. binary "input" predicate symbol, and $Q(\mathbf{t})$ is a *dl-query*. Informally, a dl-atom of the above form amounts to the query $Q(\mathbf{t})$ which is evaluated as a subjunctive statement on the underlying description logic knowledge base L. The operator $op_i = \uplus$ (resp., $op_i = \cup\!\!\!\!-$) increases S_i (resp., $\neg S_i$) in L by the extent of predicate p_i in an interpretation (given by a consistent set of ground literals), while $op_i = \cap\!\!\!\!-$ constrains S_i to p_i. For details, see [3].

For dl-programs, two basic types of semantics have been defined: the *answer-set semantics* [3] and the *well-founded semantics* (*WFS*) [4] (under necessary restrictions). They are

*This work was partially supported by the Austrian Science Fund (FWF) under grant P17212 and by the European Commission through the IST REWERSE Network of Excellence (IST-506779).

[1] See www.w3.org/TR/2004/REC-owl-features-20040210/.

conservative extensions of the standard answer-set semantics [9] and the standard WFS [11], respectively, and share many of their appealing properties.

We note that the answer-set semantics may yield no, one, or multiple models (i.e., answer sets) in general, while the WFS yields a canonical (three-valued) model. Thus, under the answer-set semantics, for query answering, *brave* and *cautious reasoning* (i.e., truth in some resp. all models) are considered in practice, depending on the application.

3 System Prototype

A fully operational NLP-DL prototype, ready for experiments, is available through a Web interface at `http://www.kr.tuwien.ac.at/staff/roman/semweblp/`.

The system accepts dl-programs as input, given by an ontology formulated in OWL-DL (as processed by RACER) and a set of dl-rules in the language above, where $\leftarrow$, $\uplus$, $\cup\!\!\!-$, and $\cap\!\!\!-$ are written as ":-", "+=", "–=", and "?=", respectively. It features the following reasoning tasks:

- Computing models (answer sets or the well-founded model) of a given dl-program. For computing the answer sets, a preliminary computation of the well-founded model may be issued, which semantically approximates the answer sets—this is exploited for optimization.
- Evaluating a given query on the given dl-program. Under the answer-set semantics, both *brave reasoning* and *cautious reasoning* are available.

The system architecture integrates the external DLV and RACER engines, the latter being embedded into a caching module, a well-founded semantics module, an answer-set semantics module, a pre-processing module, and a post-processing module.

Each internal module has been implemented in the PHP scripting language; the overhead is insignificant, provided that most of the computing power is devoted to the execution of the two external reasoners. In particular, efficient usage of RACER is critical for the system performance. Respective techniques, mainly based on caching query results and exploiting monotonicity of description-logic reasoning, are described in [2]. The current prototype, whose development is ongoing, already incorporates several of these techniques, which are implemented in the caching module.

4 Examples

A suite of various reasoning examples in different domains, showing the applicability of NLP-DL, including partial applications of the *closed-world assumption*, incomplete information, and defaults, is available on the system Web page. We consider here briefly an example in the Web-service domain.

The OWL-S ontology for describing Web services has been recently submitted as a W3C standard [8]. Matchmaking of OWL-S (formerly DAML-S) services is an important target to be achieved, and the first on-purpose techniques are being published (see, e.g., [10]). The small example program below shows how our language can be adopted for specifying matching policies, focusing on processes of services about food and drink recommendations.

```
% Extracts all known processes from the knowledge base.
processes(X) :-
   DL[http://www.daml.org/services/owl-s/1.1/Process.owl
             #AtomicProcess](X).

% The concept 'servesWhiteWine' extract all those pro-
% cesses which are known to be giving WhiteWine as
% output and taking some kind of food as input.

servingWhiteWine(W) :- DL[servesWhiteWine](W).

% The concept 'takesFish' extracts all those processes
% which are known to have a process taking fish as
% input and giving some wine as output.

takingFish(F) :- DL[takesFish](F).

% By default (unless proven otherwise), a 'takingFish'
% process always has a WhiteWine in the output.
% Thus, it can be added to servingWhiteWine.

 aServingWhiteWine(W) :-
           takingFish(W), not -aServingWhiteWine(W).
-aServingWhiteWine(W) :-
           DL[servesWhiteWine+=aServingWhiteWine;
           -servesWhiteWine](W), takingFish(W).

servingWhiteWine(W) :- aServingWhiteWine(W).
```

The rule for `aServingWhiteWine` states that a known process `W` for taking fish should suggest a white wine by default. The subsequent rule checks, feeding the default conclusions into the ontology, whether its consistency is preserved (if not, then no default conclusion could have been drawn). The answer-set semantics enforces that, as desired, defaults are applied to a largest extent.

Evaluated against the input ontology, the brave (resp., cautious) consequences of form `servingWhiteWine(W)` are all those processes that suggest white wine under a credulous (resp., skeptical) assumption. The resulting set may be (much) larger than under classical inference, given that such default information can not be captured by it. Process selection may then be based on this set, or on defaults if no process provably suggests white wine, using a modified program.

References

[1] Antoniou, G., *et al.*: Combining rules and ontologies. A survey. Technical Report IST506779 REWERSE, Linköping/I3-D3/D/PU/a1, Linköping University (2005).

[2] Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Nonmonotonic Description Logic Programs: Implementation and Experiments. In: Proc. LPAR 2004. LNCS 3452, (2005).

[3] Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining Answer Set Programming with Description Logics for the Semantic Web. In: Proc. KR 2004. (2004) 141–151.

[4] Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Well-founded Semantics for Description Logic Programs in the Semantic Web. In: Proc. RuleML 2004. LNCS 3323, (2004).

[5] Fensel, D., Wahlster, W., Lieberman, H., Hendler, J., eds.: Spinning the Semantic Web: Bringing the World Wide Web to its Full Potential. MIT Press (2002).

[6] Leone, N., *et al*: The DLV System for Knowledge Representation and Reasoning. ACM Trans. Comp. Logic, to appear.

[7] Haarslev, V., Möller, R.: RACER System Description. In: Proc. IJCAR 2001. LNCS 2083.

[8] W3C: OWL-S: Semantic Markup for Web Services. W3C Member Submission. Available at `http://www.w3.org/Submission/OWL-S/`.

[9] Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Deductive Databases. New Generation Computing **17** (1991) 365–387.

[10] Sycara, K., Paolucci, M., Soudry J., and Srinivasan N.: Dynamic Discovery and Coordination of Agent-Based Semantic Web Services. IEEE Internet Comp., 8 (3), 66-73, 2004.

[11] Van Gelder, A., Ross, K.A., Schlipf, J.S.: The Well-Founded Semantics for General Logic Programs. Journal of the ACM **38** (1991) 620–650.

System Demo Description

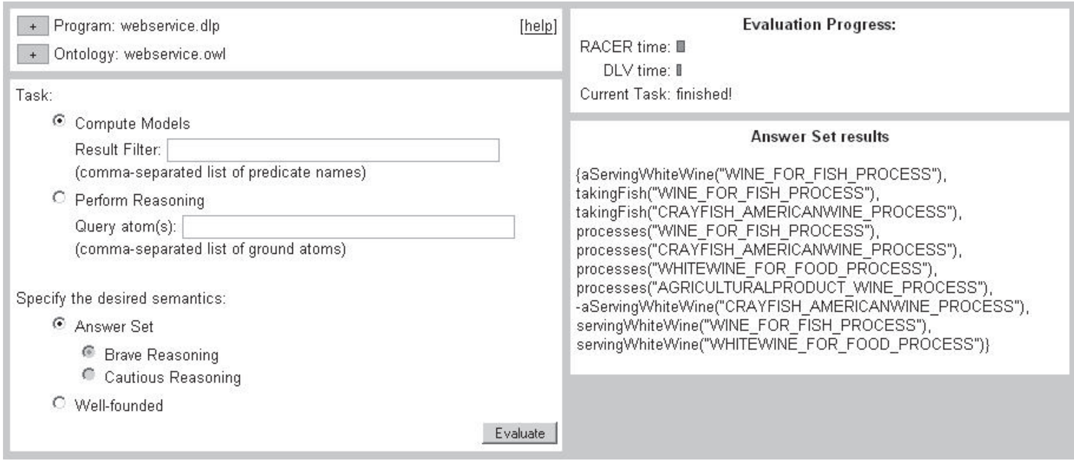

Figure 1: NLP-DL Web evaluation prototype.

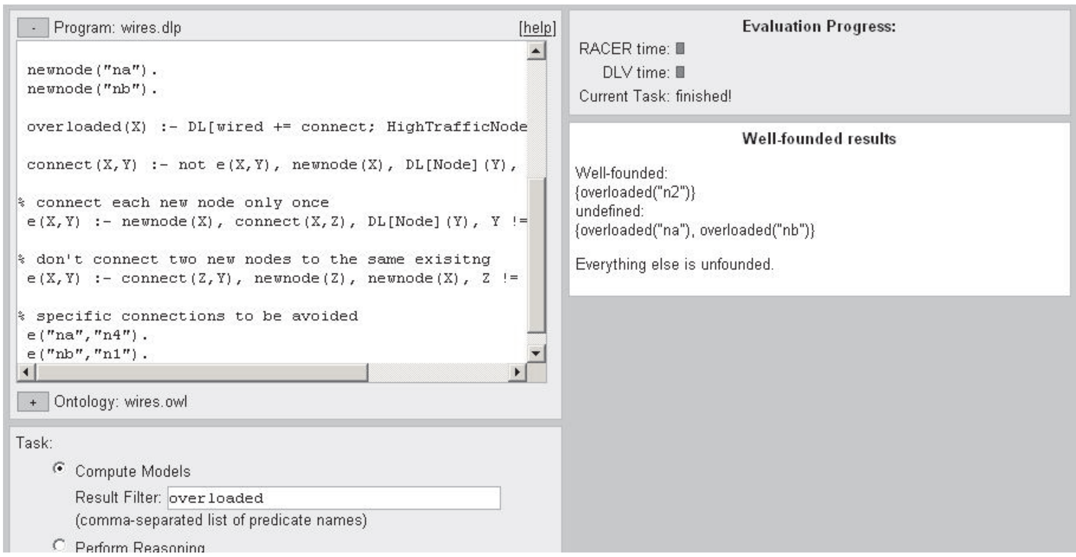

Figure 2: Editing the logic program.

The current prototype is accessible through a publicly accessible Web page, which can compute the models or, respectively, evaluate queries for a given dl-program. The rules of the latter are entered into a text field, using traditional logic-programming syntax (extended by the syntax for dl-atoms), whereas the description-logic knowledge base can either be given by a second text field or—considering the ontology to be substantially larger than the logic program—be specified by a URL.

On pushing the “Evaluate”-button, the computation procedure is started, which iteratively calls DLV and RACER. Two corresponding progress bars display the time spent by each of these external applications. Below these bars, a status message informs about the currently executed subtask and additionally indicates that the system is in a running state. If the model generation task is selected, the found answer set(s) resp. the well-founded model are shown upon termination; if the query-evaluation task is selected, the corresponding query answer is given. If query answering under the answer-set semantics is chosen, one can additionally decide between brave and cautious reasoning.

Figure 1 shows the task selection part of the Web page together with the answer set result for the wine Web service example. The text fields for the logic program and the ontology can be toggled on and off. Our aim was to find a concise page layout which provides room for the user input and the evaluation result simultaneously. In Figure 2, the model of another example is evaluated under the well-founded semantics. Here, the result filter specification is used in order to restrict the result output to one or more specified predicates.

In addition to the two examples mentioned here, the NLP-DL Web page, which is reachable under `http://www.kr.tuwien.ac.at/staff/roman/semweblp/`, includes a selection of further dl-programs demonstrating various aspects of our formalism. They can be loaded into the prototype and altered arbitrarily.