RULE SUBSUMPTION IN OBJECT BASES

MAURICE D. MULVENNA, MAUREEN MURPHY, JOHN G. HUGHES

Northern Ireland Knowledge Engineering Laboratory (NIKEL), Faculty of Informatics, University of Ulster at Jordanstown, Shore Road, Newtownabbey, Co. Antrim BT37 0QB UNITED KINGDOM
Tel: 044 (0) 1232 368197 Fax: 044 (0) 1232 366068 Email: nikel@ulst.ac.uk

ABSTRACT

Coincident with the arrival of methodologies specific to knowledge based systems, there has been a realisation that such knowledge based systems frequently form part of larger information systems. Therefore, a single method for building general information systems that incorporates knowledge based components is preferable. The most popular methods in use today are based upon object-oriented modelling. In this paper, the technique of incorporating knowledge, in the form of subsumed rules as first class objects into object bases is introduced. Aspects of the research are examined using a simple prototype, while other issues pertaining to the research are presented.

KEYWORDS

Knowledge Based Systems, Knowledge Engineering, Knowledge Modelling, Rule Subsumption, Object-Oriented.

INTRODUCTION

The field of Knowledge-Based Systems (KBS) has seen a number of major developments over the past decade. Firstly, in contrast to the predictions made by the early proponents of the technology, who envisioned KBS as stand-alone problem-solvers, KBS have found a more natural environment acting as a problem-solving component within the context of a larger information system. Secondly, the applicability of knowledge engineering development methods which have been engendered by the KBS community, for example KADS [1], is being questioned as more structured and globally relevant information systems development methodologies appear. Thirdly, the traditional representational
paradigm of rules is being augmented by the popular object-oriented schema. What impact will these changes have on the long-term future of KBS and how will the role of KBS development evolve?

The primary issue to be considered is that knowledge engineering must become an integral part of the full information systems development lifecycle. The traditional software engineering methodologies emphasise structured analysis and design techniques. However, despite the efforts which have been made within knowledge engineering for the promotion of more structured development techniques, these methodologies have not received widespread support within the community. This has been mainly due to their inherent complexity which is seen to overwhelm the vast majority of KBS activities which are predominantly small to medium size projects. In contrast to the software engineering field, these methodologies also lack enabling software support, which can magnify the problems for the software developer.

Software engineering techniques have themselves seen a number of major developments within the last 5 years. One of the most popular techniques which has emerged for the development of information systems is Object-Oriented Modelling (OOM) [2]. OOM originates from entity-relationship modelling [3] and seems destined to be the most popular and extensible framework for information system specification and design. It follows that the contemporary KBS, which needs to integrate fully into any envisaged system architecture, should also adopt the guidelines for OOM. It would seem that this is a complementary path for KBS development as its representational emphasis has also moved towards that of incorporating object technology.

Building KBS that incorporate knowledge as rules into object-oriented systems is changing how we develop such systems. Research is shifting focus from large, opaque rule bases towards small 'nuggets' of knowledge embedded in object-oriented systems as pattern-matching rules and rule sets. The dominant inferencing mechanism is forward; to provide state changes to the objects that contain domain knowledge.

In the following sections a rationale for embedding knowledge in object bases is given, and is termed knowledge modelling. Also, contemporary research in this area is selectively examined, and the concept of rule subsumption in object bases, which allows rules to be treated as first class objects, is introduced.

RATIONAL FOR KNOWLEDGE MODELLING

The concept of knowledge modelling, which in this context means the modelling of knowledge in object bases through the use of objects and subsumed rules, is inextricably linked with the research-driven goal of coupling knowledge bases with databases. Historically, this is best exemplified by loosely-coupled Prolog/relational database systems [4,5]. To overcome the 'impedance mismatch' of such systems, researchers intuitively realised the benefits of combining the two systems into one environment. This has only been realised, however, when the benefits of object-oriented programming became clear to database and knowledge base developers.

With respect to contemporary systems [6,7,8,9], the design of KBS using OOM techniques is problematic when attempting to combine knowledge with objects. The problems arise at every stage in the knowledge engineering lifecycle. In knowledge elicitation, should object specification proceed in
parallel with rule specification? When developing KBS, a ‘bottom-up’ approach is recommended, i.e.,
generate the rules first and then build the objects. Yet this conflicts directly with the object-oriented
approach. When implementing, where do the rules go? The root cause of the problem would seem to
be the mismatch between the popular rule-based representational formalism and the object formalism.

RULE SUBSUMPTION TECHNIQUE

Introduction

As object-oriented modelling is proving so universal, there exists a requirement for the most applicable
of KBS technologies, rules, to be incorporated into such object-oriented systems. This knowledge
modelling facility may be termed ‘rule subsumption in object bases’. Rule Subsumption in Object
Bases (RSOB) allows incorporation of rules into object-oriented systems as objects [10,11], as indicated
by the technique’s name. The rules are Composite Rule objects (CR) that are aggregations of rule-part
objects. The technique facilitates the addition of a rule based component to persistent object
management systems, usually known as Object-Oriented DataBase Management Systems (OODBMS).
In the resultant system, there is no impedance mismatch between objects and rules, as the rules are
themselves composite objects. As RSOB conforms to the object-oriented paradigm, using primitives
that correspond to emerging OO standards such as CORBA, it is transportable across many object-
oriented database systems, such as Ontos, ObjectStore, and O2. The current implementation, which is
being used to investigate the usefulness of rule subsumption in object-oriented systems, has been
developed using Kappa.

Scope

The use of rules subsumed into an object-oriented environment can best be described along various
dimensions:

1. Forward inferencing v. backward inferencing (if changed v. if needed). Rules in RSOB can be
   forward inferencing or backward inferencing. Forward inferencing rules are triggered by if changed
   monitors on objects’ attributes, and backward inferencing can be triggered by if needed monitors on
   objects’ attributes.

2. Class level v. instance level (generalisation v. exceptions). Rules can be defined to operate at the
class level, or at the instance level. Rules defined at the class level may be inherited to sub classes
and instances.

3. Knowledge modelling v schema manipulation (and constraints). Rules can be used to manipulate the
   Static Domain Knowledge objects (SDK), or can manipulate the object base’s schema. Constraint
   rules also lie along this dimension.

4. Crisp v. fuzzy. Rules which infer at SDK objects’ attribute level can be crisp or fuzzy. Crisp rules
   manipulate objects’ attributes that have simple datatypes, such as integer, list, date, and text. Fuzzy
   rules manipulate objects’ attributes that have fuzzy datatypes; for example, where a slot may have
   fuzzy adjectives such as low, very low, and quite high.
RSOB results in a persistent object storage system containing SDK objects and CR objects. The points outlined above are explored below. CR objects can be ‘fired’ in either backward inferencing or forward inferencing mode. Forward inferencing is triggered by an if changed monitor attached to an attribute of an object, while backward inferencing is triggered by a similarly attached if needed monitor. The if changed monitor is activated when the attribute’s value changes, and the if needed monitor activates when a message is sent to an object, requesting a value for the monitored attribute when none exists.

CR objects can be attached to classes, and inherited by subclasses and instances. However, CR objects which scope more than one object, i.e., that are not encapsulated within a particular class, require that the objects scoped by the rules are related. For example, if there exists a general rule: if a persons car engine is big then a persons insurance is expensive,

then the Person class must have an attribute of type Object, which holds the value Car (class). Such rules allow for the incorporation of generalisations about related objects. The corollary of these CR objects are CR objects defined at the instance level, which define exceptions.

The CR objects can be used in a knowledge modelling function, i.e., as rules that change the SDK objects’ states, in reflection of the expert decision-making process. These rules correspond to those found in traditional rule-based expert systems. In addition the CR objects can be used to enforce constraints in objects, and ensure referential integrity between objects and rules. It is assumed that the underlying OODBMS maintains referential integrity between objects and associated relations. The third use for CR objects in this dimension is to allow schema evolution in the OODBMS. Schema evolution may be limited to CR objects themselves, or may include CR and SDK objects. In the latter case, RSOB could be used as the main tool for schema evolution. This third type of CR object is activated in forward inferencing mode only, and corresponds to active triggers as found in many relational systems. Also, this type of CR object is activated at the class level only, for example, a rule may be attached to respond (fire) to an event such as the creation of an instance for a class.

CR objects, defined to activate through the use of monitors when the value of an object changes or is requested, may be viewed as crisp or fuzzy rules. The car engine rule, illustrated above, is an example of a fuzzy rule, where the attribute engine in the class car is of type fuzzy, and may contain the values: tiny, average, big, and huge. The insurance attribute of the person class is also of type fuzzy, and is defined to contain one of the following values: cheap, average, expensive, prohibitive. It is possible, through the use of fuzzy rules, to define simple rules between objects that mirror more closely real-world concepts and heuristics. In the current version, only the forward inferencing aspects have been implemented fully, at class and instance level, and these work as knowledge modelling and schema manipulation rules. Work on fuzzy rules will be published in future papers. It is recognised that the inferencing mechanism utilised currently is rudimentary, and that for the ROBS system to scale up successfully, more work needs to be carried out on the design and implementation of the inference engine, particularly with respect to inter-object rule inferences.

Rule Structure

A ROBS rule is a composite object, shown below using a OO pseudo-code syntax:
The Rule object contains collection-valued attributes: premise and conclusion, which are lists of the rule’s premises and conclusions, and are of type Premise and Conclusion, respectively (multiple conclusions are allowed with forward inferencing rules only). The first four attributes of the Premise and Conclusion objects are used to facilitate knowledge-rich rules. The compiled attribute contains a string constructed from the first four attributes, for example, of the form `object.attribute relation value`, or is used to contain the more complex constraint-handling and schema commands, for example, `create instance ( Classname )`.

Forward and backward inferencing rules, which are used to manipulate the SDK hierarchy, have a premise of the form `o.a relation value`, and a conclusion of the form `o.a assignment value`.

**Inferencing**

In the RSOB System, rules are represented as complex objects in an aggregation hierarchy with root Rule Manager class (RM) (Figure.1). This object encapsulates the inferencing process, and maintains referential integrity between rules and objects.

For the initial system, an assumption is made that rules are fired as forward inferencing rules. Thus each rule may have multiple conjunctions and disjunctions in its premise, and the conclusions result in a command to modify, add or delete the static domain knowledge in the object repository. The premise of a rule may consist of range or equivalence tests on the attributes of an object, which are stored as instances of the Premise object.
For example, in an application requiring rules to manipulate knowledge about printing presses in a factory, the following rule (rule: GetPress):

*If web width of new order is less than web width of printing press and product type of new order is 'sheetfed' then print machine of new order is given value 'm/c 104'*

would be represented as in Figure 2. Table I (below) illustrates how the rule maps onto the decomposed complex rule objects.

<table>
<thead>
<tr>
<th>Object</th>
<th>Rule Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPressC1</td>
<td>- Object</td>
</tr>
<tr>
<td></td>
<td>- Attribute</td>
</tr>
<tr>
<td></td>
<td>- Relation</td>
</tr>
<tr>
<td></td>
<td>- Value</td>
</tr>
<tr>
<td></td>
<td>- Compiled</td>
</tr>
<tr>
<td>GetPressP1</td>
<td>- Object</td>
</tr>
<tr>
<td></td>
<td>- Attribute</td>
</tr>
<tr>
<td></td>
<td>- Relation</td>
</tr>
<tr>
<td></td>
<td>- Value</td>
</tr>
<tr>
<td></td>
<td>- Compiled</td>
</tr>
<tr>
<td>GetPressP2</td>
<td>- Object</td>
</tr>
<tr>
<td></td>
<td>- Attribute</td>
</tr>
<tr>
<td></td>
<td>- Relation</td>
</tr>
<tr>
<td></td>
<td>- Value</td>
</tr>
<tr>
<td></td>
<td>- Compiled</td>
</tr>
</tbody>
</table>
Inferencing is triggered by monitors that send a message to the RM object when an object’s attribute is changed (forward inferencing), or when an object’s attribute is requested (backward inferencing). Multiple triggers (for example, through the employment of several monitors with a forward inferencing rule with multiple premises) are managed through the use of internal flags in the RM object.

When a rule is added to the object repository, the RM object determines all the objects referenced by the rules premises, and maintains internally a one-to-many relationship between the rule and referenced objects. This facilitates efficient determination of applicable rules when an infer message is received by the RM object. The RM object also maintains an up-to-date index relating rule names with their component classes, i.e., the premise(s) and conclusion(s). However the inference engine developed and encapsulated within the RM object is rudimentary and requires substantial development before it may efficiently handle large object volumes.

An initial prototype has been developed using the rule subsumption technique in the area of rapid response to customer requirements, where the KBS holds knowledge on the shop floor processes and makes this available to the sales force.

**CONCLUSIONS AND FUTURE WORK**

Whilst the use of the rule subsumption technique results in flexible knowledge bases, the authors realise that much more work is required in most parts of the system, in order to enforce the technique’s constraints and support the end user of the system.

By selecting sufficient functional components of the object-oriented model and of KBS systems, it may be possible to arrive at persistent object knowledge bases, which are constructed in two stages. Stage I is the use of advanced information modelling techniques to transform abstracted real-world knowledge into the most applicable formalism, be it rules, objects, data, etc. This can be expressed as:

\[
data \text{ as objects} + \text{knowledge as rules} = \text{knowledge modelling}.
\]
Stage II will comprise the building of Object Repositories (OR) that hold the formalised knowledge and information.

It is interesting to note that persistence for knowledge needs delicate handling, and will require mechanisms to deal with shelf-life of knowledge, truth maintenance and uncertainty. In modelling knowledge with advanced software for persistent object knowledge based systems, we are turning up the gain on information, and extracting the best possible use for our data. Such systems assist ‘information flow’ through organisations, and will be a key component of information systems supporting business process re-design. It is envisaged that future, basic research on RSOB will focus on how aspects of Description Logic (sometimes called Concept Languages) [12] may be incorporated in an OO framework.

REFERENCES


