The Knowledge Engineering Review, Vol. 11:4, 1996, 373–377

Book reviews

The Knowledge Acquisition and Representation Language, KARL by Dieter Fensel, Kluwer Academic, Dordrecht, 1995, pp 238, ISBN 0-7923-9601-4, Dfl 195.00 (US$ 110.00, £77.00).

There are a large number of formal specification languages, but most are aimed at software engineering rather than knowledge engineering. Formalisation of software has been accepted by academics for a long time now (decades), although it is less widely accepted in industry.

One debate, almost religious, concerns the benefits or otherwise of executable specifications. David Parnas distinguishes between a "specification", a statement of properties, and a "model", a product with some but not all of the properties of some real product. Others do not delineate specifications and models so sharply. Executable specifications are really models in Parnas' terms of reference. Modelling is a useful exercise, but should not be confused with a higher-level specification. A possible approach to the dilemma is to use a wide-spectrum language which includes both executable and non-executable components, together with a means to transform non-executable parts into executable ones under human guidance.

The development process will typically include a specification of requirements and a specification of the product (e.g., software or "knowledge") to be designed. The requirements specification does not consider the product but rather the complete system, including the environment, and will in general by very domain specific depending on the system being considered. A specialised requirements language may be helpful in many cases. Formalisation, often only undertaken at the programming stage when it is unavoidable, may be advanced to the design, specification and even requirements stages if desired, resulting in the earlier and cheaper discovery and avoidance of many errors.

This book presents a formal specification language for use in knowledge engineering rather than software engineering. While there are many formal notations available to software engineers, some of which are well established even if not widely used, the concept seems to be more novel in the domain of knowledge engineering.

The notation presented and formalised in detail here is KARL, developed as part of the MIKE project (Model-based and Incremental Knowledge Engineering). This is a combination of two languages, a logical part (L-KARL) for static knowledge, based on ideas from logic programming (Horn logic extended by stratified negation) with object-oriented features, and a procedural part (P-KARL) to describe knowledge about the control flow. The language is formal and operational in nature. P-KARL is deterministic with no general recursion, which could be somewhat limiting in practice at the specification level.

The book includes discussion on the pros and cons of executability in a specification language. KARL falls into the executable specification camp which allows rapid prototyping. The problem of addressing executability at the specification level is that it limits expressibility. Effectively it means the language is a (very) high-level programming language rather than a more general specification language. If this is accepted, it is a useful medium for modeling in the development process. However, it does not replace a high-level specification or requirements language where executability should (initially) not be an issue.

The introduction covers the main arguments concerning formal specification reasonably comprehensively, including those concerned with executability. The logical and procedural aspects of KARL are presented, although the latter is rather cursory. A long chapter on the KARL "model of expertise" differentiates between the connected domain, inference and task layers, illustrated by a running example. This is based on the "model of cooperation" of expert systems with other agents.

The formal semantics of logical L-KARL, procedural P-KARL and the three layers above are presented in separate sections, concentrating on a model-theoretical semantics for L-KARL. However the relationship between the languages and the layers is not made explicitly clear. A separate section or chapter on this aspect would have been helpful.

The conclusions include related work, providing a comparison with the knowledge specification language $(ML)^2$ developed as part of the KADS project, shortcomings of KARL, and future work. The reference list is usefully comprehensive and the index is adequate except for abbreviations (e.g., KADS, KARL, MIKE and $(ML)^2$) which are not included.

This work is based on a PhD thesis, and is thus fairly specialised. It will be of interest to researchers in the area of the formalisation of knowledge engineering, and of more indirect interest to those in the field of formal methods in general. Libraries with a comprehensive section on knowledge engineering or artificial intelligence will probably want to acquire this book but its readership will be limited by its very nature. The approach and presentation is very mathematically formal in general; although KARL is executable, there is no obvious indication as to what tool support is available or planned, if any. Without such support, the executable aspects to not seem very helpful.

In conclusion, I would prefer to see KARL considered as a formal modeling language rather than a specification language for reasons as outlined above. I am glad to see some unification and interconnections of approaches adopted by software engineering and knowledge engineering researchers; I hope that each community can learn something from the other.

Reviewed by Jonathan Bowen, The University of Reading, Department of Computer Science, Whiteknights, PO Box 225, Reading, Berks RG6 6AY, England

Reasoning about knowledge by Ronald Fagin, Joseph Y. Halpern, Yoram Moses and Moshe Y. Vardi, MIT Press, Cambridge, MA, 1995, pp 477, $45.00 cloth, ISBN 0-262-06162-7.

One way or another, academics spend much of their working lives reading. Sadly, most of us find that a large proportion of what we read is not terribly interesting. We review articles, correct student work and read books by the truck load simply to keep abreast of developments in our field. But little of this reading is done with real pleasure or interest; most of it results from a weary sense of duty. Fortunately, every now and then, a book comes along that reminds you what reading *should* be like; both enjoyable, and a real learning experience. I am happy to report that *Reasoning About Knowledge* is just such a book.

The tradition of using modal logics in the formal analysis of knowledge and belief is generally reckoned to have begun in earnest with the work of Jaakko Hintikka. It seems unlikely that Hintikka, publishing his classic 1962 book *Knowledge and Belief*, could have anticipated that his work would have any application outside the somewhat ivory tower tradition of formal philosophy within which he then worked. And yet, three decades later, modal logics of knowledge are a standard tool in both AI and theoretical computer science, and are the subject of much ongoing research.

Within AI, logics of knowledge are considered to be important because we are increasingly concerned with building *agents*: autonomous, self-contained computer systems, that are capable of communicating and cooperating with other agents in order to achieve their goals (Wooldridge & Jennings, 1995). It is commonly accepted that in order to do this, an agent must be able to manipulate representations of its peers. Logics of knowledge are intended to support exactly this kind of reasoning. To better understand what this reasoning looks like, consider the following puzzle, which is often used as a kind of benchmark for logics of knowledge:

> A certain king wishes to determine which of his three wise men is the wisest. He arranges them in a circle so that they can see and hear each other, and informs them that he will paint a white or black spot on each of