Model-Based Deduction for Knowledge Representation

[Position Paper]

Peter Baumgartner
Universität Koblenz-Landau
Fachbereich Informatik
Koblenz, Germany
peter@uni-koblenz.de

Ulrich Furbach
Universität Koblenz-Landau
Fachbereich Informatik
Koblenz, Germany
uli@uni-koblenz.de

Keywords

Knowledge Representation, Terminological Languages, Model Computation, Theorem Proving, Logic Programming

In this note, we argue that model-based automated deduction techniques are very well suited for knowledge representation purposes. This is an argument leaving the mainstream of knowledge representation research, which currently has its focus on the development of description logic systems. We want to point out that we consider this direction of research extremely successful: it led to a deep inside of computational properties of decidable subclasses of first order reasoning; it made clear some interesting links to nonclassical logics and moreover, description logic systems are nowadays outperforming most modal logic theorem provers. Despite of those successful developments we find two reasons which motivated our attempt to use first order theorem provers for knowledge representation purposes instead of dedicated description logic systems :

- Even the key researchers in the field of description logics are stating some severe deficiencies of their systems (e.g. [4]: In realistic applications it is clear, that the query language of description logic systems is not powerful enough; there is no possibility to deal with nonmonotonic reasoning and there are no efficient means to handle large A-Boxes. Only recently the community investigates seriously the extension of description logic systems towards A-Box reasoning, which is all but trivial [5].
- Currently we are using knowledge representation in two application projects. One is an EU-project, TRIAL-SOLUTION (www.trial-solution.de), where a system for the use of personalized electronic books is developed. This is done in Slicing Book Technology (www.slicing-infotech.de), such that knowledge which is represented in several books has to be combined by a knowledge representation system, in order to answer the queries given by a user, i.e. a reader. Because the chunks of knowledge are rather small, it is clear that we have to handle ten thousands of such chunks, which have to be represented as objects in an A-Box - Hence we have to handle very big A-Boxes. In a second application we have to model a certain domain of a major German bank in order to support decision making. Both applications demand language and query constructs which are not available in current description logic systems.

Semantic Web Workshop 2002 Hawaii, USA

It is obvious that most of the arguments against and deficiencies of description logic hold for the application for reasoning within the semantic web as well.

Our approach is oriented at the paradigm of logic programming and model-based theorem proving. Instead of starting with a small and efficient kernel language like ALC, which is stepwisely extended towards applicability, we start with the general language of first order logic and then, we identify sub-languages that are decidable. The largest subclass that we can handle is that of the Schönfinkel-Bernays fragment extended by a default-negation principle. The user of our system can decide to stay within this class or whether she wants to use some language construct which leave this class. It is important to note, that we offer our kernel language with a syntax which is very similar to languages like OIL.

Our approach can be summarized by the equation

$$\mathsf{KRHYPER} = \mathsf{Kernel} + \mathsf{Logic\ Programming}$$

where

- Kernel is an OIL-like language which is augmented by some additional constructs, like non-monotonic negation and second order extension.
- Logic Programming denotes rules, axioms, constraints and concrete domains from logic programming.
- KRHYPER is the (extended) first order predicate logic which can be processed by our model generating tableau theorem prover Hyper ([2, 3]).

The Kernel Language

OIL class definitions, e.g.

```
class-def defined carnivore
        subclass-of animal
```

```
slot-constraint eats
    value-type animal
```

have a similar concrete syntax in our kernel language. Most parts of OIL are covered, in particular all kinds of class definitions, inverse roles, transitive roles etc. The constructs from the Kernel language are translated to our logic programming language following standard schemes.

Beyond this, we are able to handle the following points which are mentioned explicitly as missing in [4]:

Rules/Axioms:

In addition to constructs in the syntax of the knowledge representation language we can use arbitrary formulae as constraints, rules or axioms. For instance, we can state in the rule part

```
dangerous(X) :- carnivore(X), larger_than(30,X).
```

to express sufficient conditions for being `dangerous`. The `larger_than` relation would be defined by the user as a unary Prolog-predicate.

Using Instances in Class Definitions

Although it is well known (cf. [1]) that reasoning with domain instances certainly leads to EXPTIME-algorithms, it is very clear that exactly this is mandatory in practical applications. For instance, the previous example could also be supplied as

```
dangerous <= carnivore & larger_than(30).
```

in the terminological part.

Default Reasoning

In our system we included a closed world assumption, such that we can use default negation principle "`not` " besides the classical one. For this negation we implemented a well-founded model semantics. Default negation may be used both in the rule part and in the terminological part. For the latter case, the previous example might more appropriately be written as

```
dangerous <= carnivore & not smaller_than(30).
```

Switching Back and Forth

One may switch back and forth between the terminological part and the rule part, by keeping in mind that concepts translate into unary predicates, and that roles translate into binary predicates.

A-Boxes

Concrete instances of concepts (roles) are handled via unary (binary) predicates. This is a very natural and well-understood method for model generation procedures. For instance, from

```
dangerous <= carnivore & not smaller_than(30).
```

and the A-Box consisting solely of

```
carnivore(leo).
```

the model generation prover will derive `dangerous(leo)`. Unlike as in other systems, no grounding in a preprocessing phase takes place, and the system is capable of computing with A-Boxes consisting of tens of thousands of objects.

Limited Second-Order Expressivity

Very often it is necessary to treat statements of the language as objects and to apply procedures for some kind of evaluation to them. This can be done in our context by meta-language constructs à la Prolog. For instance, via `concept_instance(Concept,Instance)` one has access to the `Concept` names where `Instance` is an instance of. For example,

```
all_dangerous(X) :-
       call(findall(Z,
           (dangerous(Y),
                 concept_instance(Z,Y)),X)).
```

describes as a Prolog-list all the concepts that have an instance of the `dangerous` concept.

Altogether we want to point out that our approach to use an existing deduction system and to carefully divide its language with respect to complexity and decidability issues leads to a knowledge representation system that allows to handle practical applications.

1. REFERENCES

[1] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In *CSL*, pages 307–321, 1999.

[2] P. Baumgartner. Hyper Tableaux — The Next Generation. In H. de Swaart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 60–76. Springer, 1998.

[3] J. Dix, U. Furbach, and I. Niemelä. Nonmonotonic Reasoning: Towards Efficient Calculi and Implementations. In A. Voronkov and A. Robinson, editors, *Handbook of Automated Reasoning*, pages 1121–1234. Elsevier-Science-Press, 2001.

[4] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In *Proceedings of the European Knowledge Acquisition Conference (EKAW-2000)*, Lecture Notes In Artificial Intelligence. Springer-Verlag, 2000.

[5] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic SHIQ. In D. MacAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, Germany, 2000. Springer Verlag.