AI Magazine Volume 14 Number 2 (1993) (© AAAI)

Advances in Real-Time Expert System Technologies

Franz Barachini

■ The Workshop on Advances in Real-Time Expert System Technologies was held on 3 August 1992 in conjunction with the Tenth European Conference on AI. Participation was limited to invited researchers only. The workshop focused on practical problems occurring during the implementation of real-time expert systems. In this respect, different industrial applications were discussed. The debate covered a wide range of applications, such as qualitative simulation and anytime algorithms for real-time process control. The workshop showed that real-time expert system techniques are getting more attention, even in Europe.

The Workshop on Advances in Real-Time Expert System Technologies was held in conjunction with the Tenth European Conference on Artificial Intelligence in Vienna on 3–7 August 1992.1 The workshop and conference were organized by the European Coordinating Committee for Artificial Intelligence and hosted by the Austrian Society for Artificial Intelligence. A summary, my personal impressions, and future directions are presented here.

Expert systems are technologies to support human reasoning by formalizing expert knowledge so that mechanized reasoning methods can be applied. In real-time systems, these reasoning methods must be reactive to external events and have to abide by stringent timing requirements. This behavior, known as timeliness, is infrequently achieved in expert systems.

One possible solution to the problem is the use of *anytime algorithms.* Anytime algorithms are algorithms whose output improves over time. Instead of creating the ultimate correct solution, such algorithms try to get better and better results the longer they run. Participants argued that anytime algorithms are useful in real-time systems and will play a more important role in the future. Real-time systems might be partitioned into a control part and reasoning parts. If the reasoning parts follow the notion of anytime algorithms, they could be interrupted by the control part as soon as the environment forces an interrupt. However, it has been shown that pattern-matching algorithms such as RETE or TREAT cannot be interrupted at any position for consistency reasons; therefore, the question of how to intermingle heuristics with anytime algorithms is not solved sufficiently and requires deeper analysis.

Generate-and-test methods can be regarded as anytime algorithms. One theoretical paper by Carl-Helmut Coulon from the German National Research Institute for Computer Science addressed the problem of defining a utility function for generate-and-test methods. These methods are conceived as incremental nonheuristic algorithms that can be called repeatedly to generate and test a hypothesis. As a stop criterion for a generate-and-test system, a lower bound on the number of solutions to be produced and an upper bound on the time to be spent were defined. Four strategies for utility management were presented and compared. Although the four strategies were ad hoc strategies, the analysis gave deeper insight into the behavior of such systems.

Another possible solution to the timeliness problem was presented by Gilles Verteneul from the Alcatel-ELIN research center. Substantial research in predicting run time for RETE algorithms has been conducted there. RETE and TREAT pattern-matching algorithms are widely used for real-time production systems because of their good performance. Their run-time worst-case behavior is exponential, but in reality, this behavior seldom occurs. The presented upper-bound method yields an upper bound for the match of a basic action in the RETE network. The idea is based on the fact that left and right memories in two-input nodes can be partitioned into intervals. By estimating these intervals statically (before run time) and calculating the number of tokens to be matched throughout the whole network, an upper bound for the complete number of matched tokens can be given. This upper bound is much closer to reality than the theoretical worst-case complexity of the match would lead us to expect. On average, the upper-bound method predicts five times more matches than actually performed during run time. The method can be seen as a generalization of the unique attribute technique used in SOAR systems. It was mentioned that the time needed for performing the prediction can be rather large, sometimes twice the time needed for the match. It can be concluded that the upper bound is only a first step in the direction of run-time prediction for production systems because it is unable to predict run time for several recognize-act cycles.

Another interesting discussion at the workshop was on the efficiency of managing temporal facts in rule-based systems. Expert systems that make use of temporal reasoning require some form of automatic lifetime management for temporal facts. Whereas others have used inefficient truth maintenance systems for this purpose, a pragmatic approach has been taken in the language PAMELA-C. The proposed reasoning scheme is able to handle only events that have occurred at a discrete time point in the past, and no hypothetical reason-

Copyright © 1993, AAAI. 0738-4602-1993 / $2.00

ing about future events is supported. Only relative temporal dependencies between events can be specified by the user. It can be considered an intelligent garbage collector for RETE networks handling temporal facts.

Three of the application-oriented papers presented deserve special attention. First, Jean-Luc Dormoy from EDF Research and Development Center presented an architecture for building real- time systems from models by using a model-compiling technique. He claimed that classical model-based reasoning techniques are of no use to real-time problems because of their low performance. Therefore, an architecture called KSE, which has been used in nuclear plants, has been developed. The aim of KSE is to causally explain undesired events in a plant and provide the operators with a description of the plant operational state. To achieve high performance, knowledge-compiling techniques were used to automatically generate a sufficiently fast expert system from a model-based description of the plant.

KSE contains three large knowledge chunks. The first chunk, which comprises 12,000 components and 150,000 attribute-value pairs, represents the model of the plant. This description of components and relations between them is modeled in an object-oriented way. The second chunk is a description of causal relationships between components. This description is represented by 250 so-called principles that take the form of a subset of predicate calculus causal implications. The third chunk is a simple logical model of the intended operation of the system, namely, deducing a description of the plant state from the instrument data that is as complete as possible, generating possible assumptions, and removing the assumptions. The logical model consists of six general rules.

The knowledge compiler of KSE transforms the logical model into first-order production rules. These rules are then compiled into zero-order rules that can be seen as hard-wired if-then statements. In the current version of the plant, 47,000 zero-order rules have been generated. It was claimed that the worst-case run time of the whole system is no more than five seconds. However, the compilation process needs more than 10 hours.

Second, Monika Pfau-Wagenbauer and Thomas Brunner from Siemens Austria discussed the functioning of an expert system acting as part of a supervisory control and data-acquisition (SCADA) system for the public utility board of Singapore, controlling its 22-kV distribution network. The expert system is an operator support tool that diagnoses network disturbances and device malfunctions. The SCADA system provides the expert system with relevant (filtered) process data and meets hard real-time deadlines. In contrast, the expert system runs on separate hardware and does not guarantee response time.

The topological data representation in the knowledge base is modeled in an object-oriented way. There are hierarchical diagnosis levels using heuristic rules as well as compiled model-based knowledge. Based on a dynamically determined time window, it is decided when the main diagnosis process has to start. During the passing of a time window, disturbances are gathered from the SCADA system, and a prediagnosis method uses relevant component models to diagnose correct protection system behavior. The observed behavior is compared to the correct behavior of the models, and conclusions about the correct behavior can be drawn. In this way, correct behavior assumptions increase monotonically. These assumptions are used later by the main diagnosis, which is structured as a hierarchy of different rule classes. The rule classes comprise 190 rules. There are about 25,000 objects in the system. The average reasoning time is about 5 seconds, which is satisfactory because the SCADA system itself needs 8 minutes to scan and filter all 10,000 peripheral events.

Third, a paper by Zsusa Csaki and Karl Hangos from the Computer and Automation Institute of the Hungarian Academy of Sciences described lessons learned from using qualitative simulation in a chemical plant. The main message was that qualitative simulation–based advice generation for operators seems to be too complex to guarantee stringent timing requirements. To overcome the problem, only small, intermediate simulation steps should be performed, and these steps should be guided by the operator according to his/her heuristic knowledge. In this way, the simulator is supported by choosing the most interesting branch in the system's behavior tree.

The workshop showed that real-time expert system techniques are getting more attention, even in Europe. Timeliness and reactivity will play a role not only in so-called low-level tasks but also in planning and reasoning. Future systems will be heterogeneous in nature, comprising different reasoning methods as anytime algorithms, probabilistic reasoning, and subsymbolic techniques. The issue is how to put it all together. In gluing different techniques together, the engineering task gets complex, and as the size of the systems grow, the correctness issue becomes more important. Using design and verification methods might be one answer to dealing with complex heterogeneous architectures. Thus, AI and software engineering must definitely come together. Acknowledgments I would like to thank Wolfgang Nejdl, the coorganizer of the workshop, for his comments and Peter Patel-Schneider for proofreading.

Note

1. The proceedings of the workshop can be obtained from Alcatel-ELIN Research, Ruthnergasse 1-7, 1210 Vienna, e-mail: Franz.Barachini@rcvie.co.at.

Franz Barachini received his M.Sc. in 1980 and his Ph.D. in 1983, both from the Technical University in Vienna. He spent six months at Stanford University as an invited researcher in his major area of interest—real- time knowledge-based systems. He is currently head of the Knowledge-Based Systems Department at the Alcatel-ELIN Research Laboratory in Vienna. He is a member of the American Association for Artificial Intelligence, the Institute of Electrical and Electronics Engineers, and the Austrian Society for Artificial Intelligence. Since 1992, he has been a lecturer at the Technical University in Vienna.