Reasoning with a Domain Model

Steffen Leo Hansen
København

Abstract

A domain model is a knowledge base containing both domain specific and world knowledge. You may take the domain model to be both a universe of interest and a universe of problems. As a universe of interest the model contains all the information relevant and necessary for the intended use of the model as a store of information, a knowledge base. As a universe of problems the model represents a problem space and the relevant and necessary inferential tools needed by the model for the intended use as a problem-solving mechanism. Problem solving, in this case, means finding answers to queries about domain-specific knowledge. In this paper we shall discuss some fundamental problems related to the construction and use of a domain model called FRAME_WORLD.

1 Introduction

The domain model presented in this paper is thought of as a module in a knowledge system using a natural language interface to retrieve information in a database. As a module of the overall system the domain model serves the purpose of evaluating user queries with respect to domain-specific knowledge and that of generating appropriate arguments for subsequent SQL commands.

The domain-specific knowledge of the model comprises facts about domain-specific entities, their properties and possible relations between these entities, whereas world knowledge comprises information not represented in the domain but necessary for the model as a problem solver, e.g heuristics, general rules about causal or spatial relations and the like. The relevant rules and facts are used by an inference machine, not only to state information already explicitly at hand, but also to support the system in making implicit domain-specific knowledge explicit.

The knowledge representation schemes used in the domain model presented are a semantic network, frames, so-called model predicates and heuristics. In the following sections we shall present and discuss the implementation and intended use of FRAME_WORLD, first of all problems of reasoning with inferential structures given by virtue of a specific representation scheme.

2 The domain model

The knowledge in the domain model and the structure of the model depends entirely on the purpose it serves. As already mentioned, the model is thought of as a kind of filter, a means of controlling and checking the knowledge represented in the queries posed to the system by the user and either reject the query as a senseless one or compute and generate one or more arguments to be used in an SQL command to retrieve the required information.

The basis for building and constructing the domain model, therefore, is the set of possible and allowed queries to the system, like for instance: who is the colleague of X, how many people are employed in the sales department, or how much is the salary of X?

To answer questions of this sort you have to have access to both domain-specific and world knowledge. To know whether X and Y are colleagues, you have to have some rule telling you what it means to be colleagues and some means of checking if X and Y in our domain actually do fit this definition. If this is not the case, we do not want the system to react by simply answering 'No', but an output like: 'X is a customer, and Y is an employee'.

To this purpose the domain model needs information about entities and relations in the domain and in the world outside the domain as well as some kind of machinery that uses this knowledge for information retrieval and query answering.

Entities and relations between entities inside and outside the domain are represented as a network of nodes and links. The nodes in the net are conceptual entities, the knowledge primitives of the model. The links in the net either relate concepts as conceptual entities to each other or concepts as arguments of a semantic predicate to each other. The former kind of links are called conceptual links, the latter, the relational links, are called role relations.

The description of a node comprises both the set of incoming and outgoing links as structural information about the concept as well as the set of conceptual features characterising the specific concept in question. This description is implemented as a frame. The role relations, too, are mapped into frames such that for each concept and for each role relation in the net there will be a frame with the same name as a description of that particular knowledge unit.

3 Representation schemes

3.1 The network

Using a network for knowledge representation in a domain model seems obvious. Knowledge pictured as a network makes it possible to represent a conceptual hierarchy as a nice structure of nodes and links representing all available information immediately ready for use. All you need is the right algorithm extracting the information or transfering information from more to less general nodes of concepts. It seems to reduce knowledge retrieval to simply finding the right node or nodes and the right path connecting two or more nodes with each other.

It is, however, not as simple as that. Reasoning with a network presupposes a well-defined syntax and semantics of the net as discussed and emphasized in several papers (e.g. Woods 1987 & 1990, Thomasson/Touretzky 1991).

The idea of using networks as a representation scheme is that of making information attached to some node X accessible for other nodes connected to X. This property of a network is the fundamental principle of *inheritance* and *path-based reasoning*, and probably the most important reason for the popularity of this way of organizing knowledge and using a network as an inferential tool.

Inheritance means that information kept in a node X is inherited by a node Y if Y is connected to X. Path-based reasoning means infering conclusions by way of finding a correct path through the net, in most cases simply by computing the transitive closure of a set of links in the net (Thomason/Touretzky 1991:239). Let us illustrate these principles using a fragment of the domain net.

In this fragment (fig. 1) we have two different kinds of conceptual links labelled *ako* and *apo*, *a kind of* and *a part of* , and a relational link labelled *works_in* stating that an employee works in a department. Both the *ako* and *apo* relations are transitive relations, and without any further restrictions one might infer that a subordinate is a kind of legal person.

This conclusion is derived by simply computing the transitive closure of the links involved, but it not a valid one because it is based on two different and incompatible concepts: the concept *firm* as a subconcept of the superconcept *legal person*, a generic concept defined by a set of conceptual features, and the concept *firm* defined by the set of parts constituting it as a whole, one of which is a department.

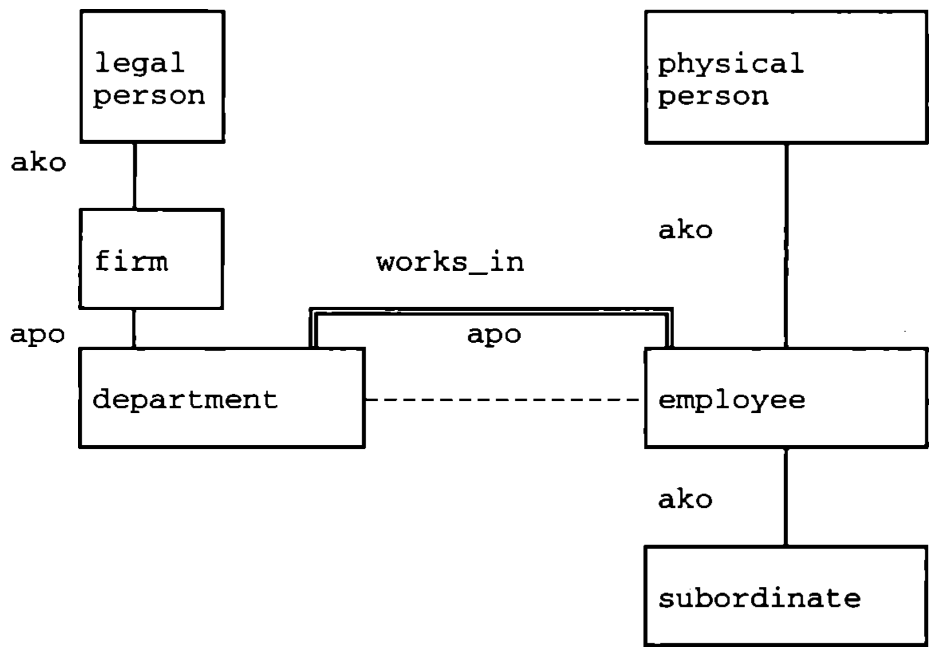

FIG. 1

To avoid conclusions like the one just presented we have to define both the syntax and the semantics of the net. The net in FRAME_WORLD consists of the following components:

(1) A set of nodes F = $\{C_{f1},...,C_{fn}\}$, generic concepts defined by a set of conceptual features,

(2) A set of nodes P = $\{C_{p1},...,C_{pn}\}$, part-whole concepts defined by a set of parts,

(3) A link type: L_{ako}, labelled 'ako',

(4) A link type: L_{apo}, labelled 'apo', and

(5) A link type: L_{role}, labelled with the name of the role.

A well-formed link in the net is a triple of one of the following types:

(6) $<L_{ako},C_{fi},C_{fj}>$

(7) $<L_{apo},C_{pk},C_{pl}>$

(8) $<L_{apo},C_{fm},C_{pn}>$

A well-formed path in the net is a structure of well-formed links. The interpretation of a well-formed link goes as follows:

(9) $L_{ako}(X) = Y: X < Y$,
X is a subconcept of the superconcept Y,

(10) Lapo(X) = Y: X ¤ Y,
X is a part of Y.

Using these definitions we can reject the conclusion: a *subordinate* is a kind of *legal person* because the final link of the path: $*\langle L_{ako}, C_p, C_f \rangle$, the *firm* being a kind of *legal person* is not a wellformed link.

It is easy to see now how the definition of a well-formed link and of a well-formed path at the same time defines the inferential structure of the net as a sequence of well-formed links. The syntax of a well-formed link also defines the syntax of a well-formed query, and the interpretation of a well-formed query is the same as that of a well-formed link.

The link type L_{role} is not part of the inferential structure in the net. This link type is part of the definition of concepts and a means of associating concepts with thematic roles like

(11) $L_{deal\_with}(X)$ = Y: deal_with(X:actor,Y:locus)

3.2 Frames

The network, as demonstrated in the previous section, is a knowledge base mapping a conceptual hierarchy into nodes representing conceptual entities and links representing conceptual relations. These nodes and links are the knowledge primitives in the domain model. In addition, the network also keeps information about role relations associating concepts as arguments of a semantic predicate with thematic roles.

The description of the nodes and the role relations as objects of information is placed in the frames in the model. A structural description of a node comprises all incoming and outgoing labelled links in the traditional slot:filler structure, using the label of a link as slot and the value of a link as filler. The description of a generic concept, further, comprises the conceptual features defining the concept in a slot labelled *attributes.*

For each concept and for each role relation there will be a frame describing the entity in question. Role relations as knowledge objects are treated in the same way as conceptual entities, i.e. as structured objects of a taxonomic hierarchy. Based on the syntax adopted by the project all the frames in the domain model are represented as Prolog terms like for instance:

```
frame( employee,
        [ ako-[ val physical_person],
          apo-[ val department],
          role-[ val work]
                                  ] ).
```

The concept *employee* is described as a kind of *physical person*, as a part of a *department* and as a valid argument in the role relation *work*.

```
frame( deal_with,
        [ ako-[ val process],
          roles-[ calculate deal_with(X,Y) ]
                                            ] ).
```

The role relation *deal_with* is described as a kind of *process* with a role structure to be computed by the procedure *calculate*. The possible values of X and Y are computed using the so called model predicates.

3.3 Model predicates

Model predicates were introduced by (Henriksen/Haagensen 1991) as a means of checking the validity of types of arguments. Thus the interpretation of the model predicate:

```
deal_with(FIRM,CUSTOMER)
```

defines the valid arguments of the semantic predicate *deal_with* to be of the type FIRM and CUSTOMER.

In FRAME_WORLD we have extended the function of model predicates to also associating types of arguments with thematic roles. In our domain model we have the following three instances of *handle_med* (eng. deal_with):

```
handle_med(actor:firma,locus:kunde)
handle_med(actor:firma,theme:vare)
handle_med(actor:kunde,locus:firma)
```

Instead of having a frame for each reading of the predicate the procedure *calculate* will compute the relevant role structure. The actual use and function of the model predicates will be demonstrated in section 4.4.

3.4 Rules

The core of the domain model as a reasoning system comprises the network and the frames. In addition, the model may use both the model predicates and a set of domain-independent rules as part of an inference procedure. The rules, representing general world knowledge, play a very important role in making implicit domain-specific knowledge explicit defining *where to look* and *what to look for* in the knowledge base.

For the present only three rules have been implemented defining the concepts *superior* and *colleague* and the role relation *an_employee_of.* These rules, however, illustrate the need for and use of world knowledge implemented as rules.

3.5 The inference machinery

The inference machinery of the model is a set of Prolog procedures. The strategy implemented is based on the principle of inheritance and path-based reasoning using build-in facilities of Prolog. The basic operation of the machinery is that of applying the interpretation of a link as a function to a node yielding as value another node. This is not the place, however, to go into details with the inference machinery. Let us, instead, take a look at how the domain model actually may be used and how it functions as a knowledge filter and generator in a question-answering system.

4 Reasoning with the domain model

In this section we shall focus on the intended use of the domain model. For the present, we can only show how to use the network, the frames, the model predicates and the rules as part of a reasoning system. This may, however, give you an idea of the intended performance of the model as a whole

4.1 The frames

The frame structure is utilized in two ways: (a) either to instantiate variables used by the inference machinery with values found in a frame, or (b) to find one or more frames matching a description:

(a) ?- frame(leder,**Slots**).

Slots = [ako-[val ansat], role-[val lede]]

```
?- frame(lede,Slots).

        Slots = [ako-[val arbejde],
                 roles-[calc lede(_8210,_8211) ]]
```

(b)

```
?- frame(Name,[ako-Ako,role-Role]).

        Name = leder
        Ako  = [val ansat]
        Role = [val lede]
        ...
```

4.2 The network

As you have probably already noticed, the structure of a frame as a description of a node is an encoded fragment of the network. The inference machinery uses this property of a frame in path-based reasoning. Actually, there is no network explicitly at hand in the domain model, but using the structure of the frames the inference machinery may generate one or more sub-nets computing the transitive closure of a link in the net:

```
? - get_frame(Name,ako-Ako).

        Name = person
        Ako = entity

        Name = physical person
        Ako = person

        Name = employee
        Ako = physical person
        ...

?- get_frame(Name,apo-Apo).

        Name = department
        Apo = firm

        Name = employee
        Apo = department

        Name = manager
        Apo = department
        ...
```

Generating hierarchies of this kind may at a later time be used as an instrument to check whether some inferred type value, say, *secretary*, is subsumed by some other type value, *employee*, and, consequently, a valid argument of the semantic predicate *work* as in: *work(secretary,department)*.

4.3 Inheritance

Inheritance normally means inheriting properties. This is also true of the domain model although inheritance in this case rather means structure copying (Winston 1974:263). The concept *physical person* in the net is defined by the features: *Navn*, *Adresse* and *CPR*. These features are representend in the corresponding frame in a slot labelled attributes and may be inherited by all subsumed concepts like

```
? - get_frame(sekretær,attr-Attr).

        Attr = [navn:NAVN,adresse:ADRESSE,cpr:CPR]
```

This is also true of role relations as features defining a generic concept:

```
? - get_frame(sekretær,[role-Role,roles-Roles]).

        Role = arbejde
        Roles = arbejde(actor:ansat,locus:firma)
```

In this case the role relation and the role structure is inherited from the superconcept *employee*.

4.4 Model predicates

The model predicates are potential inferential tools, tools to support the inference machinery as a means of controlling types and values instantiated by the inference machinery. These predicates may be used in three different ways:

(1) the procedure *calculate* called in a frame computes all possible role structures of a specific predicate:

```
?- get_frame(handle_med,roles-RoleStr).

        handle_med(actor:firma,locus:kunde)
        handle_med(actor:firma,theme:vare)
        handle_med(actor:kunde,locus:firma)
```

(2) given a specific conceptual entity as argument *calculate* computes the corresponding role structure:

```
? - get_frame(kunde,roles-RoleStr).

        handle_med(actor:firma,locus:kunde)
        handle_med(actor:kunde,locus:firma)
```

(3) the procedure *calculate* computes the role structure inherited from a subsuming argument type:

```
? - get_frame(sekretær,roles-Roles).

        arbejde(actor:ansat,locus:firma)
```

4.5 The rules

The rules in the domain model are implemented as Prolog rules. The definition of two colleagues, X and Y, presupposes that they are both in the same department and that they are both at the same level of employment, that is either subordinates or managers of a kind. The latter condition means that the persons in question as nodes in the net has to be either sister nodes or subsumed by the same superconcept, the former condition is implemented using a shared variable, *AFD*, in the call of the knowledge base. A simplified version of the actual rule, then, is:

```
kollega(X,Y):-
     get_frame(STX,ako-Ako),
     get_frame(STY,ako-Ako),
     table(X,STX,AFD),
     table(Y,STY,AFD),
     X \== Y.
```

Using rules like this one is one way of incorporating domain-independent knowledge in the domain model. The user of the system is not supposed to have any knowledge about the data structures in the knowledge base. If you don't want to tune the knowledge base to some specific application or to be usable for only a limited amount of users you will have to supply the domain model with several rules like the one just presented, changing general knowledge about the domain into domain-specific knowledge and making implicit knowledge explicit.

5 Summary

In this paper we have presented some principles and methods used to map domain-specific and world knowledge into a domain model called FRAME_WORLD. We also showed that, having access to knowledge about conceptual entities and relationships in the domain in question, this model may be used as part of a reasoning mechanism to both check and generate types and values as valid arguments of semantic predicates . The aim of using such a domain model is to facilitate the dialogue between the end-user and a knowledge database. FRAME_WORLD is still just a toy model, but yet a useful tool to investigate and test principles and methods underlying the construction and use of a domain model.

References

Henriksen, Lina and Lene Haagensen. 1991. *Speciale om domænemodellering*. Institut for Datalingvistik, HHK.

Thomason, Richmond and David S. Touretzky. 1991. *Inheritance Theory and Networks with Roles* In: Sowa (ed.): *Principles of Semantic Networks*, Kaufmann, pp. 231–267.

Østerby, Tom. 1992. *Kunstig intelligens, metoder og systemer*. Polyteknisk Forlag.

Winston, Patrick H. 1974. *Artificial Intelligence*, Addison-Wesley.

Woods, W.A. 1987. *What's in a link? Foundations for Semantic Networks*, in: Brachmann/Levesque (eds): *Readings in Knowledge Representation*, Kaufmann, pp. 217–243.

Woods, W.A. 1991. *Understanding Subsumption and Taxonomy: A Framework for Progress*. In: Sowa (ed.): *Principles of Semantic Networks*, Kaufmann, pp. 45–95.