PLANNING SYSTEMS AND ARTIFICIAL INTELLIGENCE

K.-D. ALTHOFF, N. KRATZ, M.M. RICHTER and P. SPIEKER

University of Kaiserslautern, Department of Computer Science, P.O. Box 30 49, D-6750 Kaiserslautern, W.-Germany

Abstract

Traditional O.R. systems are compared with problem solving in Artificial Intelligence via Expert Systems. The discussion centers on explicit knowledge representation. The general aspects are illustrated by two planning systems:

- LESP 2: A learning system for inspection plan generation
- IDA: A system for finding functions and solutions in construction

Keywords: Planning systems, expert systems, knowledge representation, inspection plans, construction plans.

0. Introduction

Traditionally, planning systems have as a rule been generated using O.R. methods. In recent years Expert Systems and AI methods have also served to tackle these problems. Both alternatives have their specific advantages and weaknesses, which in turn restrict their appropriate fields of application. Roughly speaking, the use of AI methods appears adequate, if there is a lack of uniform mathematical modelling and complex knowledge structures require flexible possibilities of expression which do not exist in the language of conventional algorithms.

In the following we will discuss these aspects and will present two Expert Systems–LESP 2 and IDA–as examples of planning systems in which AI methods are applied.

1. Aspects of traditional O.R. system

If O.R. methods are used one tries to describe a given situation by means of modelling, which allows to solve problems through mathematical operations and algorithms. This provides the advantage that due to a considerable level of know-how and experience in this domain many cases can be handled in an optimal way. The scope of such methods is being extended permanently. Thus it has become possible to deal with problems of impressive dimensions far beyond human abilities. There is no reason to replace O.R. systems working in a satisfactory manner by any other methods.

There are, however, many tasks resulting from real applications where pure O.R. methods are not able to supply "adequate" solutions. Here the term "adequate" is interpreted in different but always informal ways. This is revealed by the fact that the language capacities of knowledge representation in O.R. systems are rather restricted. A language with a limited scope of expressions is not–or not sufficiently–adequate to describe complex situations.

First of all we will consider the design of a mathematical model for an O.R. procedure. Here the intuitive relations between the objects of the real situation are lost, unless they are explicitly part of the model. These semantic relations then only exist in the designer's mind. They can only be communicated by references to agreements made earlier; but above all they are not accessible to the system itself. Particularly in the case of highly efficient procedures, due to the simplicity of the model, intentions of the initial problem may in addition be misrepresented.

The procedures following the modelling stage use again a language with an expressive power comparable to the propositional calculus. This does not exclude at all that far more complex problems can be dealt with. They have, however, to be encoded, and it becomes impossible to recognize them from the linguistic point of view. The designer who knows the code can of course talk about the system, but he is no longer able to make these "meta-reflections" accessible to the system. If such meta-reflections are sufficiently complex the user may wish to automate them. If this attempt fails due to an insufficient expressiveness of the programming language, then for the corresponding tasks a solution in general or a good solution (both in the sense of the model) may no longer be found.

The loss of semantic information results in the fact that O.R. methods are more or less used as black box methods. It is generally impossible to explain parts of the solution to a naive user. In particular there is no possibility of adding a flexible explanation component to the system. This has two consequences: Firstly, even a suitable solution may not be accepted if it is not entirely understood. The mere mathematical correctness of a solution would not be considered to be adequate. Secondly, the possibilities for interactive work are strongly restricted, too. The user can only apply his skills if he understands sufficiently well the actual state of a problem solving process. If this is not the case the problem may not even be solved at all.

2. Approaches to problem solving in AI: expert systems

Since the mid sixties, parallel to the development of O.R. systems, research groups have been engaged in solving problems which were either not adequate representable in numerical forms or could not be solved analytically due to their high complexity. Here the fundamental new idea was to solve such problems by imitating human patterns of thought and behaviour.

Essentially, Expert Systems differ from O.R. systems in providing explicit representations of the available knowledge. Here explicit knowledge representation means that knowledge is represented in a formalism which is similar to the way it is structured in the mind. These demands require new programming techniques which will be briefly presented in the following sections. Typical applications are problems with varying or dynamic constraints.

The methods of knowledge representation that have been developed and applied basically by Artificial Intelligence are no longer primarily oriented towards traditional models of computation and computer architectures, but towards cognitive human abilities. Hence one is farther away from traditional formalisms and available hard- and software but closer to human problem solving abilities. One of the main challenges to Artificial Intelligence is to study situations in which these methods can be applied efficiently.

2.1. A SURVEY OF THE MOST IMPORTANT METHODS OF KNOWLEDGE REPRESENTATION

One of the striking properties of methods for (explicit) knowledge representation is their strongly declarative character. This allows to represent knowledge about a problem without already programming the actual algorithms for a solution. This solution is found by an inferential component which is applied to the available knowledge. In this way a large scope of problems can be tackled.

So far four methods have been established:

(i) Rule-based representation is the earliest one. The knowledge is put into so-called "if-then" rules which are later on processed by a rule interpreter. This way of representation is based on the assumption that the knowledge of experts can be defined adequately in this "if... then" manner. Therefore, the early Expert Systems such as DENDRAL (Buchanan et al. [3]), MYCIN (Shortliffe [12]) or R 1(Mc Dermott [8]), were purely rule-based systems. Although these systems have been successful in principle, they nevertheless revealed the deficiencies of the rule-based approach. Declarative knowledge representation of contexts and nested facts turns out to be very tedious; moreover structuring is not supported. Procedural knowledge about the application of the rules which is necessary for their efficient use can again only be expressed implicitly.

(ii) The use of predicate calculus is a well-known, theoretically well-founded and widely favored way of representation. Languages such as PROLOG (Clocksin et al. [4]) and Krypton (Brachman et al. [2]) are results of this approach. Yet, until now, for these languages there have only been available inefficient (Krypton) or incomplete and incorrect (PROLOG) deduction components. The variety of possible methods of representation in predicate calculus proves to be a handicap. Still complicating is the fact that it is of first order and causes problems with regard to meta-knowledge.

(iii) Frames play an important part in modelling highly declarative aspects. This mode of representation is based on the idea that human beings perceive their environment in contexts. In addition to certain characteristics of these contexts the relevant procedural knowledge is also handled. Initially, frames arose from an extension of semantic networks which are not based on a procedural interpretation (Minsky [9]). The fundamental notions are inheritance and default reasoning. Although (or because) virtually all aspects can be expressed by frames there is still considerable lack of experience to use frames efficiently. These points need much more consideration.

Along with the idea of message passing as a procedural environment for frames the concept of object-oriented and programming was created. Today frames are quite often preferred to production rules and predicate calculus because they allow to integrate a natural mode of description of declarative knowledge into an established concept meeting the demands resulting from Software Engineering, particularly during the encoding stage. Using object-oriented concepts, notions like modularity and information hiding can be consistently realized. The most popular implementation of this programming style is Smalltalk-80 by Xerox PARC (Goldberg et al. [5]).

For quite a long time not enough importance was attached to these aspects, which becomes apparent when one considers languages such as OPS-5 or PROLOG. The absence of any explicit means of structuring in these systems has turned out to be an unacceptable obstacle for the management of large-scale applications (e.g. if several people cooperate in one project). Meanwhile modular concepts (M-PROLOG) modelled after procedural programming languages (MODULA-2) or world concepts (MULTI-LOG) (Kaufmann et al. [7]) adapted more strongly to the logic background have reached a higher level of popularity.

Also with respect to knowledge representation frame-based systems have an advantage over production rule-based or logic-based systems. Particularly in planning and configuration systems the problem of adequately representing the central object, the plan, is a difficult task. In his article on frames Minsky refers to the fact that the complexity of configuration tasks can be reduced by means of efficient assumptions, so-called default values. These are parameters of the plan which are given values in advance and enable one to define the remaining parameters under adequate restrictions. Only a later contradiction results in a withdrawal and change of these parameters.

All Expert Systems have in common the necessity to represent strategic knowledge, generally known as meta-knowledge. As this meta-knowledge has a planning character even in diagnostic systems, frame-based knowledge representation is a great help. Therefore the structuring possibilities provided by frames are essential also in diagnostic systems.

(iv) In order to improve, respectively to increase the expressiveness of production rule systems, the possibilities of structuring, hybrid systems (LOOPS, KEE, BABYLON etc.) have been created and made available to designers of Expert Systems. Essentially, all systems proceed from an object-oriented basis and add rules, logic and/or functional elements. Here the main disadvantage is a lacking support for choosing an adequate mode of representation and jumping to another mode if the situation indicates this.

2.2. ADVANTAGES OF EXPLICIT KNOWLEDGE REPRESENTATION

Comparing conventional programming languages and explicit knowledge representation we observe the following differences (as far as software is concerned): As shown in fig. 1, the use of AI methods reduces the encoded and therefore invariant part of the system. In addition to the knowledge representation mechanisms and the inferential component there are conventional programs, e.g. mask or file managers.

In the case of AI methods the explicit knowledge takes up by far the biggest part of the system. Here a fixed component of the system is represented in a way which is directly accessible to the user (respectively to the expert). Thus the expert's assistance in the design or modification of the system in order to adapt it to changing demands can be simplified.

Another advantage is the availability of knowledge in order to generate explanations automatically. Even a simple trace of the rules applied in a production rule system can explain much better the process of problem solving in the system than a register dump or a procedure trace of a conventional system. The information given to a user can vary–according to the respective user type–in the kind of details and emphasis given. This makes it possible to get more sophisticated explanations which have very little in common with the dumps or traces

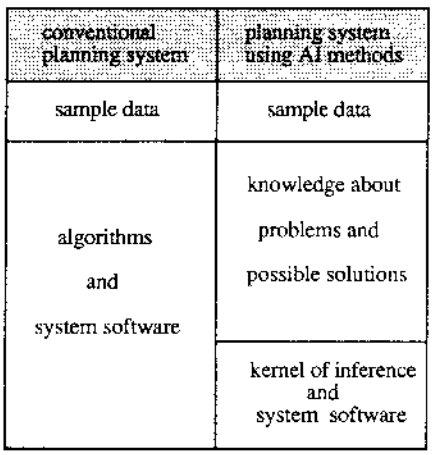

Fig. 1. Comparison of conventional with AI planning systems.

mentioned above. The requirement for this ability is not only given by the loss of the black box character which causes an increasing degree of acceptance. In addition the system allows to check the results in domains requiring reliable security standards (medicine, process engineering etc.).

Knowledge representation, considered as a very high-level programming language, has still further advantages:

Most representation methods have a special expressive power, which permits to design programming systems with a great variety of functions. This is the reason for their high level of flexibility i.e. these systems can be applied to a wide range of particular problems. The price to pay is a loss of performance. This effect has only partially been compensated by more efficient hardware. It should be emphasized that in Expert Systems parts of the tasks should be handled by conventional programs whenever this is compatible with the rest of the system.

Due to the declarative character of Expert Systems the development of large systems can be supported by an incremental design process. Whereas the conventional procedure requires a system specification on the whole range of functions and later changes in the design or even encoding phase turn out to be expensive. The concept of prototyping plays an important role in Expert Systems. Here we refer to a program development which provides a preliminary version which is gradually extended. For that purpose especially the already existing components of the system can be used.

3. Examples

A wide range of applications for planning systems can be found in construction, production planning and quality control. The two Expert Systems presented in the following can be placed in these categories.

IDA, an Expert System which supports the conceptional stage of construction in Mechanical Engineering, and LESP, a learning system serving to generate inspection plans have been developed in the Laboratorium für Werkzeugmaschinen und Betriebslehre (WZL) at the Technische Hochschule Aachen (Althoff [1], Kratz [6]). The IDA project is now continued in a joined research project (SFB 314) at the University of Kaiserslautern in cooperation with the WZL, LESP 2 is continued together with the PFAFF company at Kaiserslautern. Both systems put different demands on knowledge representation mechanisms which range, however, within the above-mentioned scope.

3.1. LESP 2–A LEARNING SYSTEM FOR INSPECTION PLAN GENERATION

Inspection planning includes all operations needed to plan tests which are economically necessary to ensure the quality of a product. In quality control the term inspection planning stands for developing inspection plans, planning inspection devices, determining inspection characteristics, test methods, instants of

inspection, test costs, test data processing, and inspection place as well as the demands for staff.

LESP 2 is conceived as an Expert System shell in the area of inspection planning which allows (within a certain range) to produce quite easily specific Expert Systems satisfying the needs of different applications. Therefore its particular task is, in addition to modelling general mechanisms and knowledge, to offer the possibility of integrating specific knowledge and data of a particular company into the system. Access to these data is provided best by a suitable data bank connection. For the integration of domain specific knowledge it is essential that LESP 2 has explicitly represented the adequate structures and mechanisms. Only in this way one can get access to the semantic contents of the problems.

The representation in LESP 2 includes general knowledge about inspection planning, knowledge about the principal way developing a plan (e.g. a simple modelling of time and sequencing conditions), and knowledge about taking advantage of analogous situations. Thus LESP 2 orientates towards the VDI/VDE/DGQ standard 2619 (VDI [14]) about inspection planning. Its target is to structure the various problems, to standardize concepts and to obtain a widely accepted inspection procedure.

Moreover, LESP 2 offers the possibility of integrating specific knowledge structures and domain dependent methods. These are for instance general guidelines for inspection planning, specific instructions for the particular field of production under consideration, the classification of the spectrum of products, the empirical knowledge of the person involved in inspection planning or production scheduling as well as particular company interval methods of taking advantage of similarities and analogies.

The procedure in LESP 2 is subdivided by the cycles of the operation sheet corresponding to the inspection plan. For every edited cycle the system suggest the corresponding inspection steps. If this cycle is accomplished through modification of another one carried out earlier LESP 2 directly generates an inspection step by analogy. Thus the similarity of the cycles is used to reduce the search space of the possible inspection steps. This, however, requires knowledge about the relations between all the cycles. The demands made on LESP 2 with respect to knowledge representation result in the possibility of adequately representing knowledge about inspection planning and the relations of the different cycles by production rules, whereas the structuring of this knowledge is achieved by the use of frames. Thus, e.g. a cycle is interpreted as a frame with knowledge about the relevant quality requirements and characteristics. The knowledge about similarities and the company dependent spectrum of products is modelled in object hierarchies.

It should have become clear that realizing LESP 2 by conventional representation mechanisms would be very time-consuming. Nevertheless LESP 2 is dependent on the integration of the efficiency provided by a (conventional) data base system.

3.2. IDA–A SYSTEM FOR FINDING FUNCTIONS AND SOLUTIONS IN CONSTRUCTION

The purpose of the IDA project is to support in mechanical engineering the stage of design named "conception", in which functions and technical realizations are determined. First a detailed functional description of the object to be designed is generated. This description has to meet certain requirements. Subsequently, on this basis, the system searches for technical realizations for the corresponding subfunctions. In doing so different levels of abstraction have to be considered.

Here the fundamental problem is to restrict the complexity concerning the choice of possible subfunctions and their technical realizations on the different levels of specification by means of appropriate heuristics (use of particular boundary conditions). Moreover, it appears difficult to describe and to handle dependencies among objects of the same or different level of detail.

This problem description already makes the difficulties of an computer aided approach using conventional programming apparent. Due to the possible alternatives of selection and combination and the resulting interdependencies such an approach has presently a very limited chance of success. A further problem originates from the need for continuous updates of the fast growing technical knowledge. Today conventional systems do not supply satisfactory support in this direction. These reason seem to justify an AI approach to problems dealing with planning and configuration on different levels of abstraction.

As in the LESP 2 system the realization is based on a hybrid mode of representation of the relevant design knowledge. This is due to the fact that neither an entirely object-oriented nor an exclusively rule-based representation adequately mirrors the given knowledge structures:

- The object-oriented representation serves to describe functions and technical realizations in terms of their desired properties and restrictions.
- The rule-based control serves to guide the inferential process as well as to represent dependencies among objects on the same or different levels of abstraction. Here the choice of a particular object triggers off the application of a rule. An application of a rule changes the conditions and requirements on certain objects of higher levels of abstraction, i.e. rules modify the context of objects.

The actual state of specification is stored in a dynamic data basis. If a new object is generated, then by means of adequate rules the actual context of all involved objects in the dynamic data basis is modified before another step of specification is started.

It would be fatal to believe that a system as described above could be realized without consideration of its environment in construction. So one has to support interfaces to database systems (containing technical realizations), to calculation programs, and to CAD-systems. Only in combination with these traditional

systems an expert system like IDA can become a powerful tool within the construction process.

4. Prospects

As the second example has illustrated it is not sufficient to consider Expert Systems in complete isolation from other aspects only on an abstract and high level as it has been done in the past. It is often rather necessary to make use of certain fast algorithms, data base operations or conventional O.R. programs. Such a system then requires interfaces between its components.

The speed of the conventional algorithms is due to the fact that a comparatively great number of operations is carried out in a uniform and possibly parallel way. The problems arising from applications usually do not immediately allow to separate such a uniform part. There are two approaches which offer themselves to solve such problems:

1) The algorithm is called by the system to deal with a non-uniform question; the result is usually a severe reduction of the efficiency.
2) The system calls the algorithm to solve a modified and a more uniform problem which can be efficiently solved by this algorithm and is still useful for the main task.

The main problem is to subdivide the original task into parts, such that classical algorithms can be applied as indicated in 2).

Along with an increasing application of Expert Systems in real situations this type of interface questions and the adequate distribution of tasks will become more and more essential. The actual state of the art in this area is, however, still very dissatisfying. Only the connection to data bases shows some initial progress (Spieker [13]). Nevertheless, the future possibilities of applying Expert Systems will decisively be influenced by such developments.

References

[1] K.-D. Althoff, What are expert systems?, in: *Expert Systems in Production Engineering* (Springer Verlag, 1987) 20–34.
[2] R.J. Brachman, R.E. Fikes and H.J. Levesque, KRYPTON: a functional approach to knowledge representation, IEEE Computer 16 (October 1983) 67–73.
[3] B.G. Buchanan, G. Southerland and E. Geirgenbaum, Heuristic DENDRAL: A program for generating explanatory hypotheses in organic chemistry, in: *Machine Intelligence* 5, eds. B. Meltzer and D. Michie (New York: Elsevier. 1969).
[4] W.F. Clocksin and C.S. Mellish, Programming in PROLOG (Springer-Verlag, 1981).
[5] A. Goldberg and D. Robson, *Smalltalk-80: The Language and its Implementation* (Addison Wesley, 1983).

[6] N. Kratz, Entwicklung eines Software moduls zur rechnerunterstützten Funktions- und Lösungsfindung in der Konstruktionsphase Konzipieren, Diplomarbeit RWTH Aachen, 1986.
[7] H. Kauffman and A. Grumbach, Representing and manipulating knowledge within "worlds", in: *Proc. of the First Int. Conf. on Expert Database Systems*, Charleston, South Carolina: April 1986, ed. L. Kerschberg.
[8] J. McDermott, R1: A rule-based configurer of computer systems, Computer Science Department, Carnegie-Mellon University, 1980.
[9] M. Minsky, A framework for representing knowledge, in: *Psychology of Computer Vision*, ed. P.H. Winston (McGraw-Hill, New York, 1975).
[10] M.M. Richter, Expertensysteme und konventionelle Programme–Unterschiede und Kopplungsprobleme, in: *Festschrift "Technologie, Wachstum, Arbeitslosigkeit"*, ed. R. Henn. Zum 50. Geburtstag von Lothar Späth (Springer-Verlag), to appear.
[11] M.M. Richter, Some abstract problems in knowledge representation, in: *Expert Judgement and Expert Systems*, ed. J. Mumpower, NATO ANSI Series (Springer-Verlag) to appear.
[12] E.H. Shortliffe, *MYCIN: Computer-based medical consultations* (American Elsevier/North-Holland, New York, 1976).
[13] P. Spieker, Möglichkeiten der Kopplung von PROLOG und einem Datenbanksystem, ausgeführt am Beispiel der IDM, Diplomarbeit RWTH Aachen, 1985.
[14] VDI/VDE/DGQ-Richtlinie 2619: Prüfplanung, Juli 1982.