June 1970

THE FRAME PROBLEM IN PROBLEM-SOLVING SYSTEMS

by

Bertram Raphael

Paper prepared for publication in the Proceedings of the Advanced Study Institute on Artificial Intelligence and Heuristic Programming, held at Menaggio, Italy, August 1970.

Artificial Intelligence Group

Technical Note 33

SRI Project 8259

This research is sponsored by the Advanced Research Projects Agency and the National Aeronautics and Space Administration under Contract NAS 12-2221.

I INTRODUCTION

The frame problem has taken on new significance during recent attempts to develop artificially intelligent systems. The problem deals with the difficulty of creating and maintaining an appropriate informational context or "frame of reference" at each stage in certain problem-solving processes. Since this is an area of current research, we are not prepared to present a solution to the frame problem; rather, the purpose of this paper is to sketch the approaches being pursued, and to invite the reader to suggest additions and improvements.

Although broader interpretations are possible, we think of an "artificially intelligent system" as meaning a programmed computer, with associated electronic and mechanical devices (e.g., a radio-controlled robot vehicle and camera), that is intelligent in the sense defined by McCarthy and Hayes:[1]*

> "... we shall say that an entity is intelligent if it has an adequate model of the world (including the intellectual world of mathematics, understanding of its own goals and other mental processes), if it is clever enough to answer a wide variety of questions on the basis of this model, if it can get additional information from the external world when it wants to, and can perform such tasks in the external world that its goals demand and its physical abilities permit."

Reference (2) discusses the research significance of attempting to build such an intelligent robot system.

---

* References are listed at the end of this paper.

The intelligent entity, as defined above, will have to be able to carry out tasks. Since a task generally involves some change in the world, it must be able to update its model so that it remains as accurate during and after the performance of a task as it was before. Moreover, it must be able to plan how to carry out a task, and this planning process usually requires keeping "in mind," simultaneously, a variety of possible actions and corresponding models of the hypothetical worlds that would result from those actions. The bookkeeping problems involved with keeping track of these hypothetical worlds account for much of the difficulty of the frame problem.

II THE FRAME PROBLEM

We shall illustrate the frame problem with a simple example. Suppose the initial world description contains the following facts (expressed in some suitable representation, whose precise form is beyond our immediate concern):

(F1) A robot is at position A.

(F2) A box called B1 is at position B.

(F3) A box called B2 is on top of B1.

(F4) A,B,C, and D are all positions in the same room.

Suppose, further, that two kinds of actions are possible:

(A1) The robot goes from x to y, and

(A2) The robot pushes B1 from x to y,

where x and y are in {A,B,C,D}. Now consider the following possible tasks:

Task (1): The robot should be at C.

This can be accomplished by the action of type A1, "Go from A to C." After performing the action, the system should "know" that facts F2

through F4 are still true, i.e. they describe the world after the action, but F1 must be replaced by

(F1′) The robot is at position C.

Task (2): B1 should be at C.

Now a "push" action must be used, and both F1 and F2 must be changed.

One can think of simple procedures for making appropriate changes in the model, but they all seem to break down in more complicated cases. For example, suppose the procedure is:

Procedure (a): "Determine which facts change by matching the task specification against the initial model."

This would fail in task (1) if the problem solver decided to get the robot to C by pushing B1 there (which is not unreasonable if the box were between the robot and C and pushing were easier than going around), thus changing F2.

Procedure (b): "Specify which facts are changed by each action operator."

This procedure is also not sufficient, for the initial world description may also contain derived information such as

(F6) B2 is at position B,

which happens to be made false in task (2).

More complicated problems arise when sequences of actions are required. Consider:

Task (3): The robot should be at D and, simultaneously, B2 should be at C.

The solution requires two actions, "Push B1 from B to C" and "Go from C to D," in that order. Any effective problem solver must have access to

the full sets of facts, including derived consequences that will be true as a result of each possible action, in order to produce the correct sequence.

Note that the frame problem is a problem of finding a practical solution, not merely finding a solution. Thus it resembles the famous traveling salesman problem or the problem of finding a winning move in a chess game, problems for which straightforward algorithms are known but usually worthless.

McCarthy and Hayes[1] divide intelligence into two parts: the epistemological part, which deals with the nature of the representation of the world, and the heuristic part, which deals with the problem-solving mechanisms that operate on the representation. They then proceed to concentrate upon the epistemological questions related to several aspects of intelligence (including the frame problem). Here, on the other hand, we are concerned with constructing a complete intelligent system, including both the world representations and the closely related problem-solving programs. In the following we shall assume that the representations are basically in the form preferred in Ref. (1), namely sets of sentences in a suitable formal logical language such a predicate-calculus; and we shall describe candidate organizations for the "heuristic part," i.e. the problem solver, of an artificially intelligent system that can cope with the frame problem.

III CURRENT APPROACHES

A. Complete Frame Descriptions

A frame can generally be completely described by some data structure, e.g. by a set of facts--expressed as statements in a predicate

calculus. If we think of each such frame as an object and each possible action as an operator that can transform one object (frame) into another, then we may use a problem-solving system such as GPS[3] for attempting to construct an object for which the desired goal conditions are true. Unfortunately, when the data base defining each frame reaches a non-trivial size, it becomes impractical to generate and store all the complete frame-objects. For example, suppose each possible frame is defined by 1000 elementary facts, an average of six different actions are applicable and heuristically plausible in any situation, and a typical task requires a sequence of four actions--not unreasonable assumptions about a simple robot system. Then the search tree of possible frames may have about 1000 nodes; it is not practical to store 1000 facts at each node. If each action causes changes in, say, three facts, then storing just the change information at each node is practical--provided appropriate bookkeeping is done to keep track of which of the original facts still holds after a series of actions. This bookkeeping seems to require considerable program structure in addition to (and quite separate from) the basic object, operator, and difference structure of a GPS-type system. The following approaches are concerned with this new bookkeeping problem.

B. State Variables

One way to keep track of frames is to consider each possible world to be in a separate state and to assign names to states. In this formulation, actions are state transition rules, i.e. rules for transforming one state into another. Since action rules are generally applicable to large classes of states, the description of an action can contain variables that range over state names.

Green describes an approach of this kind in detail in Ref. (4). Each fact is labeled with the name of the state in which it is known to be true. Additional facts that are state-independent describe the transitional effects of actions. For example, if S_0 is the name of the initial state and At(ob,pos,s) is a predicate asserting that object ob is at position pos in state s, then the conditions of the previous example may be partially defined by the following axioms:

(G1) At(Robot, A, S_0) (from F1)

(G2) At(B1, B, S_0) (from F2)

(G3) Box(B1) $\wedge$ Box(B2) (B1 and B2 are boxes)

(G4) $(\forall x,y,x)$[At(Robot,x,s) $\supset$ At(Robot,y,go(x,y,s))](from A1) .

At this point some explanations seem in order. Box(x) asserts that x is a box. Perhaps it would have been more consistent to write, e.g., Box(B1, S_0), because we only know that B1 is a box in the initial state. However, we do not contemplate allowing any actions that destroy box-ness, such as sawing or burning, so we could add the axiom $(\forall s)$Box(B1,s). Since we would then be able to prove that B1 is a box in all states, we suppress the state variable without loss of generality.

Each action, in this formalism, is viewed as a function. One argument of the function is always the state in which the action is applied, and the value of the function is the state resulting after the action. Thus, e.g., the value of go(A,C,S_0) is the name of the state achieved by going to C after starting from A in the initial state.

The appeal of this approach is that, if we have a theorem-proving program, no special problem-solving mechanisms or bookkeeping procedures are necessary. Action operators may be fully described by ordinary axioms

(such as G4 for the go operation) and the theorem-proving program, with its built-in bookkeeping, becomes the problem solver. For example, task (1) may be stated in the form, "Prove that there exists a state in which the robot is at C," or in predicate calculus, prove the theorem:

$$(*) \qquad (\exists s)\ \mathrm{At}(\mathrm{Robot},C,s) \quad .$$

From (G1) and (Gr), we can prove that (*) is indeed a theorem. By answer tracing during the proof (Ref. (5)), we can show that $s = \mathrm{go}(A,C,S_0)$, which is the solution.

For more complex actions, however, the major problem with this approach emerges: After each state change, the entire data base must be reestablished. We need additional axioms that tell not only what things change with each action, but also what things remain the same. For example, we know that B1 is at B in state S_0 (by G2), but as soon as the robot moves, say to state $\mathrm{go}(A,C,S_0)$, we no longer know where B1 is! To be able to figure this out, we need another axiom, such as

$$(\forall x,y,u,v,s)\ [\mathrm{At}(x,y,s) \wedge x \neq \mathrm{ROBOT} \supset \mathrm{At}(x,y,\mathrm{go}(u,v,s))] \quad .$$

("When the robot goes from u to v, the object x remains where it is at y.") Thus a prodigious set of axioms is needed to define explicitly how every action affects every predicate, and considerable theorem-proving effort is needed to "drag along" unaffected facts through state transitions. Clearly this approach will not be practical for problems involving many facts.

C. The World Predicate[6]

Instead of using a variety of independent facts to represent knowledge about a state of the world, suppose we take all the facts about a particular world and view the entire collection as a single entity, the

model $\mathfrak{M}$. We may then use a single predicate P, the "world predicate," whose domains are models and state-names, $P(\mathfrak{M},s)$ is interpreted as meaning that s is the name of the world that satisfies all the facts in $\mathfrak{M}$. One possible structure for $\mathfrak{M}$ is a set of ordered n-tuples, each of which represents some elementary relation; e.g., ⟨At, Robot, A⟩ and ⟨At, B1, B⟩ are elements of the initial model, $\mathfrak{M}_i$.

The initial world is defined by the axiom $P(\mathfrak{M}_i,S_o)$ (except that the complete known contents of $\mathfrak{M}_i$ must be explicitly given). We can now specify that an action changes a particular relation in $\mathfrak{M}$, and does not change any other relations, by a single axiom, e.g. the go action is defined by the axiom

$$(\forall x,y,\bar{w},s)\,[P(\{\langle At,Robot,x\rangle,\bar{w}\},s) \supset P(\{\langle At,Robot,y\rangle,\bar{w}\},go(x,y,s))] \quad .$$

Here $\bar{w}$ (read "w-bar") is a variable whose value is an indefinite number of elements of a set, namely all those that are not explicitly described.

This approach preserves the advantages of the previous state-variable approach; namely, the problem solving, answer construction, and other bookkeeping can be left to the theorem prover. In addition, properties of the model are automatically carried through state changes by the barred variables. On the other hand, several difficulties are apparent: theorem-proving strategies may be grossly inefficient in the domain of problem solving; the logic must be extended to include domains of sets and n-tuples; complex pattern-matching algorithms will be needed to compare expressions containing variables that range over individuals, n-tuples, sets, and indefinite subsets; and the fact that properties of the world are stored as data, instead of as axioms, constrains the problem-solving process by restricting the class of inferences that are possible. Further study is necessary to determine the feasibility of this approach.

D. Contexts and Context Graphs[7]

Suppose we let a state correspond to our intuitive notion of a complete physical situation. Since the domain of our logical formalism includes physical measurements such as object positions, descriptions, etc., every consistent statement of first-order logic is either true or false for every state. We think of each such statement as a predicate that defines a set of states, namely those for which it is true. We call such a set of possible states the context defined by the predicate.

We shall find it convenient to allow certain distinguished variables, called parameters, to occur in predicates. Since each such predicate with ground terms substituted for parameters defines a context, a predicate containing parameters may be thought of as defining a family of possible contexts--and each partial instantiation of parameters in the predicate defines a subfamily of contexts (or, if no parameters remain, a specific context).

For example, the predicate At(B1,B) defines a context (the set of all states) in which object B1 is at position B. If x and y are parameters, At(x,y) defines the family of contexts in which some object is located any place. At (B1,y) is a subfamily of this family in which the object B1 must be located at some (as yet unspecified) place.

A problem to be solved is specified by a particular predicate called the goal predicate. The problem, implicitly, is to achieve a goal state, i.e., produce any member of the context defined by the goal predicate.

An action will consist of an operator name, a parameter list, and two predicates--the preconditions K and the results R. In addition,

any of the elementary relations in the preconditions may be designated as transient preconditions. For example, the go action is defined by

$$\overbrace{\text{go}}^{\text{name}}\ \overbrace{(x,y)}^{\text{parameters}}$$

$$K\{\underline{At(Robot,x)} \mid At(Robot,y)\}R \ ,$$

where underlining designates a transient condition. Each action operator thus corresponds to a family of specific actions. An action is applicable in any state that satisfies K; when an action is applied, the resulting state no longer need satisfy the transients but must satisfy R.

In this approach, the conjunction of predicates in the robot's model of the world is an initial predicate I, defining as an initial context the set of all states that have, in common, all the known properties of the robot's current world. The goal context, defined by a given goal predicate, is the set of satisfactory target states. When an operator is applied in a context, it changes the defining predicate (roughly, by deleting transients and conjoining results), thereby changing the context. The problem-solving task is to construct a sequence of operators that will transform the initial context into a subset of the goal context.

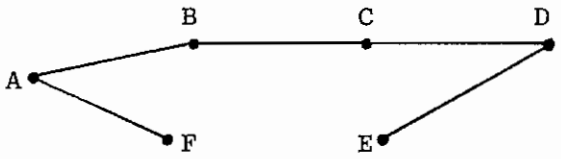
Any context that can be reached from the initial context by a finite sequence of operators is called an achievable context. Any context from which a subset of the goal context can be reached by a finite sequence of operators is called a sufficient context. The main task may be restated, then, as finding an operator sequence to show that the goal is achievable, or that the initial context is sufficient, or, more generally, that some achievable context is a subset of some sufficient context (and therefore is itself sufficient).

The main loop of the problem solver consists of two steps:

(1) Test whether any known achievable context is a subset of any known sufficient context. If so, we are done.

(2) Either generate a new achievable context by applying some operator in a known achievable context ("working forwards"), or generate, as a new sufficient context, one that would become a known sufficient context by the application of some operator ("working backwards"). Then return to step 1 to test the newly generated context.

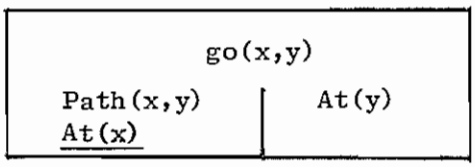
An advantage of this approach is that all states and all properties of operators are defined by first-order predicates, so a standard theorem-proving program can do most of the work of testing operators and results and selecting values of parameters. On the other hand, a separate data structure, called a context graph, is needed to keep track of the trees of achievable and sufficient states and the operators that relate their nodes. For example, suppose we wish to get from A to D in the directed graph:



We shall abbreviate by $\mathcal{G}$ the predicate that gives the graph's topology:

$$\mathcal{G} \equiv \text{Path}(A,B) \wedge \text{Path}(B,C) \wedge \text{Path}(C,D) \wedge \text{Path}(A,F) \wedge \text{Path}(E,D) \quad .$$

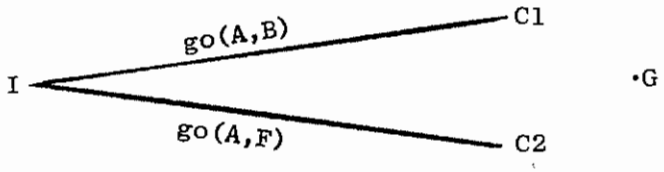
The initial predicate is $I \equiv \text{At}(A) \wedge \mathcal{G}$. The goal predicate is $G \equiv \text{At}(D)$. We shall define the operator go, for this problem, by:



The operator is applicable in context I only if we can prove that

$$(\exists x,y)[I \supset At(x) \wedge Path(x,y)]$$

is a theorem. The proof can be done by resolution with answer tracing.[5] The above statement can be shown to be a theorem when x = A and y is either B or F. Therefore, the go operator can be used two ways to generate new achievable contexts C1 and C2, with corresponding predicates $P_{C1} = \not\Diamond \wedge At(B)$, $P_{C2} = \not\Diamond \wedge At(F)$. To keep track of actions and instantiations, we shall draw the context graph:



Similarly, from C1 we can prove the applicability of go(B,C), which, when applied, gives C3

$$P_{C3} = \not\Diamond \wedge At(C) \quad .$$

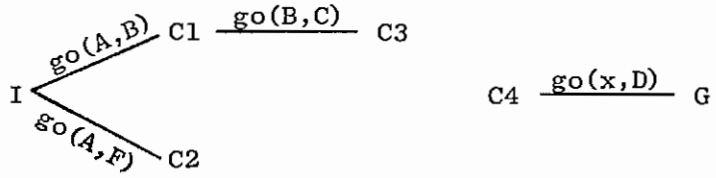
To illustrate working backwards, consider whether the result of a go implies G. The relevant problem for a theorem prover is

$$(\exists y)[At(y) \supset At(D)] \quad .$$

This is trivially true if y = D, so any state that satisfies the preconditions of the operator go(x,D) is sufficient (because the operator will then be applicable, and will produce the goal). Thus a new sufficient context is given by the preconditions,

$$P_{C4} = At(x) \wedge Path(x,D) \quad .$$

(Note that C4 is really a family of contexts, because of the parameter x.) The context graph is now:



Finally, the theorem prover can show that

$$P_{C3} \supset P_{C4} \text{ when } x = C \quad ,$$

completing the solution.

Most problems are considerably more difficult than the above example because of several complications. Suppose in trying to work backwards from G (using an operator Op with preconditions K and results R) we find that we cannot prove $R \supset G$, but instead discover a statement S such that $R \wedge S \supset G$. We may still work backwards with Op, but the new sufficient context is defined not by K alone but rather by $K \wedge S$. Furthermore, some extra bookkeeping must remind us that S may not be disturbed, in a valid solution, by applying Op--e.g., no transients of Op may appear in S. Similar additional subgoals--and bookkeeping complications--arise from each incomplete attempt to prove that an achievable context is contained in a sufficient context.

Additional complexities arise from dependencies. That is, when an expression E is deleted by a transient during an action, other expressions that were deduced from E in previous contexts can no longer be guaranteed to be true in new contexts. Thus each deduced expression is said to depend upon all its ancestors, adding to our growing burden of bookkeeping problems.

On the other hand, the context graph can take care of much of the bookkeeping automatically. Each logical expression need only be stored once, with notations telling in which contexts it was created and destroyed, rather than being either copied or rederived from context to context. Finally, if predicates of achievable contexts and operator results are stored in clause form, and predicates of sufficient contexts and operator preconditions are stored in negated clause form, preliminary experiments show that most of the nuts-and-bolts work of attempting solutions and generating new contexts can be done in a straightforward manner by an existing resolution-type theorem-proving program.

E. Other Approaches

Several other approaches to the frame problem have been suggested, although few have been worked out in sufficient detail to test on a computer.

Richard E. Fikes at SRI is developing a system whose formal framework is similar to that of D above ("contexts and context graphs"), but which does not use resolution techniques. Instead, proofs are strongly dependent upon the semantics of the logic, and the problem solver proceeds by a heuristic, goal-directed, case-analysis approach. This work is still in an early stage of development.

Eric Sandewall at Stanford is extending some ideas suggested by John McCarthy for formalizing the concepts of causality and time dependence, using a method proposed by J. Alan Robinson for embedding higher order logic in first-order predicate calculus. The resulting system provides an interesting model for inevitable sequences of events

(e.g. "if it is raining then things will get wet,") but may not be as useful for describing alternative possible actions by an external agent (e.g., the robot).

Methods for proving theorems in higher order predicate calculus are being developed in several places, and the use of this more powerful formalism may eventually vastly simplify our tasks. Finally, McCarthy and Hayes[1] suggest some other approaches including modal logics and counterfactuals, but the details have not been extensively explored.

IV CONCLUSIONS

This paper has described the frame problem and the principal methods that have been proposed for solving it.

Let us review the approaches listed above. A, complete frame and frame-transition descriptions, was simply a stage-setting "straw man" that we would not consider actually using. B, the logic-cum-state-variable approach, is beautifully elegant for "toy" problems, but both the representational effort and the theorem-proving effort grow explosively with problem complexity. C, the world predicate idea, preserves some of the elegance of approach B while carrying along necessary frame information implicitly; however, it places a burden on theorem-proving abilities in new domains and requires an awkward use of two levels of logical representation (that is, relations among the n-tuples in the model must be defined in terms of the world predicate), so that the practicality of the approach is open to serious question. Approach D, the use of contexts and context graphs (without explicit state names in the logic), is a more-or-less brute-force attempt to combine the use of first-order theorem-proving methods with a GPS-like structure of subgoals and operators;

although the bookkeeping problems are complicated, they seem to be tractable, so that the approach is reasonably promising. Finally, under E we mentioned several interesting ideas that warrant further exploration before they can be meaningfully compared with the other approaches.

Until now most research in problem solving has dealt with fairly static situations in narrow subject domains. As we become interested in building complete artificially independent systems, a new kind of problem-solving research emerges: We must study how to solve problems in an environment containing a large store of knowledge, while considering the possible effects of a variety of sequences of actions. This paper has described some of the first exploratory steps into this important area of research.

V ACKNOWLEDGMENT

Many of the ideas discussed above, including development of the contexts and context graphs approach, are the result of joint discussions between the author and L. S. Coles, R. E. Fikes, J. H. Munson, and N. J. Nilsson. The "world predicate" idea is completely due to R. Waldinger. C. C. Green developed the state variable approach, and he is largely responsible for our early realization of the importance of the frame problem.

The research described herein is supported by the Advanced Research Projects Agency of the Department of Defense, and by the National Aeronautics and Space Administration under Contract NAS12-2221.

REFERENCES

1. McCarthy and Hayes, "Some Philosophical Problems from the Standpoint of AI," Machine Intelligence 4, B. Meltzer and D. Michie, eds. (Edinburgh University Press, Edinburgh, Scotland, 1969).

2. B. Raphael, "The Relevance of Robot Research to AI," in Formal Systems and Non-Numeric Problem Solving by Computer (Springer-Verlag, 1970).

3. G. W. Ernst and A. Newell, GPS: A Case Study in Generality and Problem Solving (Academic Press, New York, N.Y., 1969).

4. C. C. Green, "Application of Theorem Proving to Problem Solving," Proc. International Joint Conference on Artificial Intelligence, Washington, D.C., May 7-9, 1969.

5. C. Green and B. Raphael, "The Use of Theorem-Proving Techniques in Question-Answering Systems," Proc. 1968 ACM Conference, Las Vegas, Nevada (August 1968).

6. R. Waldinger, "Robot and State Variable," TN 26, Artificial Intelligence Group, Stanford Research Institute, Menlo Park, California (April 1970).

7. B. Raphael, "Robot Problem Solving without State Variables," TN 30, Artificial Intelligence Group, Stanford Research Institute, Menlo Park, California (May 1970).