THE ROLE OF OBJECT ORIENTATION IN EXPERT SYSTEMS: A SHORT REVIEW

Frans Coenen
Department of Computer Science The University of Liverpool L69 7ZF

Since its inception the expert system community has been concerned with mechanisms to support the representation of knowledge. Much of this work is founded on the concept of objects as we conceive of them today. In some cases the term *object* is used specifically as in *Structured Object* or *conceptual object*, in other cases different terminology is used but the "hall marks" of object orientation are still present.

The earliest attempts at representing knowledge, in the expert systems context, can be trace back to Quillian's use of *semantic networks* for language understanding [7]. Quillian suggested that human memory could be modeled by a network of nodes representing concepts and relations, i.e. attributes and methods. The popularity of the network concept continued through out the 1970s, however many problems were encountered associated with; (a) information retrieval, and (b) lack of consensus on what the nodes and links actually represented (see [9] for a complete critique).

As a result of the above, researchers started to cast round for "simpler" knowledge representation mechanisms concentrating on the idea of *structured objects*, i.e. structures that combine both attributes and rules relating to those attributes, into a single "package". Typically such rules (often referred to as *demons*) defined procedures to be invoked when an attribute was assigned a value or was altered in some way. The most well known and popular results of this work is the *frame* based approach. Frames comprise *slots* which represent attributes, demons and inheritance rules. As such frames can be thought of as a *prototypical* conceptualisation of knowledge organised into a loose *hierarchy* (or heterarchy) where by frames "lower down" the hierarchy can inherit values from slots in frames "higher up" in the hierarchy (but not demons). Minskey described frames as "data structures for representing stereotyped situations" [6]. The process of creating an *instance* of a frame involves the *filling* of fact slots with specific information concerning the instance. However, to carry out any form of reasoning/inference some form of external program is required — the rules attached to individual frames are not sufficient for this task. The use of frames as data structures for storing information about prototypical objects and events is now well established in the AI community.

A further consequence of the work on networks and frame systems to represent knowledge was the development of various *knowledge representation languages* (as opposed to techniques). The best known early example of this approach is KRL (Knowledge Representation Language) [1]. The building blocks of KRL are referred to as *conceptual objects* arranged in a class structure. Other features of KRL that set it apart from earlier frame systems are that: (a) it provided for more general rules to be attached to objects (instead of demons hovering over slots), and (b) it allowed rules to be inherited by objects lower down the class hierarchy. Although KRL was really a vehicle for research into the theoretical nature of knowledge representation, the language it self is of interest here because it bears many of the "hall marks" of object-orientation (the concept of *message passing* is a notable omission) as espoused by subsequent work on OO programming languages, e.g. Smalltalk [3].

Although Smalltalk was available in the 1980s it was not until 1990s and the introduction of more "user friendly" OO languages (notably C++ [8] and later Java [4]) that the concept was widely adopted to later become the dominant programming paradigm that it is today. The popularity of the OO concept acted as a catalyst for the vendors of imperative programming languages, observing a commercial threat, to add OO extensions to their existing catalogues. "Expert system" languages (particularly LISP [5]) also underwent a similar metamorphosis. The AI languages LOOPS and Flavors were two early OO extensions of LISP. Languages such as LOOPS and Flavors included concepts such as message passing, multiple inheritance and meta-classes.

In 1984 the LISP programming language, which was breaking up into many dialects, was standaradised through international consensus; the result was Common LISP. CLOS (Common LISP Object System) is an object oriented version of Common LISP which subsequently eclipsed languages such as LOOPS and Flavors although retaining many of the features of such languages.

The popularity of object orientation as both a design and a programming paradigm is also evident in many current Expert System development environments most of which include object-oriented facilities. One example is CLIPS (C Language Integrated Production System) [2] which was developed at Johnson Space Center in the mid 1980s and is founded on a number of earlier LISP based tools. CLIPS provides an environment for the construction of both *rule* and/or *object based expert systems*. The object-oriented programming language provided within CLIPS is called COOL (CLIPS Object-Oriented Language) and is closely modeled on CLOS. Another example is AION which includes an OO programming language and a Callable Object Building Systems (COBS).

From the above; object orientation has been used, in one form or another, to represent knowledge in reasoning systems since their inception. Traditionally the inference engine has been written in an imperative language. In more recent years, in response to the growing popularity of language such as C++ and Java the inference engine has been written using some OO language. Their are now moves to bring OO inference engines and object oriented representations of knowledge together in a single OO reasoning systems a witnessed by development environments such as CLIPS and AION.

References

[1] Bobrow, D.G. and Winograd, T. (1979). KRL: Another Perspective. Cognitive Science, 3(1).

[2] http://www.jsc.nasa.gov? clips/CLIPS.html.

[3] Goldberg, A. and Robson, D. (1983). Smaltalk-80: The Language and its Implementation. Addison-Wesley, Reading, MA.

[4] http://java.sun.com/docs/books/jls/html/

[5] McCarthy, J. Abrahams, P.W. Edwards, D.J, Hart, T.P. and Levin, M.I. (1965). LISP 1.5 Programmer's Manual. MIT Press.

[6] Minsky, M. (1975). A Framework for representing Knowledge. Winston. P.H. (Ed.), The Psychology of Computer Vision, McGraw-Hill, New York, pp211-77.

[7] Quillian, M.R. (1968). Semantic Memory. In Minskey, M., Ed., Semantic Information Processing, MIT press, Cambridge, MA, pp227-70.

[8] Stroustrup, B. (1991). The C++ Prgramming Language, 2nd Ed. Addison-Wesley, Reading, MA.

[9] Woods, W. (1975). What's in a Link: Foundations for Semantic Networks. In Bobrow, D.G. and Collins, A. Eds., Representation and Understanding, Academic Press, New York.