12

Mobile Robots

THE HAND–EYE SYSTEMS DESCRIBED EARLIER MIGHT BE THOUGHT OF AS "ROBOTS," but they could not move about from their fixed base. Up to this time, very little work had been done on mobile robots even though they figured prominently in science fiction. I have already mentioned Grey Walter's "tortoises," which were early versions of autonomous mobile robots. In the early 1960s researchers at the Johns Hopkins University Applied Physics Laboratory built a mobile robot they called "The Beast." (See Fig. 12.1.) Controlled by on-board electronics and guided by sonar sensors, photocells, and a "wallplate-feeling" arm, it could wander the white-walled corridors looking for dark-colored power plugs. Upon finding one, and if its batteries were low, it would plug itself in and recharge its batteries. The system is described in a book by Hans Moravec.[1]

Beginning in the mid-1960s, several groups began working on mobile robots. These included the AI Labs at SRI and at Stanford. I'll begin with an extended description of the SRI robot project for it provided the stimulus for the invention and integration of several important AI technologies.

12.1 Shakey, the SRI Robot

In November 1963, Charles Rosen, the leader of neural-network research at SRI, wrote a memo in which he proposed development of a mobile "automaton" that would combine the pattern-recognition and memory capabilities of neural networks with higher level AI programs – such as were being developed at MIT, Stanford, CMU, and elsewhere. Rosen had previously attended a summer course at UCLA on LISP given by Bertram Raphael, who was finishing his Ph.D. (on SIR) at MIT.

Rosen and I and others in his group immediately began thinking about mobile robots. We also enlisted Marvin Minsky as a consultant to help us. Minsky spent two weeks at SRI during August 1964. We made the first of many trips to the ARPA office (in the Pentagon at that time) to generate interest in supporting mobile robot research at SRI. We also talked with Ruth Davis, the director of the Department of Defense Research and Engineering (DDR&E) – the office in charge of all Defense Department research. We wrote a proposal in April 1964 to DDR&E for "Research in Intelligent Automata (Phase I)" that would, we claimed, "ultimately lead to the development of machines that will perform tasks that are presently considered to require human intelligence."[2] The proposal, along with several trips and discussions culminated, in November 1964, in a "work statement" issued by the then-director of

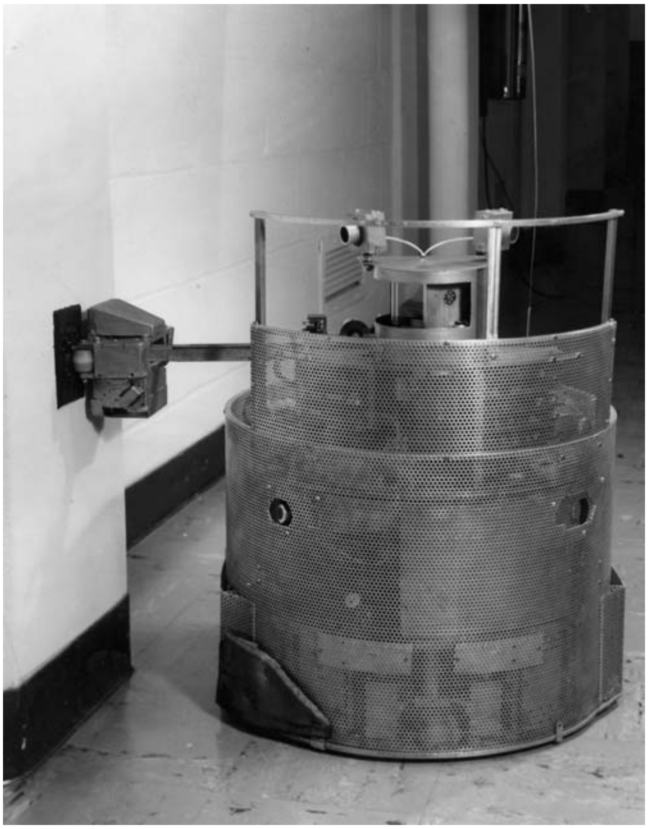
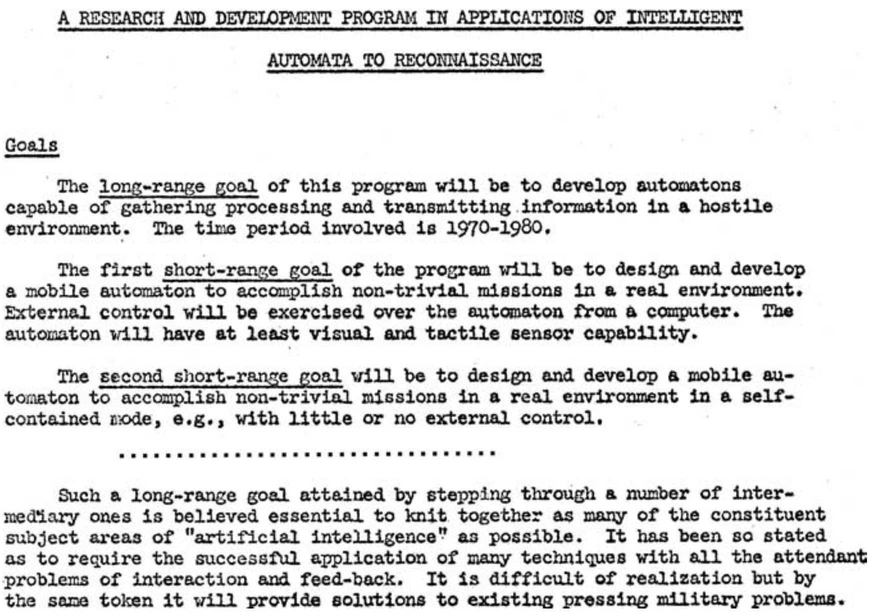


Figure 12.1. The Johns Hopkins "Beast." (Courtesy of Johns Hopkins University Applied Physics Laboratory.)

ARPA's Information Processing Techniques Office, Ivan Sutherland. The excerpt in Fig. 12.2 describes the goals of the program.[3]

In the meantime, Bertram Raphael completed his MIT Ph.D. degree in 1964 and took up a position at UC Berkeley for an academic year. In April 1965, he accepted our offer to join SRI to provide our group with needed AI expertise. After several research proposal drafts and discussions with people in the relevant offices in the Defense Department (complicated by the fact that Ivan Sutherland left ARPA during this time), SRI was finally awarded a rather large (for the time) contract based essentially on Sutherland's work statement. The "start-work" date on the project, which was administered for ARPA by the Rome Air Development Center (RADC) in Rome, New York, was March 17, 1966. (Coincidentally, just before joining SRI in 1961, I had just finished a three-year stint of duty as an Air Force Lieutenant at RADC working on statistical signal-processing techniques for radar systems.) Ruth Davis played a prominent role in getting ARPA and RADC to move forward on getting the project started. The "knitting together" of several disparate AI technologies was one of the primary challenges and one of the major contributions of SRI's automaton project.[4]


A RESEARCH AND DEVELOPMENT PROGRAM IN APPLICATIONS OF INTELLIGENT AUTOMATA TO RECONNAISSANCE

Goals

The long-range goal of this program will be to develop automatons capable of gathering processing and transmitting information in a hostile environment. The time period involved is 1970-1980.

The first short-range goal of the program will be to design and develop a mobile automaton to accomplish non-trivial missions in a real environment. External control will be exercised over the automaton from a computer. The automaton will have at least visual and tactile sensor capability.

The second short-range goal will be to design and develop a mobile automaton to accomplish non-trivial missions in a real environment in a self-contained mode, e.g., with little or no external control.

....

Such a long-range goal attained by stepping through a number of intermediary ones is believed essential to knit together as many of the constituent subject areas of "artificial intelligence" as possible. It has been so stated as to require the successful application of many techniques with all the attendant problems of interaction and feed-back. It is difficult of realization but by the same token it will provide solutions to existing pressing military problems.

Figure 12.2. Excerpt from the typescript of the automaton work statement.

One of the tasks was the actual construction of a robot vehicle whose activities would be controlled by a suite of programs. Because of various engineering idiosyncrasies, the vehicle shook when it came to an abrupt stop. We soon called it "Shakey," even though one of the researchers thought that sobriquet too disrespectful. [Shakey was inducted into the "Robot Hall of Fame" (along with C-3PO among others) in 2004.[5] It was also named as the fifth-best robot ever (out of 50) by *Wired Magazine* in January 2006. *Wired*'s numbers 2 and 4 were fictional, "Spirit" and "Opportunity" (the Mars robots) were number 3, and "Stanley" (winner of the 2005 DARPA "Grand Challenge") was named "the #1 Robot of All Time." Shakey is now exhibited at the Computer History Museum in Mountain View, California.][6]

Shakey had an on-board television camera for capturing images of its environment, a laser range finder (triangulating, not time-of-flight) for sensing its distance from walls and other objects, and cat-whisker-like bump detectors. Shakey's environment was a collection of "rooms" connected by doorways but otherwise separated by low walls that we could conveniently see over but Shakey could not. Some of the rooms contained large objects, as shown in Fig. 12.3. The size of Shakey can be discerned from inspection of Fig. 12.4.

Most of the programs that we developed to control Shakey were run on a DEC PDP-10 computer. Between the PDP-10 and the mobile vehicle itself were a PDP-15 peripheral computer (for handling the lower level communications and commands to on-board hardware) and a two-way radio and video link. The PDP-10 programs were organized in what we called a "three-layer" hierarchy. Programs in the lowest level drove all of the motors and captured sensory information. Programs in the

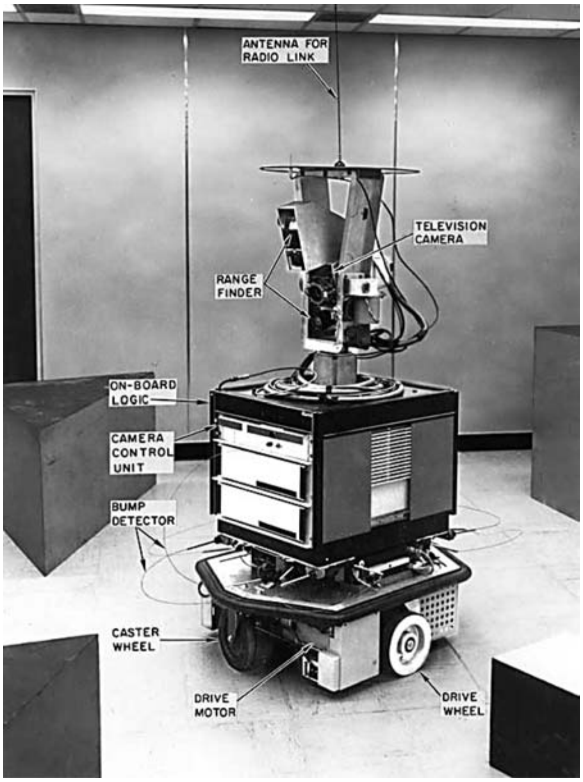


Figure 12.3. Shakey as it existed in November 1968 (with some of its components labeled). (Photograph courtesy of SRI International.)

intermediate level supervised primitive actions, such as moving to a designated position, and also processed visual images from Shakey's TV camera. Planning more complex actions, requiring the execution of a sequence of intermediate-level actions, was done by programs in the highest level of the hierarchy. The Shakey project involved the integration of several new inventions in search techniques, in robust control of actions, in planning and learning, and in vision. Many of these ideas are widely used today. The next few subsections describe them.

12.1.1 *A*: A New Heuristic Search Method*

One of the first problems we considered was how to plan a sequence of "way points" that Shakey could use in navigating from place to place. In getting around a single obstacle lying between its initial position and a goal position, Shakey should first head toward a point near an occluding boundary of the obstacle and then head straight for the unobstructed final goal point. However, the situation becomes more complicated if the environment is littered with several obstacles, and we sought a general solution to this more difficult problem.

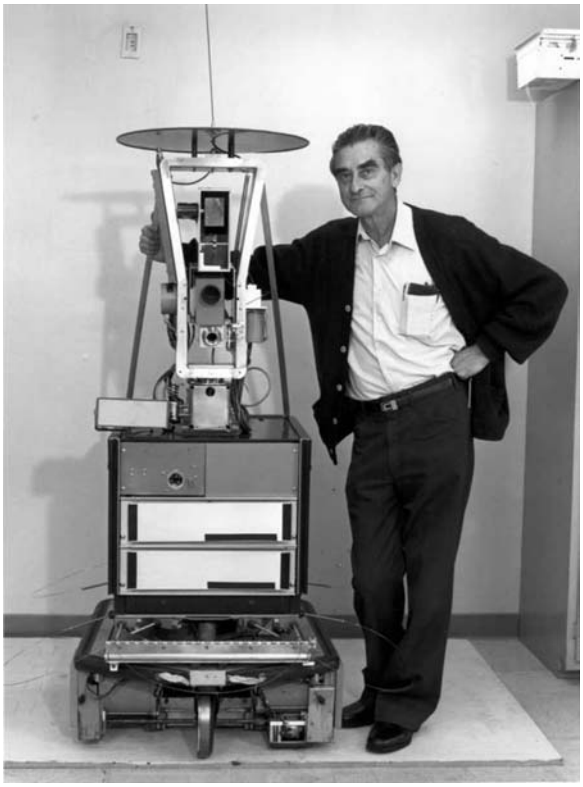


Figure 12.4. Charles A. Rosen with Shakey. (Photograph courtesy of SRI International.)

Shakey kept information about the location of obstacles and about its own position in a "grid model," such as the one shown in Fig. 12.5. (To obtain the required accuracy, grid cells were decomposed into smaller cells near the objects. I think this was one of the first applications of adaptive cell decomposition in robot motion planning and is now a commonly used technique.) Consider, for example, the navigation problem in which Shakey is at position R and needs to travel to G (where R and G are indicated by the shaded squares). It can use a computer representation of the grid model to plan a route before beginning its journey – but how? The map shows the positions of three objects that must be avoided. It is not too difficult to compute the locations of some candidate way points near the corners of the objects. (These way points must be sufficiently far from the corners so that Shakey wouldn't bump into the objects.) The way points are indicated by shaded stars and labeled "A," "B," and so on through "K." Using techniques now familiar in computer graphics, it also is not difficult to compute which way points are reachable using an obstacle-free, straight-line path from any other way point and from R and G.

Looked at in this way, Shakey's navigation problem is a search problem, similar to ones I have mentioned earlier. Here is how a search tree can be constructed and then searched for a shortest path from R to G. First, because A and F are directly

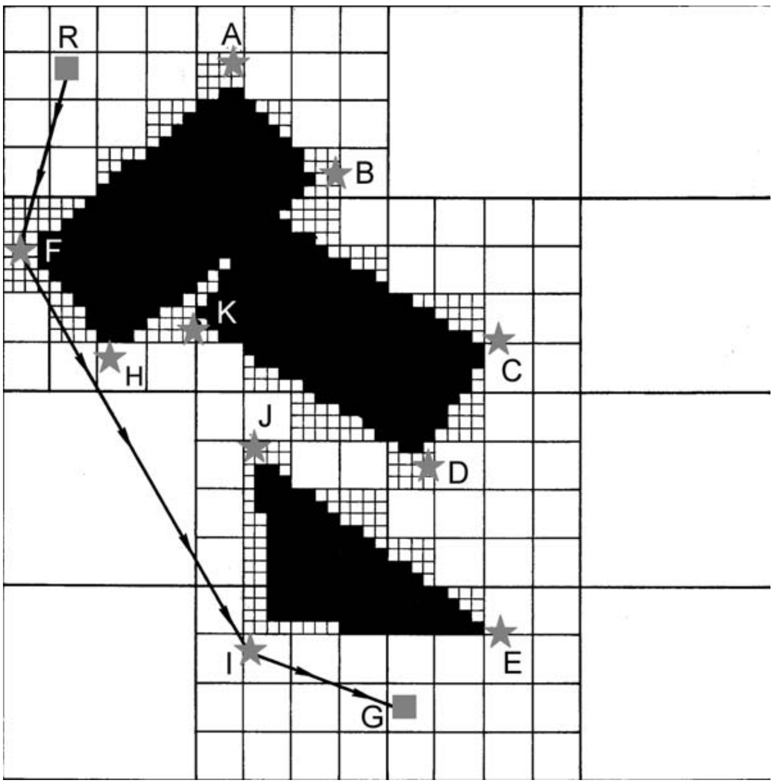


Figure 12.5. A navigation problem for Shakey. (Illustration used with permission of SRI International.)

reachable by obstacle-free, straight-line paths from R, these are set up as direct-descendant "nodes" of R in the search tree. We continue the process of computing descendant nodes (along obstacle-free, straight-line paths) from each of A and F and so on until G is added to the tree. Then, it is a simple matter to identify the shortest path from R to G.

Several methods for searching trees (and their more general cousins, graphs) were already in use by the mid-1960s. One point in favor of these known methods was that they were guaranteed to find shortest paths when used to solve Shakey's navigation problems. However, they could be computationally inefficient for difficult problems. Of course, solving simple navigation problems (such as the one in the diagram) does not involve much search, so any search method would solve such problems quickly. But we were interested in general methods that would work efficiently on larger, more difficult problems. I was familiar with the heuristic search method proposed by J. Doran and Donald Michie for solving the eight-piece, sliding-tile puzzle. They assigned a numerical value to each node in the search tree, based on the estimated difficulty of reaching the goal from that node. The node with the lowest score was the one that was selected next to have its descendants generated.[7]

I reasoned that a good "heuristic" estimate for the difficulty of getting from a way point position to the goal (before actually searching further) would be the "airline distance," ignoring any intervening obstacles, from that position to the goal. I suggested that we use that estimate as the score of the corresponding node in the search tree. Bertram Raphael, who was directing work on Shakey at that time, observed that a better value for the score would be the sum of the distance traveled so far from the initial position plus my heuristic estimate of how far the robot had to go.[8]

Raphael and I described this idea to Peter Hart, who had recently obtained a Ph.D. from Stanford and joined our group at SRI. Hart recalls[9] "going home that day, sitting in a particular chair and staring at the wall for more than an hour, and concluding" that if the estimate of remaining distance (whatever it might be) was never larger than the actual remaining distance, then the use of such an estimate in our new scoring scheme would *always* find a path having the shortest distance to the goal. (Of course, my heuristic airline distance satisfied Hart's more-general condition.) Furthermore, he thought such a procedure would generate search trees no larger than any other procedures that were also guaranteed to find shortest paths and that used heuristic estimates no better than ours.

Together, Hart, Raphael, and I were able to construct proofs for these claims, and we named the resulting search process "A*." (The "A" was for algorithm and the "*" denoted its special property of finding shortest paths. I think Hart and Raphael did most of the heavy lifting in devising the proofs.) When paths have costs associated with them that depend on more than just distance, and when such costs (rather than distances) are taken into account in computing scores, A* is guaranteed to find lowest cost paths.[10]

The inclusion of the estimate of remaining distance (or cost) to the goal contributes to searching in the general direction of the goal. The inclusion of the actual distance (or cost) incurred so far ensures that the search process will not forever be led down promising but perhaps futile paths and will be able to "leak around" obstacles.

A* has been extended in many ways – especially by Richard Korf to make it more practical when computer memory is limited.[11] Today, A* is used in many applications including natural language parsing,[12] the computation of driving directions,[13] and interactive computer games.[14]

12.1.2 *Robust Action Execution*

The A* algorithm was embedded in Shakey's programs for navigating from one place to another within a room containing obstacles and for pushing an object from one place to another. Navigation programs, along with others, occupied the middle level of the hierarchy of Shakey's programs. These intermediate-level programs were all designed to achieve certain goals, such as getting an object in front of a doorway for example. They were also quite robust in that they "kept trying" even in the face of unforeseen difficulties. For example, if an object being pushed happened accidentally to slip off the front "pushing bar," the push program noticed this problem (through built-in contact sensors in the pushing bar) and repositioned Shakey so that it could reengage the object and continue pushing.

In thinking about how to achieve this robustness, I was inspired both by Miller, Galanter, and Pribram's TOTE units and by the idea of homeostasis. (Recall that a TOTE unit for driving in a nail keeps pounding until the nail is completely driven in and that homeostatic systems take actions to return them to stability in the face of perceived environmental disturbances.) I wanted the mid-level programs to seek and execute that action that was both "closest" to achieving their goals and that could actually be executed in the current situation. If execution of that action produced a situation in which, as anticipated, an action even closer to achieving the goal could be executed, fine; the mid-level program was at least making progress. If not, or something unexpected caused a setback, some other action would be executed next to get back on track. Richard Duda and I developed a format, called "Markov tables," for writing these intermediate-level programs having this "keep-trying" property.[15]

12.1.3 *STRIPS: A New Planning Method*

The mid-level programs could accomplish a number of simple tasks, such as getting Shakey from one place to another in the same room, pushing objects, and getting Shakey through a doorway into an adjoining room. However, to go to some distant room and push an object there into some designated position would require joining together a sequence of perhaps several of these mid-level programs. Just as humans sometimes make and then execute plans for accomplishing their tasks, we wanted Shakey to be able to assemble a plan of actions and then to execute the plan. The plan would consist of a list of the programs to be executed.

Information needed for planning was stored in what was called an "axiom model." This model contained logical statements in the language of the predicate calculus (which I talked about earlier.) For example, Shakey's location was represented by a statement such as `AT(ROBOT, 7,5)`, the fact that Box1 was pushable was represented by the statement `PUSHABLE(BOX1)`, and the fact that there was a doorway named D1 between rooms R1 and R2 was represented by the statement `JOINSROOMS(D1, R1,R2)`. The axiom model had close to two-hundred statements such as these and was the basis of Shakey's reasoning and planning abilities.

Our first attempt at constructing plans for Shakey used the QA3 deduction system and the situation calculus. We would ask QA3 to prove (using a version of the axiom model) that there existed a situation in which Shakey's goal (for example, being in some distant room) was true. The result of the deduction (if successful) would name that situation in terms of a list of mid-level actions to be executed.[16]

The use of the situation calculus for planning how to assemble mid-level actions involved using logical statements to describe the effects of these actions on situations. Not only did we have to describe how a mid-level action changed certain things about the world, but we also had to state that it left many things unaffected. For example, when Shakey pushed an object, the position of that object in the resulting situation was changed, but the positions of all other objects were not. That most things in Shakey's world did not change had to be explicitly represented as logical statements and, worse, reasoned about by QA3. This difficulty, called the "frame problem," has been the subject of a great deal of research in AI, and there have been many attempts to mitigate it, if not solve it.[17] Because of the frame problem, QA3 could be used only for putting together the simplest two- or three-step plans. Any attempt to generate plans very much longer would exhaust the computer's memory.

The problem with the situation calculus (as it was used then) was that it assumed that all things might change unless it was explicitly stated that they did not change. I reasoned that a better convention would be to assume that all things remained unchanged unless it was explicitly stated that they did change. To employ a convention like that, I proposed a different way of updating the collection of logical statements describing a situation. The idea was that certain facts, specifically those that held before executing the action but might not hold after, should be deleted and certain new facts, namely, those caused by executing the action, should be added. All other facts (those not slated for deletion) should simply be copied over into the collection describing the new situation. Besides describing the *effects* of an action in this way, each action description would have a *precondition*, that is, a statement of what had to be true of a situation to be able to execute the action in that situation. (A year or so earlier, Carl Hewitt, a Ph.D. student at MIT, was developing a robot programming language called PLANNER that had mechanisms for similar kinds of updates.)[18]

For example, to describe the effects of the program `goto((X1,Y1),(X2,Y2))` for moving Shakey from some position (X1,Y1) to some position (X2,Y2), one should delete the logical statement `AT(ROBOT, X1,Y1)`, add the statement `AT(ROBOT, X2,Y2)`, and keep all of the other statements. Of course, to execute `goto((X1,Y1), (X2,Y2))`, Shakey would already have to be at position (X1,Y1); that is, the axiom model had to contain the precondition statement `AT(ROBOT,X1,Y1)`, or at least contain statements from which `AT(ROBOT,X1,Y1)` could be proved.

Around this time (1969), Richard Fikes (1942–) had just completed his Ph.D. work under Allen Newell at Carnegie and joined our group at SRI. Fikes's dissertation explored some new ways to solve problems using procedures rather than using logic as in QA3. Fikes and I worked together on designing a planning system that used preconditions, delete lists, and add lists (all expressed as logical statements) to describe actions. Fikes suggested that in performing a search for a goal-satisfying sequence of actions, the system should use the "means–ends" analysis heuristic central to Newell, Shaw, and Simon's General Problem Solver (GPS). Using means–ends analysis, search would begin by identifying those actions whose add lists contained statements that helped to establish the goal condition. The preconditions of those actions would be set up as subgoals, and this backward reasoning process would continue until a sequence of actions was finally found that transformed the initial situation into one satisfying the goal.

By 1970 or so, Fikes had finished programming (in LISP) our new planning system. We called it STRIPS, an acronym for Stanford Research Institute Problem Solver.[19] After its completion, STRIPS replaced QA3 as Shakey's system for generating plans of action. Typical plans consisting of six or so mid-level actions could be generated on the PDP-10 in around two minutes.

The STRIPS planning system itself has given way to more efficient AI planners, but many of them still describe actions in terms of what are called "STRIPS operators" (sometimes "STRIPS rules") consisting of preconditions, delete lists, and add lists.

12.1.4 *Learning and Executing Plans*

It's one thing to make a plan and quite another to execute it properly. Also, we wanted to be able to save the plans already made by STRIPS for possible future use. We were able to come up with a structure, called a "triangle table," for representing plans that was useful not only for executing plans but also for saving them. (John Munson originally suggested grouping the conditions and effects of robot actions in a triangular table. Around 1970, Munson, Richard Fikes, Peter Hart, and I developed the triangle table formalism to represent plans consisting of STRIPS operators.) The triangle table tabulated the preconditions and effects of each action in the plan so that it could keep track of whether or not the plan was being executed properly.

Actions in the plans generated by STRIPS had specific values for their parameters. For example, if some `goto` action was part of a plan, actual place coordinates were used to name the place that Shakey was to go from and the place it was to go to, perhaps `goto((3,7),(8,14))`. Although we might want to save a plan that had that specific `goto` as a component, a more generally applicable plan would have a `goto` component with nonspecific parameters that could be replaced by specific ones depending on the specific goal. That is, we would want to generalize something like `goto((3,7),(8,14))`, for example, to `goto((x1,y1),(x2,y2))`. One can't willy-nilly replace constants by variables, but one must make sure that any such generalizations result in viable and executable plans for all values of the variables. We were able to come up with a procedure that produced correct generalizations, and it was these generalized plans that were represented in the triangle table.

After a plan was generated, generalized, and represented in the triangle table, Shakey's overall executive program, called "PLANEX," supervised its execution.[20] In the environment in which Shakey operated, plan execution would sometimes falter, but PLANEX, using the triangle table, could decide how to get Shakey back on the track toward the original goal. PLANEX gave the same sort of "keep-trying" robustness to plan execution that the Markov tables gave to executing mid-level actions.

12.1.5 *Shakey's Vision Routines*

Shakey's environment consisted of the floor it moved about on, the walls bounding its rooms, doorways between the rooms, and large rectilinear objects on the floor in some of the rooms. We made every effort to make "seeing" easy for Shakey. A dark baseboard separated the light-colored floor from the light-colored walls. The objects were painted various shades of red, which appeared dark to the vidicon camera and light to the infrared laser range finder. Even so, visual processing still presented challenging problems.

Rather than attempt complete analyses of visual scenes, our work concentrated on using vision to acquire specific information that Shakey needed to perform its tasks. This information included Shakey's location and the presence and locations of objects – the sort of information that was required by the mid-level actions. The visual routines designed to gather that information were embedded in the programs

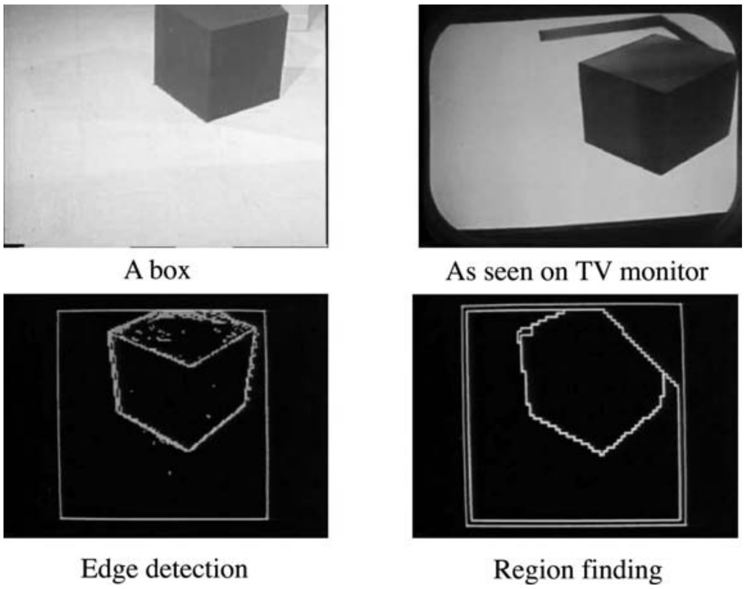


Figure 12.6. Using vision to locate an object. (From the film *Shakey: An Experiment in Robot Planning and Learning*. Used with permission of SRI International.)

for performing those actions. Known properties of Shakey's environment were exploited in these routines.

Exploiting the fact that the objects, the floor, and the wall contained planes of rather constant illumination, Claude Brice and Claude Fennema in our group developed image-processing routines that identified regions of uniform intensity in an image.[21] Because the illumination on a single plane, say the face of an object, might change gradually over the region, the region-finding routine first identified rather small regions. These were then merged across region boundaries in the image if the intensity change across the boundary was not too great. Eventually, the image would be partitioned into a number of large regions that did a reasonable job of representing the planes in the scene. The boundaries of these regions could then be fitted with straight-line segments.

Another vision routine was able to identify straight-line segments in the image directly. Richard Duda and Peter Hart developed a method for doing this based on a modern form of the "Hough transform."[22] After edge-detection processing had identified the locations and directions of small line segments, the Hough transform was used to construct those longer lines that were statistically the most likely, given the small line segments as evidence.

Both region finding and line detection were used in various of the vision routines for the mid-level actions. One of these routines, called `obloc`, was used to refine the location of an object whose location was known only roughly. The pictures in Fig. 12.6 show a box, how it appears as a TV image from Shakey's camera, and two of the stages of `obloc`'s processing. From the regions corresponding to the box and the

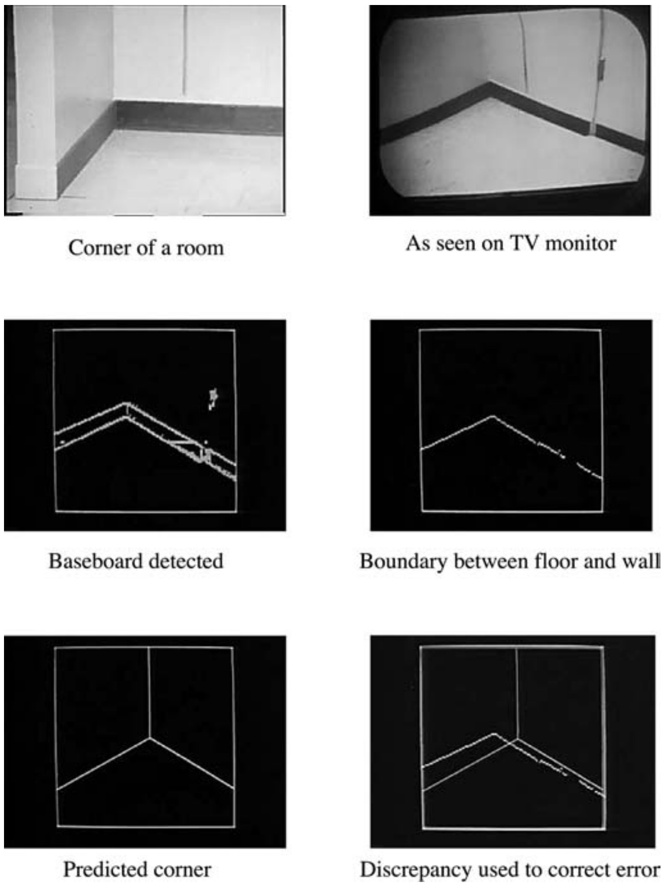


Figure 12.7. Using vision to update position. (From the film *Shakey: An Experiment in Robot Planning and Learning*. Used with permission of SRI International.)

floor (and using the fact that Shakey is on the same floor as the box), straightforward geometric computations could add the box and its location to Shakey's models.

Shakey ordinarily kept track of its location by dead-reckoning (counting wheel revolutions), but this estimate gradually accumulated errors. When Shakey determined that it should update its location, it used another vision routine, called `picloc`. A nearby "landmark," such as the corner of a room, was used to update Shakey's position with respect to the landmark. The pictures in Fig. 12.7 show how `obloc` traces out the baseboard and finds the regions corresponding to the walls and the floor. The final picture shows the discrepancy between Shakey's predicted location of the corner (based on Shakey's estimate of its own location) and the actual location based on `picloc`. This discrepancy was used to correct Shakey's estimate of its position.

Before Shakey began a straight-line motion in a room where the presence of obstacles might not be known, it used a routine called `clearpath` to determine whether its path was clear. This routine checked the image of its path on the floor (a trapezoidal-shaped region) for changes in brightness that might indicate the presence of an obstacle.

In appraising Shakey's visual performance, it is important to point out that it was really quite primitive and subject to many errors – even in Shakey's specially designed environment. As one report acknowledges, "Regions that we wish to keep distinct – such as two walls meeting at a corner – are frequently merged, and fragments of meaningful regions that should be merged are too often kept distinct." Regarding `clearpath`, for example, this same report notes that ". . . shadows and reflections can still cause false alarms, and the only solution to some of these problems is to do more thorough scene analysis."[23] Nevertheless, vision played an important part in Shakey's overall performance, and many of the visual processing techniques developed during the Shakey project are still used (with subsequent improvements) today.[24]

12.1.6 *Some Experiments with Shakey*

To illustrate Shakey's planning and plan-execution and learning methods in action, we set up a task in which Shakey was to push a specified box in front of a specified doorway in a nonadjacent room. To do so, Shakey had to use STRIPS to make a plan to travel to that room and then to push the box. Before beginning its execution of the plan, Shakey saved it in the generalized form described earlier. In the process of executing the plan, we arranged for Shakey to encounter an unexpected obstacle. Illustrating its robust plan execution procedure, Shakey was able to find a different version of the generalized plan that would take it on a somewhat different route to the target room where it could carry on.[25]

One of the researchers working on the Shakey project was L. Stephen Coles (1941–), who had recently obtained a Ph.D. degree under Herb Simon at Carnegie Mellon University working on natural language processing. Coles wanted to give Shakey tasks stated in English. He developed a parser and semantic analysis system that translated simple English commands into logical statements for STRIPS. For example, the task of box pushing just mentioned was posed for Shakey in English as follows:

Use BOX2 to block door DPDPCLK from room RCLK.

(BOX2, DPDPCLK, and RCLK were the names Shakey used to identify the box, door, and room in question. We were obliging enough to use Shakey's names for things when giving it tasks to perform.)

Coles's program, called ENGROB,[26] translated this English command into the following condition to be made true (expressed in the language of the predicate calculus):

BLOCKED(DPDPCLK, RCLK, BOX2)

This condition was then given to STRIPS to make a plan for achieving it.

Coles was also interested in getting Shakey to solve problems requiring indirect reasoning. He set up an experiment in which Shakey was to push a box off an elevated platform. To do so, it would have to figure out that it would need to push a ramp to the platform, roll up the ramp, and then push the box. This task was given to Shakey in English as "Push the box that is on the platform onto the floor." The task was successfully executed and described in one of the Shakey technical reports.[27]

The "push-the-box-off-the-platform" task was Coles's way of showing that Shakey could solve problems like the "monkey-and-bananas" problem. That problem, made famous by John McCarthy as an example for deductive reasoning, involved a monkey, a box, and some bananas hanging out of reach. The monkey was supposed to be able to reason that to get the bananas, it would have to push the box under the bananas, climb up on the box, and then grab the bananas.[28] McCarthy is said to have heard Karl Lashley at the 1948 Caltech Hixon symposium describe a similar problem for demonstrating intelligent problem solving by chimpanzees.

One of the persons who was impressed with Shakey was Bill Gates, who later co-founded Microsoft. He saw the 1972 Shakey film as a junior in high school and drove down from Seattle to SRI (with Paul Allen, who would be the other co-founder of Microsoft) to have a look. According to one source, he was "particularly excited about Shakey moving things around so it could go up a ramp."[29]

12.1.7 *Shakey Runs into Funding Troubles*

Shakey was the first robot system having the abilities to plan, reason, and learn; to perceive its environment using vision, range-finding, and touch sensors; and to monitor the execution of its plans. It was, perhaps, a bit ahead of its time. Much more research (and progress in computer technology generally) would be needed before practical applications of robots with abilities such as these would be feasible. We mentioned some of the limiting assumptions that were being made by robot research projects at that time in one of our reports about Shakey:

> Typically, the problem environment [for the robot] is a dull sort of place in which a single robot is the only agent of change – even time stands still until the robot moves. The robot itself is easily confused; it cannot be given a second problem until it finishes the first, even though the two problems may be related in some intimate way. Finally, most robot systems cannot yet generate plans containing explicit conditional statements or loops.

Even though the SRI researchers had grand plans for continuing work on Shakey, DARPA demurred, and the project ended in 1972. This termination was unfortunate, because work on planning, vision, learning, and their integration in robot systems had achieved a great deal of momentum and enthusiasm among SRI researchers. Furthermore, several new ideas for planning and visual perception were being investigated. Many of these were described in detail in a final report for the Shakey project.[30]

Among these ideas, a particularly important one involved techniques for constructing plans in a hierarchical fashion. To do so, an overall plan consisting of just "high-level" actions must be composed first. Such a plan can be found with much less searching than one consisting of all of the lowest level actions needed. For example, one's plan for getting to work might involve only the decision either to take the subway or to drive one's car. Then, gradually, the high-level plan must be refined in more and more detail until actions at the lowest level (such as which set of car keys should be used) would eventually be filled in.

A Stanford computer science graduate student working at SRI, Earl Sacerdoti (1948–), proposed two novel methods for hierarchical planning. First (as part of his master's degree work), he programmed a system he called ABSTRIPS.[31] It consisted of a series of applications of STRIPS – beginning with an easy-to-compose plan that ignored all but the most important operator preconditions. Subsequent applications of STRIPS, guided by the higher level plans already produced, would then gradually take the more detailed preconditions into account. The result was a series of evermore-detailed plans, culminating in one that could actually be executed.

For his Ph.D. work, Sacerdoti went on to develop a more powerful hierarchical planning system he called NOAH (for Nets of Action Hierarchies).[32] Unlike ABSTRIPS, whose action operators were all at the same level of detail (albeit with preconditions that could be selectively ignored), NOAH employed action operators at several levels of detail. Each operator came equipped with specifications for how it could be elaborated by operators at a lower level of detail. Furthermore, NOAH's representation of a plan, in a form Sacerdoti called a "procedural network," allowed indeterminacy about the order in which plan steps at one level might be carried out. This "delayed commitment" about ordering permitted the more detailed steps of the elaborations of nonordered plans at one level to be interleaved at the level below, often with a consequent improvement in overall efficiency.

Sacerdoti was hoping to use his hierarchical planning ideas in the Shakey project, so he and the rest of us at SRI were quite disappointed that DARPA was not going to support a follow-on project. (Basic research on robots was one of the casualties of the DARPA emphasis on applications work that began in the early 1970s.) However, we were able to talk DARPA into a project that had obvious military relevance but still allowed us to continue work on automatic planning, vision, and plan execution. Interestingly, the project was pretty much a continuation of our research work on Shakey but with a human carrying out the planned tasks instead of a robot. We called it the "computer-based consultant (CBC) project." I'll describe it in a subsequent chapter.

Sacerdoti and the SRI researchers were not alone in recognizing the importance of hierarchical planning. As part of his Ph.D. work at the University of Edinburgh, Austin Tate (1951–) was developing a network-based planning system called INTERPLAN.[33] In 1975 and 1976, supported by the British Science Research Council, Tate and colleagues from operations research produced a hierarchical planner called NONLIN.[34] The planner took its name from the fact that, like NOAH, some of the plan steps were left unordered until they were elaborated at lower levels of the hierarchy.

Other planning systems grew out of the NOAH and NONLIN tradition. One was the interactive plan-generation and plan-execution system SIPE-2 developed by David E. Wilkins at SRI International.[35] Another was O-PLAN developed by Tate and colleagues at the Artificial Intelligence Applications Institute (AIAI) at the University of Edinburgh.[36] These systems have been widely used, extended, and applied.[37]

12.2 The Stanford Cart

In the early 1960s, James Adams, a Mechanical Engineering graduate student at Stanford (and later a Stanford professor), began experimenting with a four-wheeled, mobile cart with a TV camera and a radio control link. Lester Earnest wrote (in his

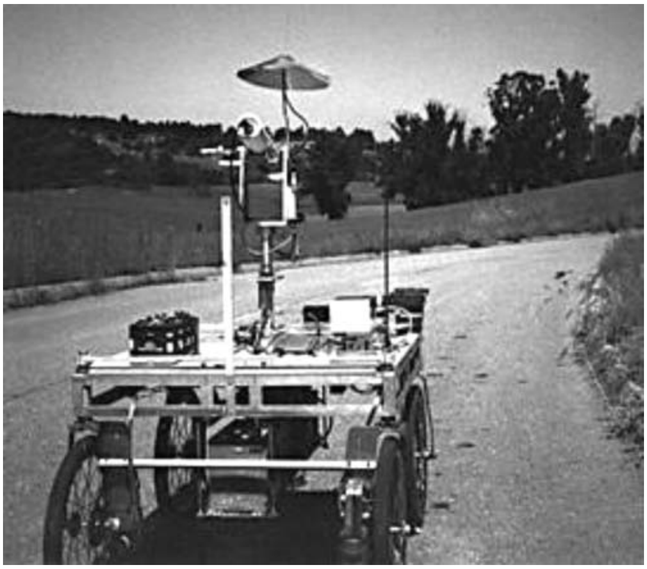


Figure 12.8. The Stanford cart. (Photograph courtesy of Lester Earnest.)

history of the several projects using this cart) "Among other things, Adams showed in his dissertation that with a communication delay corresponding to the round trip to the Moon (about $2\frac{1}{2}$ seconds) the vehicle could not be reliably controlled if traveling faster than about 0.2 mph (0.3 kph)."[38]

After Earnest joined the Stanford AI Laboratory, he and Rodney Schmidt, an Electrical Engineering Ph.D. student, got an upgraded version of the cart to "follow a high contrast white line [on the road around the Lab] under controlled lighting conditions at a speed of about 0.8 mph (1.3 kph)." Other AI graduate students also experimented with the cart from time to time during the early 1970s. A picture of the cart (as it then appeared) is shown in Fig. 12.8.

When Hans Moravec came to Stanford to pursue Ph.D. studies on visual navigation, he began work with the cart, "but suffered a setback in October 1973 when the cart toppled off an exit ramp while under manual control and ended up with battery acid throughout its electronics." By 1979 Moravec got the refurbished cart, now equipped with stereo vision, to cross a cluttered room without human intervention. But it did this very slowly. According to Moravec,[39]

The system was reliable for short runs, but slow. The Cart moved 1 m every 10 to 15 min, in lurches. After rolling a meter it stopped, took some pictures, and thought about them for a long time. Then it planned a new path, executed a little of it, and paused again. It successfully drove the Cart through several 20-m courses (each taking about 5 h) complex enough to necessitate three or four avoiding swerves; it failed in other trials in revealing ways.

A short video of the cart in action can be seen at http://www.frc.ri.cmu.edu/users/hpm/talks/Cart.1979/Cart.final.mov. Along with Shakey, the Stanford Cart resides in the Computer History Museum in Mountain View, California. They were the progenitors of a long line of robot vehicles, which will be described in subsequent chapters.

Notes

1. Hans P. Moravec, *Robot: Mere Machine to Transcendent Mind*, pp. 18–19, Oxford: Oxford University Press, 1999. **[162]**
2. A copy of the proposal is available online at http://www.ai.sri.com/pubs/files/1320.pdf. Its cover page says "Prepared by Nils J. Nilsson," but it was really a team effort, and many of the ideas were elaborations of those put forward in Rosen's 1963 memo. **[162]**
3. A copy of the complete statement can be found at http://ai.stanford.edu/~nilsson/automaton-work-statement.pdf. **[163]**
4. Online copies of SRI's proposals for the automaton project, subsequent progress reports, and related papers can be found at http://www.ai.sri.com/shakey/. **[163]**
5. See http://www.robothalloffame.org/. **[164]**
6. See http://www.computerhistory.org/timeline/?category=rai. **[164]**
7. J. Doran and Donald Michie, "Experiments with the Graph Traverser Program," *Proceedings of the Royal Society of London*, Series A, Vol. 294, pp. 235–259, 1966. **[167]**
8. The first written account of this idea was in Charles A. Rosen and Nils Nilsson, "Application of Intelligent Automata to Reconnaissance," pp. 21–22, SRI Report, December 1967. Available online at http://www.ai.sri.com/pubs/files/rosen67-p5953-interim3.pdf. **[168]**
9. Personal communication, October 24, 2006. **[168]**
10. See Peter Hart, Nils Nilsson, and Bertram Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions System Science and Cybernetics*, Vol. 4, No. 2, pp. 100–107, 1968, and Peter Hart, Nils Nilsson, and Bertram Raphael, "Correction to 'A Formal Basis for the Heuristic Determination of Minimum-Cost Paths,'" *SIGART Newsletter*, No. 37, pp. 28–29, December 1972. **[168]**
11. See, for example, Korf's publications at http://www.cs.ucla.edu/~korf/publications.html. **[168]**
12. In a 2003 paper titled "A* Parsing: Fast Exact Viterbi Parse Selection," Dan Klein and Christopher Manning wrote "The use of A* search can dramatically reduce the time required to find a best parse by conservatively estimating the probabilities of parse completions." **[168]**
13. In an e-mail to Peter Hart dated March 27, 2002, Brian Smart, the chief technical officer of a vehicle-navigation company, wrote "Like most of the 'location based services' and 'vehicle navigation' industry, we use a variant of A* for computing routes for vehicle and pedestrian navigation applications." **[168]**
14. In an e-mail to me dated June 14, 2003, Steven Woodcock, a consultant on the use of AI in computer games, wrote "A* is far and away the most used . . . and most useful . . . algorithm for pathfinding in games today. At GDC roundtables since 1999, developers have noted that they make more use of A* than any other tool for pathfinding." **[168]**
15. See Bertram Raphael *et al.*, "Research and Applications – Artificial Intelligence," pp. 27–32, SRI Report, April 1971. Available online at http://www.ai.sri.com/pubs/files/raphael71-p8973-semi.pdf. **[169]**
16. For a description of how QA3 developed a plan for Shakey to push three objects to the same place, for example, see Nils J. Nilsson, "Research on Intelligent Automata," Stanford Research Institute Report 7494, pp. 10ff, February 1969; available online at http://www.ai.sri.com/pubs/files/nilsson69-p7494-interim1.pdf. **[169]**
17. McCarthy and Hayes first described this problem in John McCarthy and Patrick Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," in Donald Michie and Bernard Meltzer (eds.), *Machine Intelligence*, Vol. 4, pp. 463–502, 1969. Reprinted in Matthew Ginsberg (ed.), *Readings in Nonmonotonic Reasoning*, pp. 26–45,

San Francisco: Morgan Kaufmann Publishers, Inc., 1987. Preprint available online at http://www-formal.stanford.edu/jmc/mcchay69/mcchay69.html. **[169]**

18. See Carl Hewitt, "PLANNER: A Language for Proving Theorems in Robots," *Proceedings of the First International Joint Conference on Artificial Intelligence*, pp. 295–301, 1969. **[170]**
19. See Richard Fikes and Nils Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, Vol. 2, Nos. 3–4, pp. 189–208, 1971. Available online at http://ai.stanford.edu/users/nilsson/OnlinePubs-Nils/PublishedPapers/strips.pdf. **[170]**
20. The generalization and execution mechanisms are described in Richard Fikes, Peter Hart, and Nils Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence*, Vol. 3, No. 4, pp. 251–288, 1972. Available online (as an SRI report) at http://www.ai.sri.com/pubs/files/tn070-fikes72.pdf. **[171]**
21. See Claude Brice and Claude Fennema, "Scene Analysis Using Regions," *Artificial Intelligence*, Vol. 1, No. 3, pp. 205–226, 1970. **[172]**
22. Richard O. Duda and Peter E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Communications of the ACM*, Vol. 15, pp. 11–15, January 1972. See also, Peter E. Hart, "How the Hough Transform Was Invented," *IEEE Signal Processing Magazine*, November, 2009. **[172]**
23. Bertram Raphael *et al.*, "Research and Applications – Artificial Intelligence," Part V, SRI Final Report, December 1971; available online at http://www.ai.sri.com/pubs/files/raphael71-p8973-final.pdf. **[174]**
24. For more information about Shakey's visual routines, in addition to the final report just cited, see Richard O. Duda, "Some Current Techniques for Scene Analysis," SRI Artificial Intelligence Group Technical Note 46, October 1970, available online at http://www.ai.sri.com/pubs/files/tn046-duda70.pdf. **[174]**
25. This experiment, as well as other information about Shakey, is described in Bertram Raphael *et al.*, "Research and Applications – Artificial Intelligence," SRI Final Report, December 1971, available online at http://www.ai.sri.com/pubs/files/raphael71-p8973-final.pdf; in Nils Nilsson (ed.), "Shakey The Robot," SRI Technical Note 323, April 1984, available online at http://www.ai.sri.com/pubs/files/629.pdf; and in a 1972 film, *Shakey: An Experiment in Robot Planning and Learning*, available online at http://www.ai.sri.com/movies/Shakey.ram. **[174]**
26. L. Stephen Coles, "Talking with a Robot in English," *Proceedings of the International Joint Conference on Artificial Intelligence*, Washington, DC, May 7–9, Bedford, MA: The MITRE Corporation, 1969. **[174]**
27. See L. Stephen Coles *et al.*, "Application of Intelligent Automata to Reconnaissance," SRI Final Report, November 1969, available online at http://www.ai.sri.com/pubs/files/coles69-p7494-final.pdf. **[175]**
28. The problem was introduced by McCarthy in his July 1963 memo "Situations, Actions, and Causal Laws," reprinted as Section 7.2 of his paper "Program with Commonsense," which appeared in Marvin Minsky (ed.), *Semantic Information Processing*, pp. 403–418, Cambridge, MA: MIT Press, 1968. **[175]**
29. E-mail from Eric Horvitz of May 12, 2003. **[175]**
30. Peter E. Hart *et al.*, "Artificial Intelligence – Research and Applications," Technical Report, Stanford Research Institute, December 1972. (Available online at http://www.ai.sri.com/pubs/files/hart72-p1530-annual.pdf.) See also Richard E. Fikes, Peter E. Hart, and Nils J. Nilsson, "Some New Directions in Robot Problem Solving," in *Machine Intelligence 7*, Bernard Meltzer and Donald Michie (eds.), pp. 405–430, Edinburgh:

Edinburgh University Press, 1972. (The SRI Technical Note 68 version is available online at http://www.ai.sri.com/pubs/files/1484.pdf.) **[175]**

31. Earl D. Sacerdoti, "Planning in a Hierarchy of Abstraction Spaces," pp. 412–422, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, 1973. (The SRI AI Center Technical Note 78 version is available online at http://www.ai.sri.com/pubs/files/1501.pdf.) **[176]**
32. Earl D. Sacerdoti, "The Non-Linear Nature of Plans," *Proceedings of the International Joint Conference on Artificial Intelligence*, 1975. (The SRI AI Center Technical Note 101 version is available online at http://www.ai.sri.com/pubs/files/1385.pdf.) Also see Earl D. Sacerdoti, *A Structure for Plans and Behavior*, New York: Elsevier North-Holland, 1977. (The SRI AI Center Technical Note No. 109 version is available online at http://www.ai.sri.com/pubs/files/762.pdf.) **[176]**
33. Austin Tate, "Interacting Goals and Their Use," *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, pp. 215–218, Tbilisi, USSR, September 1975; available online at http://www.aiai.ed.ac.uk/project/oplan/documents/1990-PRE/1975-ijcai-tate-interacting-goals.pdf. Austin Tate, "Using Goal Structure to Direct Search in a Problem Solver," Ph.D. thesis, University of Edinburgh, September 1975 available online at http://www.aiai.ed.ac.uk/project/oplan/documents/1990-PRE/. **[176]**
34. Austin Tate, "Generating Project Networks," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, pp. 888–893, Boston, MA, August 1977; available online at http://www.aiai.ed.ac.uk/project/oplan/documents/1990-PRE/1977-ijcai-tate-generating-project-networks.pdf. **[176]**
35. See the SIPE-2 Web page at http://www.ai.sri.com/~sipe/. **[176]**
36. Ken Currie and Austin Tate, "O-PLAN: The Open Planning Architecture," *Artificial Intelligence*, Vol. 52, pp. 49–86, 1991. Available online at http://www.aiai.ed.ac.uk/project/oplan/documents/1991/91-aij-oplan-as-published.pdf. **[176]**
37. See, for example, AIAI's "Planning and Activity Management" Web page at http://www.aiai.ed.ac.uk/project/plan/. **[176]**
38. Lester Earnest, "Stanford Cart," August 2005; available online at http://www.stanford.edu/~learnest/cart.htm. **[177]**
39. Hans P. Moravec, "The Stanford Cart and the CMU Rover," *Proceedings of the IEEE*, Vol. 71, No. 7, pp. 872–884, July 1983. **[177]**