Knowledge-Based Systems

Michael M. Richter

FB Informatik, Universität Kaiserslautern, D-67653 Kaiserslautern, Germany.
e-mail: richter@informatik.uni-kl.de

1 Introduction

Efforts to develop knowledge-based systems have now been pursued for over two decades. Languages, inference methods, architectures, and tools have reached a certain standard and have led to interesting and successful applications. Nevertheless, there is still some mystery and confusion among the public about this area. The purpose of this article is to provide a short and simple but nonetheless informative description of the topic including some indications of present trends in research. It should also be remarked that knowledge-based systems (as well as other techniques coming from artificial intelligence) are in no way mysterious and could be understood without knowing what “intelligence” really is. They are nothing more than problem solving methods which have some interesting and (sometimes) new features.

The term knowledge-based system may be somewhat misleading, because it suggests that one is now dealing with expert knowledge and that this was not the case in classical conventional programming. Clearly, classical programs also contained knowledge (sometimes very sophisticated). Also, when a knowledge-based system is translated into machine language there is almost no way to distinguish it from a conventional program. The principle difference in this new development is a different programming style. It is the declarative programming which characterizes AI programming. Declarative programming means that the programmer does not himself have to fix all the details of the execution of the program. Instead he puts down explicitly those facts and laws which in his opinion are necessary and useful for the solution of his problem. Declarativity also implies that the language in which the domain of interest is described has a fixed semantics in which the meaning of the language constructs is described. The semantics refers to a certain domain of interest called a model; the description of such a model constitutes a large part of the knowledge base.

In order to execute the program the missing details are generated by the control structure and the inference engine. The inference engine deduces logical conclusions from the input knowledge as well as plausible hypotheses and related pieces of information. The control structure organizes this process. The separation of the knowledge base and the inference engine was a major argument in promoting knowledge-based systems: The entries in the knowledge base could vary with the application while the inference engine remains
fixed, a fact that should allow a new flexibility compared with classical programming. In this context a knowledge-based system with an empty base was called a shell. Below we will discuss how this view turned out to be very overoptimistic. A knowledge-based system can be considered on three levels which should be clearly separated, the cognitive level, the representation level, and the implementation level.

On the cognitive level the user demands can be expressed. The formal representation on the second level corresponds to a specification in a conventional program. The user demands also describe the functional behavior, i.e., the input-output relation of the system. The functional view is realized and complemented by the architectural view which describes how the functional behavior is realized. The architectural view is realized on the implementation

\[ \text{Cognitive Layer} \]

\[ \text{Representation Layer} \]

\[ \text{Implementation Layer} \]

Fig. 1. Between the knowledge utterance and its machine utilization several transformations must be performed (thick arrows). They map in the direction of increased structuring within the layer and proceed from the cognitive form to the formal, and from here to the efficiently processed form. Each syntactic result obtained in the range of a transformation must be associated with its meaning in the domain of the transformation. This is indicated by thin arrows between the layers. The most interesting and difficult one is the inverse mapping into the cognitive layer, which is usually called explanation. The letter A is intended to remind us of "Acquisition" (which is human oriented) while the letter C is short for "Compilation" (which is machine oriented).