Metalevel and Temporal Reasoning In KNOWBEL: Features and Implementation

Bryan M. Kramer
Department of Computer Science
University of Toronto
Toronto, Ontario, M5S 1A4
kramer@ai.toronto.edu

1.0 Introduction

General purpose knowledge representation systems serve two important purposes. First, such a system allows a knowledge engineer to model a new domain without being hampered by some special purpose mechanism and without having to invent new tools. Secondly, knowledge stored in a general purpose system becomes a resource that can be utilized by many users and applications within an organization. Three important requirements on such systems are that they be extensible and that they allow control of inference and temporal reasoning. This paper describes the language features provided to meet these requirements and the mechanisms that are used to implement them efficiently in the KNOWBEL knowledge representation system.

Extensibility is taken to mean the ability to define features and add assumptions to the knowledge representation language. KNOWBEL is extensible through metalevel features that can be used to define new kinds of classes, property categories and facets on slots, and concepts such as preconditions on classes.

A general purpose inference engine applied to a knowledge representation will not, in general, be efficient enough to be of practical use. KNOWBEL uses a form of metalevel reasoning[3][4] to allow the control of inference. The approach to metalevel reasoning used in KNOWBEL is unique in the mechanism used for switching, its integration with Telos objects, and the ability of the system to backtrack through metalevel decisions.

Finally, many domains can be modelled adequately only with an explicit representation of time. KNOWBEL has a novel linkage between its inference engine and a temporal reasoner based on Allen's interval calculus [1].

The features and mechanisms described in this paper are a part of a prototype version of KNOWBEL that has been implemented in Common Lisp.

2.0 Overview of KNOWBEL

KNOWBEL[6] is an implementation of the Telos knowledge representation language[7]. This implementation is intended to be a general purpose knowledge server for expert systems applications. A Telos knowledge base consists of a set of uniformly represented objects. There are several different kinds of objects distinguished by the role they play in the knowledge base. Classes are used to classify objects and specify a common structure for their instances. Every object must be an instance of some class. Metaclasses are classes that have classes as instances. Attribute propositions are objects that link objects to their attribute values. Attribute definitions are used within a class to define the attribute relationships instances. Every attribute proposition is an instance of an attribute definition. Instance propositions link objects to their classes and isa propositions are used to join classes in a specialization hierarchy. Inheritance of attribute propositions is strict - there are no defaults or over rides. Finally, there are assertion objects that represent sentences in Telos' assertion language. There are two kinds of assertion objects, one representing sentences in full first order logic, the other representing a restricted subset of first order logic. There are initially two uses for assertion objects. First, a class may have integrity constraints which are full first order assertions that are used to specify constraints that must be true of every instance of the class. Second, a class may have deductive rules which are assertions in the restricted language and are used to specify attribute values that are inferred rather than stored.

Time enters the picture as follows: every relationship, i.e. isa, instance-of, or attribute link, has an associated history time and belief time. The history time records the interval over which the relationship is true. The belief time records the interval over which the system believed this relationship to be true.

3.0 Metalevel Features

Extensibility is achieved through the combination of metaclasses, the assertion language, and occasionally, the use of the metapredicate holds in the assertion language. These mechanisms allow the definition of new kinds of classes and new kinds of attribute definition using the assertion language to influence their behavior. For example, the system supplies the attribute metadefinition single which is used in attribute definitions that require that
objects have only a single value for that particular attribute. This is done using an integrity constraint in the definition of single. An example using holds is the attribute definition for precondition which might be defined in a metaclass whose instances are classes describing activities. In such classes, a precondition must be satisfied before an instance can be created. Precondition is defined using an integrity constraint that requires that the assertion that is the precondition holds before instantiation.

Integrity constraints at the metalevel are expensive to check since they generally quantify over instances of classes and cannot be appropriately indexed. This is overcome through the use of a new assertion language predicate called compiled. When evaluated, compiled is always true, but as a side effect it adds to the knowledge base the integrity constraint specified by its arguments. Typically, compiled is used in metalevel integrity constraints for attribute metadefinitions such as single to move integrity constraint checking to the class level, i.e., classes containing instances of single, where indexing results in a very efficient check.

The second use of the metalevel in KNOWBEL is control of reasoning through the use of metarules[5]. Metarules are assertions dealing with the state of the inference, the contents of the knowledge base, and properties of the assertions about the domain. Reasoning at the metalevel results in conclusions that affect the course of inference. Examples of such conclusions involve the ordering of atoms within a goal and prioritizing certain lines of reasoning.

The implementation of metalevel control in KNOWBEL aims to limit reasoning with metarules as much as possible to situations where it is of most use. This decision is left to the knowledge engineer who specifies that metalevel reasoning should happen by associating metarules with patterns that will match classes of atoms. When an atom matches a metarule pattern, rather than performing a resolution, the theorem prover will switch to metalevel reasoning and allow the metarules to specify what will be done with the goal. Metarules are specified within classes and are efficiently indexed. One interesting feature is that the theorem prover will backtrack through metalevel decisions.

This mechanism has been used in KNOWBEL, for example, to convert the usual depth-first search to a best-first search where the cost of a node is specified by metarules.

4.0 Temporal Reasoning

The temporal reasoner uses a time point representation of intervals and uses a backward chaining search to determine the truth of a temporal query. Forward chaining is not used because, in a large knowledge base, the cost of storing the relationship between every pair of intervals would be prohibitive especially in KNOWBEL where belief times on temporal relationships result in a further explosion of the number of relationships between time points.

Performance in temporal reasoning in KNOWBEL is enhanced in two ways. First, the theorem prover provides a built in check for compatible belief times. Second, the connection to the temporal reasoner for history times is through the unification algorithm. This is done representing temporal arguments to predicates as variables that have time intervals as sorts[2]. Thus $\phi(c,<x>,Sv1/t1)$ unifies with $\phi(c,<y>,Sv2/t2)$ (the notation $Sv1/t1$ represents a variable having as its sort the interval $t1$ only if $t1$ intersects $t2$). Furthermore, the resulting substitution maps both variables to a new variable whose sort is a description of the intersection of the two time intervals. This technique saves a great number of inference steps and results in a smaller knowledge base than past systems which generally require an axiom that tests temporal relationships for every predicate having temporal arguments.

References