WORKING NOTE AC56

Alan Creak
4 August 1986

RESOLVING CONFLICTS CONSTRUCTIVELY

Classical logical systems are closed worlds. Assertions are true or false, relationships between different components of the system do not change, and a chain of inferences, once made, endures for ever. The systems are *monotonic*.

The real world is not so simple; but there are plenty of cases where classical logic is sufficiently close to reality (or vice versa) for conventional logic to be useful - if that were not so, classical logic would presumably never have been invented ! Expert systems rely on knowledge bases which contain, along with factual information, rules for arguments which we know to be reliable or to which we can attach some measure of reliability. In either case, the results are derived by logical processes, and, though their reliability may be in question, the question attaches to the validity of the rules, not to the logical process themselves.

In other cases, though, the rules themselves may be inconsistent with the known facts. If this happens, either the rules are wrong, or the facts are, in some sense, unacceptable - we have a conflict, and to resolve it we must change either the rules or the facts.

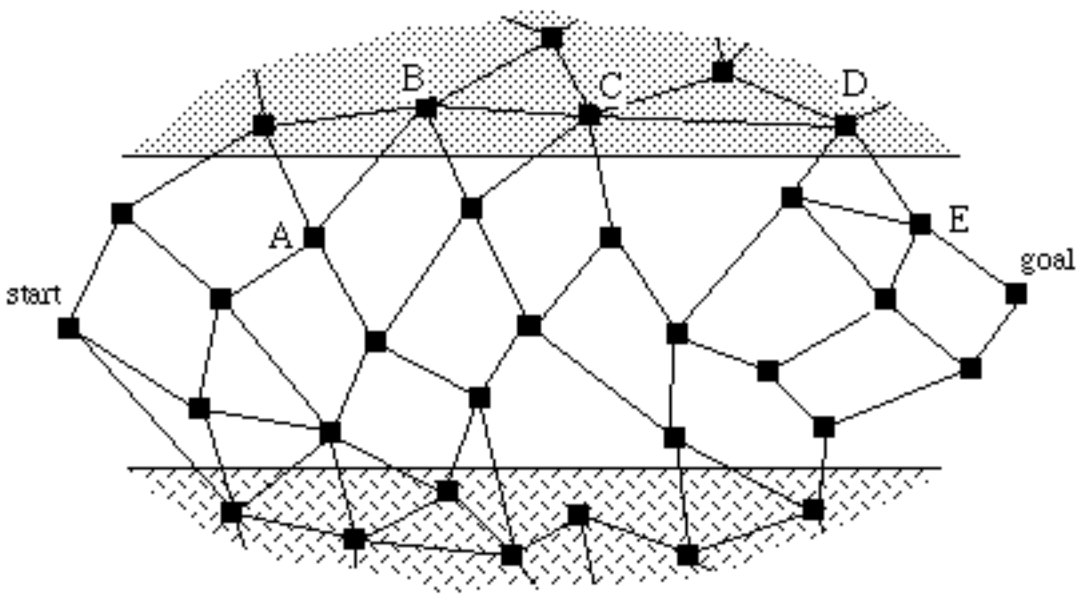
Which one we choose to change depends on circumstances. If our facts are well established and the rules are meant to describe or interpret the facts in some way, then we must change the rules. An example is scientific research, where the rules are the hypotheses we make to account for our observations, and must be changed if they don't fit the observations. If, on the other hand, the rules are meant to control the facts, then the facts must be changed - as when the rules are regulations controlling some activity, or conditions which must be satisfied.

We shall concern ourselves with the second sort of conflict. We shall be interested in systems which are governed by rules, but in which conflicts sometimes occur; and we shall expect that such conflicts should be in principle resolvable by taking some action which changes the "facts", whatever they may be. We can regard our software as an *intelligent observer*, which watches events as they unfold, notices when conflicts arise, and proposes possible courses of action which will restore the system to a normal state. We may expect that certain topics will be of interest:

- How to identify the STATES of the system;
- The RULES which describe the acceptable states of the system;
- The possible CHANGES in state of the system, and in particular the changes which move the system from an acceptable state to a conflict state, and back again;
- STRATEGIES for identifying remedial actions in cases of conflict.

It is worth commenting that we do not reject the conflict. Our computer system must be able to represent it, if only because it may take some time to resolve. The "incompleteness" of Prolog is therefore an advantage : a system which would not accept an inconsistent fact would not be useful.

We can represent thes ideas pictorially by leaning on the notion of a problem space, in which each state of the system is denoted by a node, and possible transitions between states are represented by arcs joining the nodes. The Figure is an impressionistic view of such a problem space.



Here, the "normal" states lie in the central strip, and conflict states lie in the shaded areas above and below. While such a diagram can naturally not be regarded as any sort of proof, it suggests certain useful ideas : for example, the best way to resolve a conflict such as that represented by the node B is not necessarily to backtrack to A; it may be better to continue through C and D, finally resolving the conflict with the move to E.

There follow three briefly described examples which illustrate how these ideas work out in practical circumstances.

Operating Systems

We may think of the normal state of an operating system in terms of self-sufficiency: the system is "normal" when no action need be taken. A conflict state, on the other hand, is a state requiring intervention. An intelligent observer monitoring such a system could identify the attention-needing states, and make recommendations to the operator regarding appropriate action to be taken.

The effectiveness of such an observer evidently depends on the observer's view of the system (the problem space and rules), and information on changes which take place. All of these factors may be more or less complete : a human operator may have a very imperfect understanding of the system, but is nevertheless able to keep it going tolerably well by following a few simple rules, even with the very limited information as to the systems change of state displayed on the operator's console.

IBM's expert system YES (1) is designed to work in this area. Modelled on a human operator, it has the advantage of seeing all the system's messages (a human operator will inevitably miss a sizeable proportion of them), and with that information and a model of the operating system derived from computer operators' expert knowledge it is able to maintain a record of the system's state, and to advise the human operator an action to be taken when necessary.

University Enrolment.

A student enrolling for a course of study at the unversity does not usually have a free choice of all papers offered. Most students enrol with a particular degree in mind, and a degree is only awarded if certain rules are satisfied. The primary requirement is that a certain amount of work should be done at each of stage 1, stage 2, and stage 3, and there the limits to the number of papers for which a student may enrol in one year. There are also more specific constraints: every paper may carry its own requirements for prerequisites, corequisites, and restrictions, so the selection of papers at stage 1 restricts the choice at stage 2, which in turn constrains the possible stage 3 course. Other regulations govern transfer of course credits between faculties, eligibility for bursaries, fees payable, and so on.

The enrolment procedure itself is distributed. Each department handles enrolment in its own papers; so students go from department to department, at each point dealing with (usually) an academic member of the department who is fairly well informed of the department's own course, but much less knowledgeable of the requirements of other departments, and not well placed to see the ramifications of a student's choice over a particular proposed course of study. In such circumstances it is all too easy for a student to enrol in a set of papers which, for one reason or another, is in conflict with the regulations.

Here the rules are the university's regulations, the state of the system is the student's university record, proposed course of study, and ultimate aim, and each enrolment in a paper represents a change of state. An intelligent observer should be able to notice when a proposed course creates such a conflict, and to suggest possible ways of resolving the difficulty.

Aeroplane loading schedules.

The system in this example is a given schedule of aeroplane flights and capacities, and the rules are concerned with how goods can be packed into the available aeroplanes. The changes of state are occasioned by new cargo consignments; they may conflict with an existing schedule, particularly if they have a high priority. An intelligent observer in this case should be able to identify conflicts, and suggest possible ways of resolving them. The circumstances of this example bring out particularly clearly the need to identify as many solutions as possible, as it is obviously important to choose the most profitable new schedule.

A study on a system which is at least superficially very similar to this scheduling problem has been carried out by M.S. Fox (2). He has worked on the application of artificial intelligence techniques to "job shop scheduling" - the planning of work schedules in a machine shop to satisfy a contimnuous stream of orders of various priority levels. He found it possible to a large number of widely disparate constraints into his implementation, and produced satisfactory schedules.

Approaches.

In work on conventional expert systems, it is found that different approaches are needed in different circumstances; in consequence, many different expert system building "tools" are available.

The sort of problem outlined here, where the environment is constantly changing, is still a subject of active research. It seems probable, though, that a variety of circumstances will again demand a range of responses. IBM's work on YES was based on OPS5, while Fox used SRL in his work on job-shop scheduling. In contrast, KEE was used in previous work on the aeroplane loading problems itself (3). It is far from clear that the tools found useful in static expert systems will be the best for use in changing environments, and in both the IBM work and Fox's study substantial development of the underlying expert system base was found to be necessary.

We hope that further consideration of topics of the kinds outlined here will lead us to a deeper understanding of the factors involved, and of appropriate means of solution.

REFERENCES

(1) J.H. Griesmer, S.J. Hong, M. Karnaugh, J.K. Kastner, M.I. Schor, R.L. Ennis, D.A. Klein, K.R. Milliken, H.M. VanWoerkom : "*YES/MVS : a continuous real time expert system*" (IBM research report RC10461 REVISED (#46762), 21 June 1984)

(2) M.S. Fox, B.P. Allen, S.F. Smith, G.A. Strohm : "*ISIS : a constraint-directed reasoning approach to job shop scheduling*", Technical Report CMU-RI-TR-83-8, Carnegie-Mellon University Robotics Institute, 1983; M.S. Fox : "*Constraint-directed search : a case study of job-shop scheduling*", Technical Report CMU-RI-TR-83-22, Carnegie-Mellon University Robotics Institute, 1983.

(3) Hearsay, so far.