A ROBOT PLANNING STRUCTURE USING PRODUCTION RULES

Ralph P. Sobek*

Laboratoire d'Automatique et d'Analyse des Systemes du C.N.R.S.
7, avenue du Colonel-Roche
F-31077 Toulouse CEDEX, FRANCE

Abstract

Robot plan generation is a field which engendered the development of AI languages and rule-based expert systems. Utilization of these latter concepts permits a flexible formalism for robot planning research. We present a robot plan-generation architecture and its application to a real-world mobile robot system. The system undergoes tests through its utilization in the IIILARE robot project (Ciralt, et jal, 1984). Though the article concentrates on planning, execution monitoring and error recovery are discussed. The system includes models of its synergistic environment as well SR of its sensors and effectors (i.e. operators). Its rules embody both planning specific and domair specific knowledge. The system gains generality and adaptiveness through the use of planning variables which provide constraints to the plan generation system. It is implemented in an efficient compiled Production System language (PS1).

INTRODUCTION

In general, a problem is a situation for which an organism (or program) does not have a ready response. Problem solving, involves 1) sensing and identification of a problem, 2) formulation of the problem in workable terms, 3) utilization of relevant information, and 4) generation and evaluation of hypotheses. A planner is a program that attempts to deal with points 2 through 4. In this paper we present a rule-based plan generation system called FPS (for Flexible Planning System). This system undergoes tests in a real-world robotic environment (the HILARE proiect [Giralt, et al, 1984]).

In such an environment what is a plan? It has to be a flexible and extensible structure which permits quick adaptation to unexpected situations. It must permit goa1-directed as well as data-directed processing. Goal simultaneity and interaction must be verified during planning and before attempted execution. In a real-world environment a multitude of error situations may arise. Besides correction of planning errors a planner must try to determine when an error is recoverable or when replanning is necessary. Planners must be able to select processing strategies appropriate to each situation encountered and be able to handle complex goal descriptions.

FPS is rule-based principally for the following reasons. Production System (PS) rules allow for a neat solution to the frame problem when we use the STRIPS assumption (Waldinger, 1981) in that all updates in tbe model *are* done explicitly through rules. Since rules can react in one PS cycle PSs can adapt to new situations very rapidly, the rules acting like deamons. In planning or execution monitoring this fact allows the system to deal with unexpected/serependi tious situations. In addition, rule interactions may permit parallel searchs for a best solution or may allow rapid responses to recognized problem situations (e.g. planning goal conflicts). In a PS the addition of knowledge is incremental. Therefore, the evolution of our robotic environment will be easily characterizable to FPS. Also, in the future our use of *a* PS architecture will permit FPS to organize and generalize the plans that it has created as new rules.

Some may say that PSs are inefficient. It has been shown that by the use of compilatior strategies significant gains in execution speed are attainable (Gupta and Forgy, 1983). FPS is implemented in the PS] production system language (Sobek, 1983). It does not have to sacrifice efficiency for flexibility in its representation since PS] is a compiled PS. Rule patterns are compiled into a parallel-match tree similar to but more general than OPS (Forgy, 1982). The advantages of PSs have been adequately described in (Davis and King, 1976). Some planners and expert systems have opted for a frame-based approach (Minsky, 1975). It should be noted that there is a similarity between PSs and frame-based systems.

PLANNING STRUCTURE

We present a robot planner (FPS) which deals with the dynamics of a plan. FPS has in its ancestry STRIPS (Fikes, et al> 1971), NOAH (Sacerdoti, 1977), and especially JASON (Sobek, 1975). It generalizes these planners in representation and flexibility. FPS is used in a real-world mobile-robot 'blocks-world' paradigm: the HILARE project. Superficially, FPS is similar to NOAH and its generalization JASON in that they are goa 1-ori en ted. Where plans for NOAH consist of a directed graph of procedures (procedural net), in FPS plans may be

The author is presently on visit from the University of California at Berkeley. This research is supported in part by Agence de l'Informatique contract 84/723.

described *PF* a directed graph of *pre cesses.* Each process cortairs its state in *p* structure called a "planning node" each with its associated goal (see Table 1). A process characterizes tbe dynamics of a plan step while the planning node representee tbe data aspects. A process pets its node's entries filled from three principal sources: 1) freir tbe parent node, ?) when an operator is selected for a node, or 3) by tbe executive, critic, task communication, and scheduling rules. The processes are managed by a tasking executive wbicb arranges tbe processes on a priority agenda taking into account for eacb process tbe importance, success, cost expended, and estimated allocated cost. The inter-rrecess coordination and high-level conflict resolution knowledge are called planning specific and are represented in rules. For example:

> If all sibling cbildren of a process have achieved its preconditions then check if tbe process can be decomposed into sub-processes .

1. *Cos]* Pattern
?. Coal Instantiations
3. Freconditions/Enablements
C. Continuation Conditions
5. Post-Conditions (including goal pattern)
f. Constraint Conditions
7. Parent
8. Children associated by each decomposition
9. Importance
10. Allocated Cost Estimate
11. Cost Expended
12. Success Kate for eacb Post-Condition
13. Error Recovery Handles: reason, source, locally recoverable
14. Operator List
15. Script

Table 1. Planning Node Entries

Simple goals in FPS may consist of a relational predicate,^. (INROOM BL0CK1 RM3), its negation, or tbe application of a specific operator to a goal. Compound goals may be conjunctions *or* eeouences of goals. Compound goals let FPS search for possible, conflicts whereas sequences specify an explicit required ordering of the goals involved.

A relational predicate may also contain "planning variables" similar to tbose developed by the author in (Sobek, 1975) and tbose reinvented in STPE (Wilkins, 1963). These variables do not actually contain values; they may specify restrictions upon the allowed values (bindings) that their positions in a predicate may take, e.g. in tbe predicate (INROOM ROBOT $RM) the variable $RM may specify a number of possible instantiations for the predicate. The planning variables serve three

Tbe sibling processes are all children of a process which are conjoined by the same goal instantiation ofthe parent. The parent process might have multiple instantiations for its goal description and then would have a disjunctive group of siblings for each instantiation.

functions: 1) ar rltr.Tnr.tive *to* disjunctive goals with similar disjuncts, 2) a method for the postponement of decisions, and 3) constraint expressions.

Constraints mav be attached to goals, planning nodes, and operators. Constraints *are* similar to goals except that they must be satisfied for the preconditions of a possible operator as well *as* during and after tbe operator's execution. They are taken into account in the node expansion procedure *p.rd* can cause the insertion of additional plan steps before a n o d e.

Plarrirg involves iteratier of *voc*e* expansion with plan criticism. Criticism may start *as* soon *as* a node is expanded, which eases a shortcoming of NOAH. NOAH could only apply its critics at the end of each expansion cycle. The critics in FPS are considered a major part of the planner's "planning executive." They contain knowledge that is relevant to tbe entire planning process, i.e. both planning specific and domain specific. Corcomitant and overlapping with critics are heuristic rules. For example:

> hi. If multiple choices *are* possible then select one which minimizes cost, effort, or distance
>
> H 2. If robot moves an object then it should not block a door

HI is a general rule whereas H? is domain specific.

The planner" presents a model of the robot's possible actions within its environment: it currently does not node! the robot's interactions. There is no representation for other purposive (goal oriented) organisms or causality other than tbe robot's. The possible actions *are* modelled by operators.

Operators are dynamically selected for each planning node; no a priori connection between operators and goals exists. Associated with operators are preconditions (environmental context), continuation conditions, post-conditions, and constraints. The operators are ordered in a specialization/generalization hierarchy. Operators may have scripts which specifv how they should be reduced; the scripts may define conditionals, parallel paths, goals, constraints, and sub-operator applications. If they contain subgoals then a sub-process will be created for each subgoal. Otherwise, the script will be checked by "critics" against tbe surrounding plan structure.

An example operator is GOTOROOM (see Fig. 1). Given a room ?r as argument, if the robot is in an adjoining room then it will try to apply sequentially the two sub-scripts GOTODOOR and GOTHRUDOOR. These latter two scripts are subordinate to GOTOROOM only in the current node's dynamic context: a different call sequence would create another hierarchy of goals and operators. FPS would fan out from the goal state, e.g. (INROOM ROBOT RM]), using the connexity graph provided by CONNECTS until it finds the current state.

```
GOTOROOM:
argument:
preconditions:       (INROOM ROBOT ?r2)
                     (CONNECTS ?d ?r ?r2) +
script:              (SEQ (GOTODOOR ?d)
                          (GOTHRUDOOR ?d))
post-conditions:     (INROOM ROBOT ?r)*
                     NOT (INROOM ROBOT ?r2)

      - primary result (goal condition)
    4 - static data
```

Figure 1. GOTOROOM operator specification

The search could be breadth-first, depth-first or depending on the situation it could even require heuristic rules which would remember efficient routes once found. After a route is found the two sub-ordinate scripts are tried in order to assure that the robot can get to and through the door. If there are multiple doors to a room, each would cause two parallel descendant nodes to be created.

An operator's goal is specified to the system as the primary post-condition. The above operator can also be invoked to get the robot out of a particular room. Each time that an operator succeeds with respect to a goal its correspondent level of importance or competence is rewarded.

EXECUTION MONITOR

What distinguishes planning from execution monitoring is that in the former a coherent planning structure is established, whereas in the latter the necessary verifications of coherence must come from the real-world environment. Note that a large part of the representation for both planning and execution monitoring must be the same in both in order to facilitate their communication and sharing of models. Thus, execution monitoring uses the same planning structures to establish when error recovery should be initiated. An error situation is detected when there is a discrepancy between an operator's expected possible outcomes and the real-world responses (Srinivas, 1977). These discrepancies are analysed by execution-error critics which determine whether the error is 1) unimportant, 2) has a fixed solution, or that 3) replanning will be necessary.

CONCLUSION

FPS combines domain-independent plan structuring critics with domain specific constraints and critics. They watch over a general and flexible plan structure. Since the system is rule-based, heuristics may be added at any level; for the moment few exist. Their usefulness should become apparent when FPS performs error recovery and replanning. Current work includes making the system more robust and the addition of the execution monitor.

Acknowledgements

The author would like to thank everyone involved with the HILARE project for providing the necessary support for a real-world testbed for FPS.

REFERENCES

Davis, R. and J. King. An Overview of Production Systems, in E. W. Elcock and D. Michie (Eds.) Machine Intel1igence, 8 (Wiley, New York, 1976), 300-332.

Fikes, R. et al.- Learning and Executing Generalized Robot Plans, in N. Nilsson and B. Webber (Eds.) Readings in Artificial Intel1igence (Tioga Publishing, Palo Alto, CA, 1981), 231-249.

Forgy, C. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, 19 (1982) 17-37.

Ciralt, G. et al. An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots, in M. Brady and R. Paul (Eds.) Robotics Research: The First International Symposium (MIT Press, Mass.), 191-214.

Gupta, A. and C. Forgy. Measurements on Production Systems, Technical Report CMU-CS-83-167, Carnegie-Mellon University, December 1983.

Minsky, M. Framework for Representing Knowledge, in P. Winston (Ed.) The Psychology of Computer Vision (McGraw-Hill, 1975).

Sacerdoti, E. A Structure for Plans and Behavior, Elsevier, North-Holland, New York, 1977.

Sobek, R. Automatic Generation and Execution of Complex Robot Plans, Master's Project Report, Electrical Engineering and Computer Sciences Dept., University of California, Berkeley (September 1975).

. Achieving Generality and Efficiency in a Production System Architecture. LAAS-CNRS Internal Memo, 1983.

Srinivas, S. Error Recovery in Robot Systems. Ph.D. Thesis, Computer Science Dept., California Institute of Technology (1977).

Waldinger, R. Achieving Several Goals Simultaneously, in N. Nilsson and B. Webber (Eds.) Readings in Artificial Intel1igence (Tioga Publishing, Palo Alto, CA, 1981), 250-271.

Wilkins, D. Representation in a Domain-Independent Planner, IJCAI-83, Karlsruhe, West Germany, 1983, 733-740.