AN EXPERIMENT IN ROBOT TOOL USING

by

L. Stephen Coles


Artificial Intelligence Group
Technical Note 41

SRI Project 8259

The research reported here was supported by the Advanced Research Projects Agency and the Rome Air Development Center under Contract AF30(602)-4124 and is continuing under Contract NAS 12-2221 with the Advanced Research Projects Agency and the National Aeronautical and Space Administration.
AN EXPERIMENT IN ROBOT TOOL USING

Abstract

Within the Artificial Intelligence Group of Stanford Research Institute we have been engaged in the design of an intelligent automaton for about five years. The automaton in question is a computer-controlled mobile robot capable of autonomously sensing information from its environment and performing tasks normally requiring human supervision. In keeping with our major purpose during the last several years, we have conducted a series of experiments, each designed to exhibit increasingly sophisticated problem-solving behavior on the part of the robot. The most difficult and complex task completed thus far has been a demonstration of robot tool using, i.e., the use of a simple ramp to permit the robot to roll from its initial position on the floor onto a platform about six inches above the floor in order to complete an assigned task. This paper describes the problem as specified to the robot, the assumptions made, and the methods used to accomplish the task.
I  INTRODUCTION

A classical problem in Artificial Intelligence known as the "Monkey and Bananas" problem\(^1,2^*\) can be stated briefly as follows:

A monkey is in a room in which a bunch of bananas are hanging from the ceiling, just out of reach. The monkey's problem, obviously, is to get the bananas. In a corner of the room is a chair. The solution deduced by the monkey is to push the chair to a location under the bananas, climb on top of the chair, and then easily reach for the bananas.

Psychologists have given the monkey and bananas problem to real monkeys, and although it is difficult for untrained monkeys, they do solve it in time. The problem is thus of interest because it provides a frame of reference for measuring the intelligence of our computer programs. Of more importance to workers in Artificial Intelligence, however, is the fact that this problem is characterized by one level of indirectness. That is, the solution requires an auxiliary device or tool (a chair in this case) not obviously needed at the start of the problem. Moreover, any computer system capable of solving this class of problems characterized by one level of indirectness, which also contains a logically complete deductive component, could in principle handle problems having an arbitrary number of levels of indirectness required for their solution, subject to the constraints of computer memory and response time. It should be noted in passing that problems possessing merely a half-dozen levels of indirectness more than challenge human ingenuity.

*References are listed at the end of this paper.
For the above reasons this problem has been chosen frequently by workers in Artificial Intelligence to demonstrate the capability and versatility of their particular problem-solving techniques and representations. Therefore, it seemed that this problem would be a good candidate for exhibiting the problem-solving capabilities of our robot. We had already completed experiments concerned with the exploration of unknown territory and navigation algorithms. More recently we succeeded in having the robot collect a number of specified objects into a designated place. This task is described in detail in Reference 3, where the reader will find additional information regarding the basic design of the hardware and software for the robot. The immediate difficulty with the monkey and bananas problem for the robot, however, was in devising a suitable isomorphism with the robot as monkey and the role of the bananas and chair being played by other suitable objects accessible within the robot's domain. In this manner we could preserve the logical relations inherent in the original formulation and thus retain the essential property of one level of indirectness. Although there is potentially a wide spectrum of possible reformulations of the monkey and bananas problem in the robot's domain ranging in difficulty from the trivial to the impossible, in practice, selecting one that is both nontrivial, yet feasible in a reasonable amount of time, proved difficult indeed.

The reformulation finally chosen, referred to as the "Robot and Box" problem, can be stated as follows:

The robot is in a room in which a box is resting on top of a platform. The robot's problem is to push the box off the
platform and onto the floor. (Because the robot is on wheels, the direction of approach makes no difference. The robot cannot directly reach the box, since the platform intervenes.) In a corner of the room is a ramp. The solution deduced by the robot is to push the ramp to a location adjoining the platform, align it properly against the platform, roll up the ramp onto the platform, and then easily push the box onto the floor.

After a year of detailed preparation, the robot successfully solved the robot and box problem as stated above in October 1969. It is believed that this was the first instance of a computer-controlled robot using an elementary tool as an essential ingredient in the solution of a problem. Figure 1(a) shows the robot in its laboratory environment. The box, the platform, and the ramp are shown in a legal initial configuration for the robot and box problem. Figure 1(b) shows an intermediate stage in the solution of the problem in which the robot has already deduced the relevance of the ramp to reducing differences in height and is pushing the ramp over to the platform. Figure 1(c) shows the result of proper alignment with the high side of the ramp flush against the platform, and "roll-up" half completed. Finally, Figure 1(d) shows successful completion of the task. In practice the entire procedure illustrated in the above figures takes about thirty minutes for execution. About twenty minutes is consumed by the problem-solving programs running on an SDS-940 computer operating in a time-sharing mode, while approximately ten minutes is needed by the robot to physically move and align the ramp, roll up, and push the box onto the floor.
FIGURE 1 THE ROBOT AND THE BOX. The sequence from (a) to (d) shows the robot using a ramp as a tool to climb up on a platform and push a box off the platform.
In the remainder of this paper we will examine in some detail the methods used for stating the problem to the robot in English, establishing the feasibility of a solution using a logically complete deductive system, and actually executing the task in terms of primitive actions available to the robot. Later we will speculate on some of the implications this task has for future robot experiments.

II STATEMENT OF THE PROBLEM

Although the human has a choice of several different ways of formulating the problem in natural language, one of the simplest ways of transmitting the problem to the robot is to type the following English imperative sentence on the teletype:

"Push the box that is on the platform onto the floor."

However, for this sentence to be recognized as a syntactically correct and semantically meaningful command, several assumptions must be satisfied. First, the initial configuration of the experimental room must be essentially as in Figure 1(a). The exact location of the platform in the room and location of the box on the platform must be fixed and known in advance to the robot. Also the robot's current $x$, $y$, and $\theta$ coordinates in the room and its camera tilt and pan angles must be known to the robot. Second, the robot must have available some information giving the approximate neighborhood of the ramp in the room. It should be emphasized that the robot need not know in advance the exact position and orientation of the ramp in the room. These parameters may be fixed by the experimenter at will subject to the constraint of certain boundary conditions; e.g., the ramp should not be so close to a wall that the robot could not maneuver it into position.
The first problem for the robot in properly interpreting the command is to verify that it is a well-formed English sentence. References 4 and 5 describe in some detail general methods that have been developed for recognizing and translating English sentences into predicate calculus. The transformational component of the natural-language robot grammar is first called into play. After some inspection it recognizes that the sentence is of the form

\[
push \langle \text{object} \rangle \text{ onto } \langle \text{place} \rangle
\]

where \( \langle \text{object} \rangle \) refers to the description of an object and \( \langle \text{place} \rangle \) refers to the description of a location. The object and location descriptions are then parsed by the phrase structure component of the grammar, and the syntactic correctness of the command is established.

The value of the transformational analysis is seen by considering the possibility of omitting it in favor of a purely phrase-structure analysis. In such a case, the grammar without further semantic information must label the sentence syntactically ambiguous. It cannot distinguish this sentence from the sentence "Push the box that is on the platform near the door," which has an identical surface structure and where the ambiguity is more manifest. Did the speaker intend the robot to push the box so that its final location is near the door or did he intend me to distinguish the box that is on the platform near the door from the box that is on the platform in another part of the room? Only clarification by the speaker or semantic information from the environment can resolve this ambiguity. Obviously there are many ways to proceed. The transformational component merely serves in this case to circumvent this sort of problem where the preposition "onto" makes the correct interpretation obvious.
Concurrently with the syntactic recognition procedure, a deep structure corresponding to the meaning of the command is constructed. The predicate calculus translation of the English sentence produced by the semantic component is as follows:

\[ C: (\exists s_f, x, y, z) [\text{On}(x, y, s_f) \land \text{Is}(x, \text{box}) \land \text{On}(x, z, s_f) \land \text{Is}(z, \text{platform}) \land \text{Is}(y, \text{floor})] \]

Loosely paraphrased back into English, the predicate calculus is asking: 
"Does there exist a final state, \( s_f \), in which a box located on a platform can be located on the floor? If the answer is yes, then reply in English and do it. Otherwise, explain why not."

III FEASIBILITY OF SOLVING THE PROBLEM

The next step for the robot, having achieved a formulation of the problem in its own terms, is to figure out how to solve it, given what it knows about its current environment and what it knows about its own primitive capabilities for manipulating that environment, both kinds of knowledge being expressed in a common axiomatic language. The situation calculus (SC) as briefly described in Appendix I, is used to present the problem. Let \( \mathcal{P} = \{\text{On}(x, y, s), \text{At}(x, y, s)\} \), where

\[ \text{On}(x, y, s) \text{ is a three-place predicate denoting that object } x \text{ is on object } y \text{ in state } s, \text{ and} \]

\[ \text{At}(x, y, s) \text{ is a three-place predicate denoting that object } x \text{ is at a location adjoining object } y \text{ in state } s. \]

Further, let \( \mathcal{F} = \{\text{Push}(x, y, z, s), \text{Rollup}(x, y, s), \text{Move}(x, y, z, s)\} \), where

\[ \text{Push}(x, y, z, s) \text{ is a four-argument function, which maps the state } s \text{ into a new state in which the agent } x \text{ has pushed the object } y \text{ to } z; \]

\[ \text{Rollup}(x, y, s) \text{ is a three-argument function, which maps the state } s \text{ into a new state in which the agent } x \text{ has rolled up the object } y; \text{ and} \]
Move(x,y,z,s) is a four-argument function, which maps the state s into a new state in which the agent x has moved the object y to a location adjoining the object z.

Finally, let G be given by A1,...,A12 as follows:

A1: Is(f,floor)
A2: Is(p,platform)
A3: Is(b,box)
A4: Is(w,ramp)
A5: On(r,f,s_1)
A6: On(b,p,s_1)
A7: (\forall s) [On(p,f,s) \land On(w,f,s)]
A8: (\forall s) [On(r,p,s) \land On(b,p,s) \Rightarrow On(b,f,push(r,b,f,s))]
A9: (\forall s) [At(w,p,s) \land On(r,f,s) \Rightarrow On(r,p,rollup(r,w,s))]
A10: (\forall s) [On(r,f,s) \land On(w,f,s) \land On(p,f,s) \Rightarrow At(w,p,move(r,w,p,s))]
A11: (\forall s) [On(r,f,s) \Rightarrow On(r,f,move(r,w,p,s))]
A12: (\forall s) [On(b,p,s) \Rightarrow On(b,p,rollup(r,w,move(r,w,p,s)))]

where \(G = \{f,p,b,w,r\}^\ast\).

Now, with the above formulation in the SC, we may pose such problems as:

P1: (\exists s_f) [At(w,p,s_f)]
P2: (\exists s_f) [On(r,p,s_f)]
P3: (\exists s_f) [On(b,f,s_f)]

where P3 is a simplified version of the robot and the box problem as stated in the preceding section. The solution to P1 is simple: \(s_f = move(r,w,p,s_1)\), which is obtained by one application of A5, A7, and A10 instantiating \(s_1\) for \(s\). Similarly, with a somewhat greater amount of effort the solution to P2 becomes: \(s_f = rollup(r,w,move(r,w,p,s_1))\).

The solution to the robot and the box problem under the simplified
formulation P3 is then seen to be $s_f = \text{push}(r,b,\text{rollup}(r,w,\text{move}(r,w,p,s_i)))$. 
The solution to the complete formulation by the theorem prover is then

$$s_i = s_i$$
$$s_f = \text{push}(r,b,\text{rollup}(r,w,\text{move}(r,w,p,s_i)))$$
$$x = b$$
$$y = f$$
$$z = p$$

Note that while Axioms A1 and A2 appear to be superfluous at first glance, they are indeed necessary for the solution. One must explicitly state that the robot's being on the floor is an invariant under Move and the box's being on the platform is an invariant under Rollup, etc.

This apparent difficulty is an instance of a more basic problem characteristic of the SC as a whole that John McCarthy has referred to as the "frame problem." That is, in writing axioms in the SC, one must fully specify not only what each action does do in changing the relevant features of a situation, but also what each action does not undo.

Various schemes have been proposed for providing a general solution to the frame problem, but as yet no one has achieved a fully satisfactory method that can be implemented within the framework of the first-order predicate calculus.

One should also remark that although the solution to any given problem may be feasible in principle, other prohibitions on robot behavior that are also entered as axioms by the user may prevent the robot from actually carrying out any plan establishing feasibility.
IV EXECUTION OF THE PLANNED SOLUTION

Having established the feasibility of a solution, i.e., discovering from the initial state an obtainable final state that satisfies the requirements of the problem, it remains to execute the changes of state. Why not use the solution to the feasibility stage in a constructive manner to guide the execution? This amounts to first telling the user that you are about to begin, unwinding the primitive functions composed in the solution, and evaluating them in reverse order with their appropriate arguments. "Telling the user" amounts to generating the English sentence "I will now push the box on top of the platform onto the floor."

The execution of primitive functions with proper arguments entails appropriate calls on LISP routines, which in turn call the two-letter FORTRAN commands that actually drive the vehicle through its motions. (The two-letter commands are defined in Reference 5.) A typical sequence of two-letter commands corresponding to the Move-Rollup-Push sequence is

"XG-1.,YG0.5,GOTUS1.,WHTI-5.,REPIRVPU9.1,9.5,4.6,4.4,19.1,
TU-90.,MO6.5,TU60.,MO7.5,XG4.5,YG25.,GOM02.5,OV3,TU-105.,MO6.5,
MO-1.,TU105.,OVO,MO7.,XG14.,YG15.5,GOTU-180.,MO15.,TU-90.,
OV3,MO3.,MO-3,OV0,TU-90.,MO14.," .

Of course the above string of 37 two-letter commands is meaningful only to those highly familiar with robot operations, but it does serve to illustrate the kind of complexity one rapidly encounters when attempting even the most trivial tasks. For the benefit of those not versed in this intermediate command language, we briefly describe in English what this string of commands entails from the robot's point of view.
My first subtask is to execute Move($r,w,p,s_1$) or, in other words, to move the ramp over to the platform and align it properly. To do this I must first discover where the ramp is. To do this I must first see it. To do this I must first go to a place where, if I looked in the right direction, I might see it. This sets up the subsubtask of computing the coordinates of a desirable vantage point in the room, based on my approximate knowledge of where the ramp is. Next I have the problem of getting to the vantage point. Can I go directly or will I have to plan a journey around obstacles? Will I be required to travel through unknown territory to get there if I go by an optimal trajectory, and if so, what weight should I give to avoiding this unknown territory? When I get there I will have to turn myself and tilt the television camera to an appropriate angle, then take a picture in. Will I see a ramp? The whole ramp? Nothing but the ramp? Do I need to make a correction for depth perception? etc., etc.

This sort of reasoning in the form of primitive routines and reflex actions goes on until the robot successfully takes a picture of the ramp with a reasonable probability of error (Figure 2). Based on this picture, a model of the experimental room is constructed, which might look as indicated in Figure 3.

The angle $\alpha$ is estimated and a strategy for reducing $\alpha$ to an angle within acceptable tolerances is determined. This amounts to re-orienting the ramp with an appropriate number of rotational pushes.
FIGURE 3 PLAN VIEW OF EXPERIMENTAL ROOM

RAMP

α

BOX

PLATFORM
Next comes a long translational push (cf. Figure 1b) putting the ramp into actual contact with the platform. Now comes the difficult task of alignment with all its inherent subtleties: getting in the right place to push at the right place by the right amount to cause the high side of the ramp to come up flush against the platform with a gap of at most one-half inch. To conclude the sequence—once alignment has been accomplished successfully—the rollup (Figure 1c) and the push (Figure 1d) can be handled easily.

V OPERATIONAL EXPERIENCE

In practice, the robot—as configured with software implemented on an SDS 940—is not very skilled at getting up on top of a platform in a reasonable amount of time. The recognition of the English command takes in the neighborhood of 90 seconds, even with a comparatively light load on the time-sharing system of the SDS 940 computer. Proof of the theorem establishing feasibility involves only 29 steps with well-tailored axioms, yet still takes 20 minutes on the average. Actual picture taking and pushing may add another 15 minutes, so the total time from beginning to end for accomplishing this elementary task with the indicated level of generality takes over half an hour of real time. Obviously the robot is not competitive with humans at this task.

Even allowing for patience on the part of user, he must be willing to accept the possibility that on certain occasions the robot will simply fail to accomplish the task in practice, even though in principle its strategy for achieving the goal is correct. The principal source of unreliability in the system is poor vision, although cumulative error in the stepping motors contributes to the possibility of failure. With
considerable effort in tuning the dynamic range of the TV camera, adjusting the antenna for best reception of a noisy signal (even the time of day seems crucial, since background radiation from adjoining labs can create disturbing radio interference), and carefully calibrating the software for best depth-perception factors, the vision software can generate reproducible ramp coordinates in the actual room only to within about an inch, but this is sufficient for accomplishing the overall objective.

VI DIFFICULTIES ENCOUNTERED

In addition to the problems enumerated above of "tuning" the robot hardware, and other hardships imposed by unreliable robot and computer hardware, the only theoretical difficulty encountered was the "frame problem" discussed earlier. This problem was effectively bypassed by adding a few more axioms to the system rather than really "solved."

Why then did the task actually take over one year for completion? One can distinguish at least two other classes of unanticipated difficulty worthy of comment: software compatibility and pragmatic difficulties.

The "robot and box" task was accomplished by building a top-level software system out of the existing robot programs. Some of these programs were originally designed merely to exercise various components of the robot hardware; others were meant to be trial, experimental versions of the routines for model maintenance, display, etc., for gaining experience with the system before implementing more polished versions. Nevertheless, software-compatibility problems arose far out of proportion to what was originally expected. Frequently, undocumented programs designed for one purpose had to be modified for another slightly different purpose;
the side effects became evident only at a later time. On other occasions, we found that reasonably well-documented programs had unarticulated assumptions about the use of memory, a commodity that was abundant when the software packages were checked out in isolation, but scarce when various packages were linked together and required to operate in concert. Marshaling resources and manpower to carry out the linkage of communicating software packages already in existence also proved difficult. Occasionally the author of a particular routine was no longer with the Institute. Literally weeks were spent in articulating software conventions of different subsystems—such as vision and tactics—so that appropriate parameters could be passed between them.

Some of the most trivial inconsistencies in conventions took the most time to locate and patch. For example, one group measured angles in degrees—the other in radians; one group established a coordinate system 90 degrees out of phase with the other, etc., and (perhaps most humiliating), the LISP system was perfectly content to pass the FORTRAN system the number "51.000," whereas the FORTRAN system in certain routines could only accept the number 51 if it were formatted "51.00." This supposedly is an inevitable consequence of different teams of individual software designers working too long in isolation of one another and in the absence of a master specification to guide them. The system software in which the various packages were embedded was so rigid that it practically nullified the advantages of working on a time-sharing system. This mode of operation contributed its own debugging problems, further delaying the loading and initialization process. For the new PDP-10 system, we are designing a complete new software structure that will provide a uniform framework for future robot experiments.
What we have chosen to call pragmatic difficulties added a whole new dimension to the conventional hardware/software debugging process. Three brief examples are given:

(1) Given that the ramp and the platform had already been designed and fabricated by the carpentry shop, with due attention to the proper dimensions and distribution of weight for use by the robot, it was discovered that the sliding friction of the ramp on the floor was too great for the robot to push easily. Solution: Go to the hardware store, buy metal furniture glides, bring in a hammer from home, hammer them in. Time: one day.

(2) It has been established empirically that the ramp tends to slip and fall out of alignment during pushing if not pushed directly on its center of gravity. Solution: Attach spring-loaded push bar on the robot. Time: three days.

(3) Edges on the ramp still do not show up clearly for proper identification, even though care was taken to paint adjoining faces with highly contrasting colors. Solution: Bring brushes and paint from home, paint certain edges white, tape black paper computer tape on other edges, add more fluorescent lights to the ceiling fixtures. Time: two days.

Suffice it to say that in preparing for the experiment a considerable number of expedient solutions were rapidly implemented in order to bypass purely pragmatic difficulties, some requiring considerable ingenuity. A
few of them may be regarded as "cheating" by outside observers, i.e., not abiding by certain tacitly stated ground rules; however, it should be noted that, through perseverance, none of the fundamental assumptions about the nature of the task were compromised along the way.

VII IMPLICATIONS FOR FUTURE EXPERIMENTS

The robot and box experiment has led us to consider two major directions for future research once our plans for replacing the SDS 940 computer with a PDP-10 are completed. One direction is to greatly expand the class of environments in which the robot can carry out specified tasks. By permitting the robot to move in office corridors, for example, rather than just the laboratory area, we can investigate intelligent problem solving under less contrived conditions. Among other requirements for corridor tasks, however, we will need to introduce much greater reliance on visual feedback than was needed in the more sterile environment of the laboratory.

The second direction for future research will be added emphasis on human communication with the robot during planning or replanning as a result of unsuccessful execution. Conversational monitoring at the very highest level is envisioned that will permit the human to play more of a role of "advice-giver" and the robot "advice-taker" than was possible under the old regime. In addition, we hope to devise experiments that will allow the robot to operate as a "goal-seeking" or purposeful automaton based on internally generated goals. In this mode the robot should be capable of requesting human assistance in carrying out its own objectives. This will give a more symmetric quality to the man-machine dialog and
open a new range of possibilities for demonstrating intelligent problem-solving behavior without adding new requirements for hardware modification.

VIII CONCLUSION

Although we have a long way to go before we realize practical applications, we are clearly on the threshold of exhibiting that elusive quality known as "intelligence" by the machines we are building and programming at SRI. The completion of the robot and box experiment in our judgment represents a significant step in this direction, and the software developed will be an essential ingredient in designing still more sophisticated experiments that will relax some of the environmental and conversational constraints mentioned in the preceding section.
REFERENCES


Appendix I
A BRIEF DESCRIPTION OF THE SITUATION CALCULUS

The Situation Calculus (SC) is a formal method for describing a class of transformations on a set of states in first-order predicate calculus. Its advantage lies in its ability to represent transformations in first-order logic (where extensive computer programs are available for logical deduction), in those cases where higher-order or modal logic would normally be required. It is often convenient to think of the state space and its accompanying transformations as a directed graph. For example,
where \( s_i \) and \( s_f \) are the initial and final states respectively, and the 
\( f_i, i = 1, 2, \ldots, 6 \) are transformations that map states onto states. In 
the directed graph any problem of going from the initial to the final 
state can be formulated as follows: Does there exist a path from \( s_i \) 
to \( s_f \)? In the above graph \( f_1 f_3 f_4, f_2 f_3 f_4, \) or \( f_2 f_5 f_2 \) might be given 
as a possible solution.

More formally, the situation calculus is a five tuple:

\[ SC = \{ \mathcal{G}, \mathcal{P}, \mathcal{J}, S, \Omega \} \]

where

\( \mathcal{G} \) is a set of axioms;

\( \mathcal{P} \) is a set of predicates describing states;

\( \mathcal{J} \) is a set of primitive functions that map states onto states,

\( \mathcal{J}:S \rightarrow S \);

\( S \) is a set of states; and

\( \Omega \) is a universe of discourse.

All \( n \)-ary predicates are represented in the SC as \( n+1 \)-ary predicates 
with the \( n+1 \)st argument being the universal state variable, \( s \). For 
example, \( P(x) \) becomes \( P(x,s) \) in the SC.

Axioms in the SC take the form

\[ P(x,s) \land [F(x,s)=s_2] \Rightarrow Q(x,s_2) \quad P, Q \in \mathcal{P}; \quad F \in \mathcal{J}; \quad s_1, s_2 \in S \]  \( (1) \)

where \( P \) is a predicate (or possibly a conjunction of predicates) over a 
set of arguments (represented by the vector \( x \)) describing the relevant 
features of the situation \( s_1 \); \( Q \) is a similar predicate describing the 
situation \( s_2 \), which obtains when \( F \) is applied to \( s_1 \); and thus, by con-
vention, \( F(x,s_1) = s_2 \). Simplifying the axiom schema (1) by implicitly
incorporating the fact that \( F(x,s_1) = x_2 \) and universally quantifying both \( x \) and \( s \), we may write

\[
(\forall x,s)\{P(x,s) = Q(x,F(x,s))\}
\]

Expression (2) is then the general form in which SC axioms are written, and we may think of \( P \) in our problem-solving process as describing the boundary conditions that must be satisfied in any state for the function \( F \) to be applicable. Alternatively, and perhaps even more simply, one may view the axioms as a set of productions or rewrite rules of the form \( P \rightarrow Q \). In addition, we must have some initial axioms that describe the initial conditions or the initial state, \( s_1 \), of the form

\[
P(c,s_1)
\]

where \( c \) is a vector of constants and \( P \) (possibly a conjunction of predicates) is true of the initial state.

A problem in the SC then takes the form

\[
(\exists s_f)\{R(c,s_f)\}
\]

or "Does there exist a final state such that \( R(c) \) is true in that state?"

A solution takes the form

\[
s_f = F_n F_{n-1}(c,F_2 F_1(c,s_1)) \ldots
\]

or more simply,

\[
s_f = F_n F_{n-1} \ldots F_2 F_1(c,s_1)
\]

\( F_i s_i \) \( i=1, 2, \ldots, n \leq \infty \)

i.e., a composition of functions applied to the initial state that will yield the final desired state such that \( R(c,s_f) \) is true in that state.

A complete presentation of the method for using first-order predicate calculus theorem proving in solving problems can be found in Reference 6.