Artificial Intelligence 59 (1993) 227–232
Elsevier

ARTINT 980

STRIPS, a retrospective

Richard E. Fikes and Nils J. Nilsson

Computer Science Department, Stanford University, Stanford, CA 94305, USA

Introduction

During the late 1960s and early 1970s, an enthusiastic group of researchers at the SRI AI Laboratory focused their energies on a single experimental project in which a mobile robot was being developed that could navigate and push objects around in a multi-room environment (Nilsson [11]) . The project team consisted of many people over the years, including Steve Coles, Richard Duda, Richard Fikes, Tom Garvey, Cordell Green, Peter Hart, John Munson, Nils Nilsson, Bert Raphael, Charlie Rosen, and Earl Sacerdoti. The hardware consisted of a mobile cart, about the size of a small refrigerator, with touch-sensitive "feelers", a television camera, and an optical range-finder. The cart was capable of rolling around an environment consisting of large boxes in rooms separated by walls and doorways; it could push the boxes from one place to another in its world. Its suite of programs consisted of those needed for visual scene analysis (it could recognize boxes, doorways, and room corners), for planning (it could plan sequences of actions to achieve goals), and for converting its plans into intermediate-level and low-level actions in its world. When the robot moved, its television camera shook so much that it became affectionately known as "Shakey the Robot".

The robot, the environment, and the tasks performed by the system were quite simple by today's standards, but they were sufficiently paradigmatic to enable initial explorations of many core issues in the development of intelligent autonomous systems. In particular, they provided the context and motivation for development of the A* search algorithm (Hart et al. [7]), the STRIPS (Fikes and Nilsson [4]) and ABSTRIPS (Sacerdoti [14]) planning systems, programs for generalizing and learning macro-operators

Correspondence to: R.E. Fikes, Knowledge System Laboratory, 701 Welch Road, Bldg. C, Palo Alto, CA 94304, USA. E-mail: fikes@ksl.stanford.edu.

0004-3702/93/$ 06.00 © 1993 — Elsevier Science Publishers B.V. All rights reserved

(MACROPS) (Fikes et al. [5]), "triangle tables"for plan execution (Fikes [3]), and region-finding scene analysis programs (Duda and Hart [1]). In this note, we focus on the development of the STRIPS (STanford Research Institute Problem Solver) automatic plan generator and its accompanying plan execution monitor.

The STRIPS automatic plan generator

STRIPS is often cited as providing a seminal framework for attacking the "classical planning problem" in which the world is regarded as being in a static state and is transformable to another static state only by a single agent performing any of a given set of actions. The planning problem is then to find a sequence of agent actions that will transform a given initial world state into any of a set of given goal states. For many years, automatic planning research was focused on that simple state-space problem formulation, and was frequently based on the representation framework and reasoning methods developed in the STRIPS system.

The integration of state-space heuristic search and resolution theorem proving which is the centerpiece of the STRIPS design was consistent with and motivated by the bringing together of our respective backgrounds and interests at the time. Fikes had come to SRI from CMU in 1969 steeped in a GPS-based heuristic problem solving tradition. In addition, he had just completed work on the REF-ARF problem solver (Fikes [2]) which had a multi-level design analogous to that of STRIPS in that it used a symbolic interpreter (of nondeterministic procedures) to build up (constraint-based) state descriptions that were then analyzed by a separate reasoner. Nilsson and his colleagues at SRI had been focused on logic-based representations and use of theorem proving techniques for problem solving. Notably, Cordell Green had just completed development of the QA3 [6] system for his Ph.D. thesis in which the situation calculus representation developed by McCarthy and Hayes [10] and a resolution theorem prover was used to do (among other things) automatic planning for the SRI robot domain.

Green's QA3 work focused our attention on the difficulties of describing in a formal logic the effects of an action, and particularly the difficulty of specifying those aspects of a situation that are not affected by an action (i.e., the "frame problem"). There were no default logic or circumscription theories to appeal to, and we did not invent them. Frustrated by these "technical" difficulties and driven by the pragmatic objective of developing an effective plan generator, we began considering "ad hoc" representations for robot actions and algorithms for modeling their effects, while still maintaining our logic-based representation of individual states. Those considerations produced what is arguably the key technical contribution of the STRIPS

work, namely the STRIPS operator representation and the algorithm for modeling the effects of an operator based on the "STRIPS assumption" that a plan operator affects only those aspects of the world explicitly mentioned in the operator's deletions and additions lists.

Given that we had an effective means of representing robot actions and their effects, we were then faced with the task of designing a plan generator. Using GPS as our paradigmatic problem solving architecture, we needed to define meaningful "differences" between a situation described by a set of predicate calculus sentences and a goal situation in which a given predicate calculus sentence is true. Once differences were defined, we needed to specify what it meant for an operator to be "relevant" to "reducing" the difference.

Since we were using our resolution theorem prover to determine whether a goal was true in a given state, we were faced with the problem of extracting from the theorem prover's failed proof attempts "differences" between the given state and one in which the goal is true. We noted that what was needed to complete the proof were operators which would assert clauses that resolve with the clauses at the leaf nodes of the proof tree. Thus, we could use pattern matching techniques to find operators whose additions lists were relevant to reducing the difference between incomplete and complete proofs without the need for an explicit difference table.[1] This technique, which was another key technical idea in the design, was essentially the same as that used by backward chaining production rule interpreters developed many years later. In our case, each operator corresponded to a rule of the form

> if ⟨ preconditions ⟩
> then (retract ⟨ deletions ⟩) (assert ⟨ additions ⟩).

In retrospect, STRIPS was extremely limited in both the scope of planning issues it addressed and the complexity of problems it could solve. Whereas current planning research is concerned with multiple agents operating in dynamic environments, STRIPS assumed that only one action could occur at any time, that nothing changed except as a result of the planned actions, and that actions were effectively instantaneous. Also, the STRIPS "solution" to the frame problem was vague and flawed. It was many years before the ideas were made precise and a satisfactory formal semantics developed (see Lifschitz [9]). Even with those limitations, the STRIPS representation and reasoning framework was used as the basis for most automatic planning research for many years. Perhaps the severe simplifying assumptions of

[1]Note that the technique for finding relevant operators of matching goals to add lists is only a heuristic since the delete lists of operators are ignored. If an operator is selected to complete an incomplete proof and it deletes a clause used in the incomplete proof, then the proof attempt may fail after application of the operator because of the missing clause.

the STRIPS framework were needed in order to enable early progress to be made on the extreme difficulties of the general automatic planning problem. The STRIPS framework had sufficient intuitive appeal to most researchers for them to believe that it was a viable foundation on which to develop techniques that would be effective in more realistic models. For example, techniques for abstract planning were developed as direct extensions to STRIPS (Sacerdoti [14]), and even some frameworks for planning in dynamic worlds were formulated as extensions to the static STRIPS world (e.g., Hendrix [8] and Pednault [13]).

The STRIPS execution monitor

Given that STRIPS had produced a plan to achieve a goal state, the system was then confronted with the problem that when Shakey executed the plan in a real environment, the resulting state often differed from the state expected by the planner. The differences might result from inadequacies in the robot's world model, effectors, or both. Thus, some kind of "execution monitor" was needed to determine after execution of each plan step whether the plan was "on track" or whether replanning was needed. The problem becomes interesting when one realizes that an exact match between actual and planned states will rarely be achieved and is not needed, since most aspects of a given situation will be irrelevant to the success of a plan. Thus, the challenge is to determine the requirements that must be satisfied by the state produced after each plan step in order for the remainder of the plan to succeed.

Execution monitoring was a relatively unexplored problem at the time this work was done because most AI problem solving research until then had been focused on domains where actions are assumed to produce the results described in their models (e.g., chess, puzzles, theorem proving). Because we were working with an actual rather than a simulated robot, we were led to consider this important additional aspect of problem solving in the physical world.

There were basically two notable technical ideas in the STRIPS execution monitor. The first was a simple algorithm for computing a "kernel" set of sentences which must be true after each plan step in order for the remainder of the plan to succeed. The algorithm regressed plan goals and operator preconditions back through the plan to the states where they were expected to become true and then included them in the kernel of that state and all states through which they had been regressed.

The second notable technical idea in the execution monitor was the monitoring algorithm based on the "triangle table" plan representation. Given that a plan's kernels have been computed, a simple execution monitor

might check whether the appropriate kernel is true after each plan step, execute the next plan step if it is, and initiate replanning if it is not. However, that simple algorithm is deficient in two important ways: it does not recognize situations in which the next plan step is unnecessary because the kernel sentences it is intended to achieve are already serendipitously true, nor does it recognize situations in which redoing some portion of the existing plan is a viable response to the expected kernel not being true. We designed a monitoring algorithm which overcame those deficiencies by asking after executing each plan step whether the goal (i.e., the final kernel) is true, then whether the kernel preceding the last plan step is true, then whether the kernel preceding the next to last plan step is true, etc. When a true kernel was found, the testing stopped and the appropriate action was taken; if no kernels were true, then replanning was initiated. The algorithm did not execute unnecessary steps, retried steps that had failed, and was non-redundant in that it checked each kernel sentence only once as it searched for true kernels.

The STRIPS execution monitor is a simple form of a goal-directed program interpreter in which the goal to be achieved by each program step is explicitly represented and monitored by the interpreter. Some current planning systems embed the monitoring algorithm in the plans they produce in the form of conditional actions that test the kernel sentences (e.g. Nilsson's "teleo-reactive" programs [12]). Given such embedding, execution monitoring occurs as a side-effect of simply executing the plan as a conventional program.

The STRIPS monitor's preference at each step for reusing portions of the existing plan or even to continue executing the existing plan rather than replanning is a heuristic choice except in the rare case where the states being produced during execution are identical to those anticipated by the planner. The choice is heuristic because if an actual state differs from the anticipated one, the planner may be able to produce a plan to achieve the goal from that state which is more efficient than the original plan. The preference for using the existing plan is based on an assumption that replanning is expensive.

Finally, note that the execution monitor takes as given the sensor feedback from the robot after each action is executed. It does not consider, for example, the possibility of sensor error when a kernel sentence is false nor does it consider obtaining additional sensor data when the truth value of a kernel sentence cannot be determined.

References

[1] R.O. Duda and P.E. Hart, Experiments in scene analysis, Tech. Note 20, Artificial Intelligence Center, SRI International, Menlo Park, CA (1970).

[2] R.E. Fikes, REF-ARF: a system for solving problems stated as procedures, *Artif. Intell.* **1** (1) (1970) 27–120.
[3] R.E. Fikes, Monitored execution of robot plans produced by STRIPS, in: *Proceedings IFIP Congress 71*, Ljubljana, Yugoslavia (1971).
[4] R.E. Fikes and N.J. Nilsson, STRIPS: a new approach to the application of theorem proving to problem solving, *Artif. Intell.* **2** (1981) 189–208.
[5] R.E. Fikes, P.E. Hart and N.J. Nilsson, Learning and executing generalized robot plans, *Artif. Intell.* **3** (4) (1972) 251–288.
[6] C.C. Green, Application of theorem proving to problem solving, in: *Proceedings IJCAI-69*, Washington, DC (1969) 219–239.
[7] P.E. Hart, N.J. Nilsson and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Syst. Sci. Cybern.* **4** (2) (1968) 100–107.
[8] G. Hendrix, Modelling simultaneous actions and continuous processes, *Artif. Intell.* **4** (1973) 145–180.
[9] V. Lifschitz, On the semantics of STRIPS, in: M.P. Georgeff and A. Lansky, eds., *Reasoning about Actions and Plans* (Morgan Kaufmann, San Mateo, CA, 1987).
[10] J. McCarthy and P.J. Hayes, Some philosophical problems from the standpoint of artificial intelligence, in: B. Meltzer and D. Michie, eds., *Machine Intelligence* **4** (Edinburgh University Press, Edinburgh, 1969) 463–502.
[11] N.J. Nilsson, Shakey the Robot, SRI Tech. Note 323, Menlo Park, CA (1984).
[12] N.J. Nilsson, Toward agent programs with circuit semantics, Robotics Laboratory Tech. Note, Computer Science Department, Stanford University, Stanford, CA (1992).
[13] E. Pednault, Formulating multi agent dynamic world problems in the classical planning framework, in: M.P. Georgeff and A. Lansky, eds. *Reasoning about Actions and Plans* (Morgan Kaufmann, Los Altos, CA, 1987) 42–82.
[14] E. Sacerdoti, Planning in a hierarchy of abstraction spaces, *Artif. Intell.* **5** (2) (1975) 115–135.