Model-Based Deduction for Knowledge Representation

Peter Baumgartner Ulrich Furbach Bernd Thomas

Universität Koblenz-Landau
Institut für Informatik
e-mail: {peter,uli,bthomas}@uni-koblenz.de

Abstract. This paper argues that model-based automated deduction techniques are very well suited for knowledge representation purposes. As a motivation we describe three applications and we discuss the requirements for a knowledge representation formalism. The presentation of our KR-system, KRHYPER, is centered aoround the equation

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

where we put particular emphasis on the combination of rules and axioms, the non-monotonic features and the handling of realisticly large ABoxes

1 Introduction

In this paper, 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 insight into 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 these successful developments we find two reasons which motivate our approach to use a first-order theorem prover for knowledge representation purposes instead of dedicated description logic systems. First, even the key researchers in the field of description logics are stating some severe deficiencies of their systems (e.g. [6]): research into description logics focused on algorithms for investigating properties of the terminologies, and it is clear that for realistic applications the query language of description logic systems is not powerful enough. Only recently the community investigates seriously the extension of description logic systems towards ABox and query answering, which is all but trivial [8, 7]. Second, there is no means to deal with non-monotonic reasoning or to add to the terminological part a complementary knowledge representation scheme, say by a normal logic program.

Our focus is on the development of such a language and system that combines a terminological language with (stratified) normal logic programs. The

specifications may be "mixed", in the sense that concepts and roles defined in the terminological part may be used or further extended/constrained in the logic program part. Regarding computation with such specifications, we follow a model-computation paradigm. That is, a bottom-up procedure is employed that computes a minimal model of the whole specification.

Our interest is in bottom-up model computation, and not in a top-down answer substitution computation. This is motivated by our applications, where a model is indeed the required result from the application point of view.

That such language combinations and performing model computation makes a lot of sense has been demonstrated e.g. within the On2Broker project [4]. Its input languages are the "Semantic Web" terminological language RDF/RDFS[1] and a rule-language similar to normal logic programs (in fact, On2Broker can deal with a restricted form of Frame-Logic [9]). However, RDF/RDFS is a rather weak terminological language, and we are targetting at much more expressive terminological languages like OIL [6] (which includes e.g. transitive roles).

Our approach is highly motivated by knowledge based systems from the areas of computer based teaching and information agents. A short description of these is given in the following Section 2. 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.

2 Motivation

Our approach of model based deduction for knowledge representation is strongly motivated by several practical applications in the domain of *Computer Based Teaching*, *Decision Support* and *Mobile Information Systems*. While building applications in these three domains it became very apparent that existing description logic systems lack a certain degree both of expressiveness due to the taxonomical modeling and their query language.

The following presents a broad overview of our applications using model based deduction for knowledge representation.

Computer Based Teaching. Currently we are using knowledge representation in two Computer Based Teaching 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. It builds on the Slicing Book Technology (www.slicing-infotech.de), where documents are decomposed ("sliced") into elementary units for the purpose of later, selective re-assembly. More precisely, the knowledge which is represented in several books is interconnected by metadata annotations (keywords, structural relations and relations concerning the contents) and combined by our knowledge representation system. The particular application task then is to assemble a new document, based on the user's current interest and knowledge.

[1] `http://www.w3.org/RDF/`

Technically, this task maps naturally to a model-computation task, where the formalization consists of facts for the Metadata (which can be seen as an ABox) and a normal logic program. We found features like default negation to be essential, which are beyond the scope of pure taxonomical modeling (TBox languages).

Decision Support System. Nowadays nearly all major companies have certain departments for collecting and analyzing information from newspapers, TV-stations and other common mass media sources. This information is used to decide whether a new trend is growing or new critical problems will arise in the near future. Intentionally, departments like these try to detect reputational risks relevant for their business. Obviously such a detection or at least the classification of a set of news belonging to one issue requires a lot of common knowledge about various domains. Even if the clustering of certain news articles into categories is available, the main problem is still unsolved, namely to uncover and to derive dependencies among the issues. We have modeled such a domain[2] and developed a system for a major German bank in order to support their decision making. Once again we faced the problem that existing description logic systems are not capable of providing language and query constructs demanded by such an application.

Mobile Information Agent: MIA. MIA[3] is a city information system for mobile users. The key concept of MIA is that the user can define arbitrary search profiles consisting of a topic, for example restaurants and constraints like `only italian`. MIA gathers information from web pages related to these search profiles and city and extracts location addresses from these web pages relevant to the users request. MIA can be used by a stationary web browser user or while travelling with a PDA plus a GPS (global positiong system) or thirdly with a WAP capable mobile device. MIA also offers location awareness, that is whenever the mobile user moves his position, MIA recognizes the location change and updates the current search tasks.

The strength of MIA, namely to allow a user to define freely what to search for, makes it even harder to ensure correct results. Two basic tasks have to be carried out by MIA: a) the search for relevant web pages b) the extraction of addresses. Step a) is done by an online spidering algorithm and offline learned neural network based classifiers to determine if page candidates contain addresses. The online spidering algorithm simulates human like web *surfing*. Therefore some extended knowledge about related terms of the keywords from the search profiles is needed. This is exactly the point where we use model based deduction for knowledge representation. Step b) the extraction of addresses from never before seen web pages with varying structure of addresses necessitates the use of machine learning based techniques. These techniques [11] are used to learn various extraction procedures offline and online. Nevertheless the current state of the art in machine learning based information extraction still requires some

[2] togehter with wizAI.com

post cleaning of extractions, because the correctness of extractions is not and probably will never be 100%. But to use fully automated extraction techniques in a autonomous information system like MIA also requires an automated evaluation of the extracted results. Once again we use the keywords from the search profiles in combination with model based deduction to derive additional knowledge (terms). The evaluation process than checks if one of these derived terms is contained in the extractions. It turned out that this evaluation procedure works suprisingly well without any use of linguistical methods.

3 The Core Engine: KRHYPER

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 Bernays-Schönfinkel 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 features (reification).
- 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 an extended version of our model generating tableau theorem prover Hyper ([2, 5]).

KRHYPER is currently restricted to stratified disjunctive logic programs, and is sound wrt. the possible model semantics [10].

4 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 [6]:

Rules and 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[3]

```
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.

ABoxes. 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).
```

[3] Notice that description logic languages such as OIL usually permit concept definitions via equivalences (`<=>` in our syntax) or via necessary conditions (`=>` in our syntax). However, we start off with a concrete ABox that is assumed to implicitly represent a model of some TBox, and that can be extended to an explicitly represented model by using sufficient conditions, as shown.

and the ABox 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 ABoxes 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.

5 Examples

Throughout this section we use a slightly modified example taken from the ontology used within the agents of the information system MIA (see section 2). Three major advantages of our knowledge representation and reasoning system are demonstrated, namely: the use of three negation operators (one classical and two nonmonotonic ones), the use of rules to enrich our ontology and examples how to retrieve information by our expressive query language. As a basis for demonstration the partial snapshots of the MIA ontology is used (Figure 5 and 5).

```
place_to_eat <= location & (? has_name : name).
restaurant <= place_to_eat & (? has_price : price)).
fastfood <= place_to_eat.
food <= solid.
meal <= food.
name <= abstract.
nationality <= abstract.
price <= abstract.
```

Fig. 1. MIA's TBox (part)

Using Rules. Assume a user wants to query the MIA system to find addresses of restaurants offering cheap meals. Therefore she passes only the keywords `hamburger` and `restaurant` to the system. Based on these keywords the information system has to a) search for relevant online resources b) evaluate the extraction results. The use of these two keywords is obviously not enough to cover the users original intention while searching for web pages or to check if the extracted information is correct. Therefore a senseful extension is to include additional knowledge about these two keywords. So the key idea is to define a set of rules which is used to retrieve related terms from the ontology which are semantically connected to these keywords. Referring to our example, it is clear that there are many restaurants that might not have the keyword hamburger in its name, but nevertheless should be presented as search result if we know that they sell such meals.

```
restaurant('trattoria').
has_price('trattoria','high').
has_name('trattoria','san_marino').
offers('trattoria','spaghetti').
of_nationality('spaghetti','italienisch').
of_nationality('san_marino','italienisch').
name('san_marino').
meal('spaghetti').
fastfood('pommes bude').
has_price('pommes bude','low').
has_name('pommes bude','bei fritz').
of_nationality('bei fritz','deutsch').
fastfood('drive_in').
of_nationality('hamburger','amerikanisch').
has_name('drive_in','mc_donalds').
of_nationality('mc_donalds','amerikanisch').
offers('drive_in','hamburger').
name('mc_donalds').
meal('hamburger').
```

Fig. 2. MIA's ABox (part)

The problem is, we do not know in advance if these keywords are instances of concepts, concept names or perhaps names of roles. Thus we have to find a way to define a set of rules which allows us to state this retrieval task as some sort of second-order rule. For the semantics of `concept_instance` see Section 4. To retrieve related terms independently of knowing either if a keyword is a concept or an instance we define the following two rules `related_term` that return a semantic related term for later use in the context of extended web search or extraction evaluation.

```
related_term(KeyWord,Related) :-
        concept_instance(KeyWord,Related)
        ;
        concept_instance(Related,KeyWord).
```

If we also want to derive the parent concepts related to the keywords we add the following rules making use of the meta logical predicate `is_a`:

```
related_term(KeyWord,Related) :-
       is_a(KeyWord,Related).

related_term(KeyWord,Related) :-
        concept_instance(Concept,KeyWord),
        is_a(Concept,Related).
```

To derive which other terms are related via role definitions we extend the rule set by two additional rules making use of the meta logical predicates `role_instance` and `has_a`, which provide information about the defined roles.

```
related_term(KeyWord,Related) :-
        concept_instance(Concept,KeyWord),
        (
         has_a(Concept,RoleName,ConceptFiller),
         role_instance(RoleName,KeyWord,Related)
         ;
         has_a(ConceptFiller,RoleName,Concept),
         role_instance(RoleName,Related,KeyWord)
        ).
```

Note that meta logical predicates are also computed transitively and follow an inheritance principle. This means asking for the parent concept `X` of a concept `C` enumerates all concept ancestors of the concept `C`. This also holds for the meta logical predicate `has_a(C,R,CF)` that also computes all inherited roles from the ancestor concepts of `C`.

Finally we are interested in a list of all related terms for a given keyword, therefore the final rule used by MIA is:

```
all_related_terms(X,Y) :- call(findall(Z,related_term(X,Z),Y)).
```

And a example query `all_related_terms('hamburger',Y)` will return the list: `[meal, food, solid, amerikanisch, drive_in]`

Nonmonotonic Negation. In the context of information systems in particular and taxonomical reasoning in general nonmontonic negation provides a strong help for three reasons: 1) The taxonomy itself can be kept small in terms of numbers of definitions, because negative knowledge do not has to be modeled explicitly. 2) It provides convenient methods to define concepts. 3) It allows queries

of high practical usage. Returning to our example assume a user is interested in addresses of restaurants but not italian ones. MIA allows keyword queries like `restaurant not italienisch`. Thus our ontology should also be able to provide us with information about none italian restaurants. As mentioned before it is not necessary to model our (negative) knowledge that for example a drive-in restaurant is not an italian style restaurant explicitly, as observable if we take a look at the ABox and TBoxes in Figure 5 and 5. This is a major advantage for maintenance of ontologies. Assume the case were explicitly negative information in classical logic has to be added to the ontology. Whenever a new instance is added we would also have to update our ontology. For example in the classical case we would have to extend our ontology by the ABox entry `not of_nationality('drive_in',italienisch)` to state that a drive-in is not an italian style restaurant. This would lead to a quadratic number of additional entries in the number of nationalities and instances of restaurants if done exhaustively. But we can express this much more comfortable:

```
not_italian_restaurant <= restaurant & not_italian.
not_italian(X) :-
       has_name(X,Name),
       ∼ of_nationality(Name,'italienisch').
```

or a more compact representation of this formula:

```
not_italian_restaurant <=
   restaurant & (has_name :
              (∼ of_nationality : equal('italienisch'))).
```

Thus the set of instances defined by the concept `not_italian_restaurant` in our example is {`'drive_in', 'pommes bude'`}.

The system also supports classical negation by the use of the operator (`-`). Exchanging the nonmonotonic negation operator $\sim$ with `-` yields a clause: $not\_italian(X) \vee \neg has\_name(X, Name) \vee of\_nationality(Name,' italienisch')$. This clause union the example ontology defines a empty concept (no instances).

A more generic formulation for excluding certain instances of a given concept `C` matching a constraint `N` can be formulated by the following rule:

```
neg_constraint(C,N,CI) :-
        concept_instance(C,CI),
        ∼ exists(RoleName,role_instance(RoleName,CI,N)).
```

This rule makes use of the second nonmontonic negation. By using the existential quantifier (`exists(Vars, LiteralConjunction)`) in combination with our nonmonotonic negation operator ($\sim$) we are able to express, that for all role instances `RoleName` it should hold that the instance `CI` of concept `C` is not in relation to an instance `N`. Such a rule is used by MIA to obtain instances of a

concept which are not allowed to have a specific role filler in one of its roles. For example the query "*Give me all restaurants, but those having something to do with deutsch.*" `neg_constraint(restaurant, 'deutsch', X)` computes answers for `X` with {`X/'trattoria'`, `X/'drive_in'`}.

Expressive Query Language. So far we have demonstrated how to define rules to perform some sort of second-order reasoning about existing roles and concepts. This is all motivated to retrieve knowledge related to an arbitrary keyword to extend the search and extraction capabilities of the MIA system. Though these rules itself can be seen as definitions of queries it should be clear, that we have at hand a very general query techniques offered by our system.

6 Conclusion

We have demonstrated that it is very well possible to use a first order theorem prover toghether with some necessary extensions as a knowledge representation system. In particular the model generation based approach offers some advantages in the application areas we are working in. 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, with large amounts of data.

References

1. C. Areces, P. Blackburn, and M. Marx. A roadmap of the complexity of hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, number 1683 in LNCS, pages 307–321. Springer, 1999. Proceedings of the 8th Annual Conference of the EACSL, Madrid, September 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. G. Beuster, B. Thomas, and C. Wolff. MIA - A Ubiquitous Multi-Agent Web Information System. In *Proceedings of International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA'2000)*, December 2000.
4. S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. M. et al., editor, *Semantic Issues in Multimedia Systems. Proceedings of DS-8*, pages 351–369. Kluwer Academic Publishers, 1999.
5. 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.
6. 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.

7. 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.
8. I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *AAAI'2000, Proc. 17th (U.S.) National Conference on Artificial Intelligence*, pages 399–404. AAAI Press/The MIT Press, 2000.
9. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, May 1995.
10. C. Sakama. Possible Model Semantics for Disjunctive Databases. In W. Kim, J.-M. Nicholas, and S. Nishio, editors, *Proceedings First International Conference on Deductive and Object-Oriented Databases (DOOD-89)*, pages 337–351. Elsevier Science Publishers B.V. (North–Holland) Amsterdam, 1990.
11. B. Thomas. Anti-Unification Based Learning of T-Wrappers for Information Extraction. In *Workshop on Machine Learning for Information Extraction*. Amercian Association of Artificial Intelligence, July 1999. preceeding Sixteenth National American Conference on Artifical Intelligence (AAAI-99).