TR-30

INFORMATION PROCESSING 75 – NORTH HOLLAND PUBLISHING COMPANY (1974)

from: I.F.I.P. Congress '74, Stockholm

(sic!) Yes ahead of our times!

COLLABORATIVE PROBLEM-SOLVING BETWEEN OPTIMISTIC AND PESSIMISTIC PROBLEM SOLVERS

L. SIKLÓSSY and J. ROACH

University of Texas
Austin, Texas, USA

An optimistic problem-solver assumes that a problem has a solution and attempts to find such a solution. A pessimistic problem-solver assumes that a problem has no solution, and tries to prove this lack of a solution. When one of the problem-solvers fails to achieve its goal, it is an indication that the other problem-solver may succeed. Moreover, information may be extracted from the failure to help the other problem-solver in its success. In such a case, the two complementary systems are said to collaborate.

We give examples of collaboration between an optimist, LAWALY, and a pessimist, DISPROVER, which operate on worlds of simulated robots. When collaborating, each system can solve more problems than if it worked alone.

1. INTRODUCTION

Given a problem, one of two hypotheses may be adopted. The problem may be assumed solvable, and an attempt made to solve it: we call this hypothesis optimistic. Or the pessimistic hypothesis may be adopted: the problem may be assumed unsolvable, and an attempt made to prove that no solution exists.

If the optimistic problem-solver fails to find a solution, it may be an important indication that the problem has no solution, and therefore that the pessimist will succeed. Similarly, if the pessimistic problem-solver fails to show that the problem has no solution, it may be a significant indication that a solution exists, and perhaps the optimist will find it.

If the optimist fails, it may not only report its failure to the pessimist, but also transmit enough information to the pessimist to allow it to succeed. If the pessimist could not have succeeded without the optimist's help, we can say that the two systems have collaborated. Similarly, if the pessimist fails, it may transmit to the optimist some information which will allow the optimist to find a solution. Without the additional aid from the pessimist the optimist would fail.

In this article, we describe some results of collaboration between an optimist, LAWALY [1], and a pessimist, DISPROVER [2]. Both take as input problems in simulated robot worlds, and use an axiomatization which is a generalization of that first introduced in [3]. Other robot problem solvers based on the same axiomatization are described in [4 - 9] while [10] and [11] propose different axiomatizations.

Collaborative programs have been described in computer vision [12], and in natural language processing (for example [13]) where the syntactic and semantic components of the language help each other. However, the collaboration between a prover, LAWALY, and a disprover, DISPROVER, is novel to the best of our knowledge.

We shall give an all too brief description of LAWALY and DISPROVER, emphasizing the difficulties which they may encounter. The difficulties may be due to a lack of resources --when the problem-solver cannot achieve a solution due to limited resources of time and memory-- or it may be absolute --when the problem-solver does not exhaust its resources but still cannot solve the problem.-- We shall give examples of collaboration in which difficulties of both types are overcome.

*Partially supported by grant GJ-34736 from the National Science Foundation

2. DESCRIPTION OF THE WORLD AND OPERATORS

The simulated robot worlds are described by a set of true predicates. Two small worlds are represented in fig. 1. The INITIAL world would be described by the predicates:

(INROOM ROBOT ROOMA) (CLOSED DOORAB)
(INROOM BOX ROOMB)
(JOINS DOORAB ROOMA ROOMB).

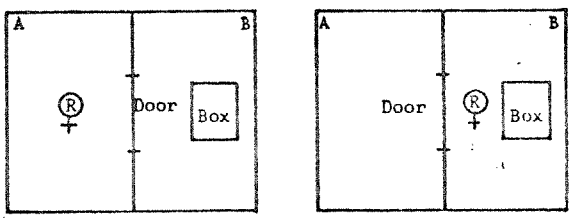

Fig. 1.

The world can be changed by applying to it an operator. An operator can be applied to a world only if the world satisfies the preconditions of the operator. The changes of the world as a result of the application of the operator result from deleting from the world the delete set of the operator, then adding to the resultant world the add set of the operator. For example, to open a door, the operator "opendoor" may be used. To opendoor(door), the robot must be near the door, and the door must be closed. After she has opened the door, the statement (CLOSED door) is deleted from the representation of the world, and the statement (OPEN door) is added.

A task is given as a set of conditions which must be simultaneously true in a final state. A goal could be (the FINAL state in fig. 1):

(CLOSED DOORAB) (NEXTTO ROBOT BOX).

p. 811

A solution to the task is a sequence of operators which transform the initial world into a world in which all the conditions of the task are satisfied.

3. BRIEF DESCRIPTION OF LAWALY

LAWALY's search towards a goal is guided, for our purposes, by two main components: an ordering of subgoals according to a hierarchy, and a selection of appropriate operators to accomplish a subgoal. (For descriptive purposes we shall make use of some hypothetical goals.)

3.1 Hierarchy of subgoals

Some subgoals should be accomplished before others. If the goal is (INROOM ROBOT ROOMG) (INROOM PAIL ROOME) (STATUS TERRARIUMD WATERED), clearly the TERRARIUMD should be watered first, next the PAIL should be carried to ROOME, and finally the robot should go to ROOMG. Trying to accomplish the subgoals in any other order would not work, since the pail is needed to water the terrarium, and the robot must carry the pail.

The hierarchies of subgoals can be either input by hand, or generated by a program from the description of the operators.

3.2 Selection of operators

For a given subgoal, some operators are more appropriate than others. To water TERRARIUMD, the operator irrigate(TERRARIUMD) will be used. To obtain the condition (INROOM ROBOT ROOMG), the robot could carry a trashbasket into ROOMG, but instead should prefer to just go there. LAWALY selects the most appropriate operator(s) for a subgoal.

3.3 LAWALY's difficulties

LAWALY encounters two types of difficulties that can be surmounted with the help of DISPROVER:

(i) Large class of subgoals with the same hierarchy

If several subgoals have the same hierarchy, LAWALY does not know in which order they should be achieved. Attempting all the permutations in the order of the subgoals would be far too expensive in terms of computer time if the number of subgoals is large. This difficulty is due to resources.

(ii) Stubbornness

LAWALY's power and efficiency --she has routinely solved problems requiring several hundred steps-- comes from focusing on one subgoal at a time. Such focusing may also be called stubbornness. We have here an example of an absolute difficulty.

For example, the problem of fig. 1 (as described in section 2) is not solved by LAWALY. LAWALY may decide to work first on the (CLOSED DOOR) condition, or first on the (NEXTTO ROBOT BOX) condition. Consider the first case. LAWALY finds the door already closed in the initial state, so she wants to obtain the NEXTTO condition. To do that, she must enter Room B, and to do that go through the DOOR. But that would mean opening the DOOR, and hence undoing what she had already achieved, (CLOSED DOOR), and so she decides to try the conditions in the reverse order. To be (NEXTTO ROBOT BOX), she goes to DOOR, opens it, goes through it, and then goes next to BOX. At that point, she realizes that she must still close the DOOR. However, that would make her undo something she wanted, namely (NEXTTO ROBOT BOX), so she quits, having failed. (Stubbornness is not a problem with an executing robot. [8])

4. BRIEF DESCRIPTION OF DISPROVER

DISPROVER is an implementation of the technique of hereditary partitions [2], a generalization of the technique of hereditary properties [14], to disprove statements about robot-like worlds. DISPROVER can be applied to find disproofs in any system which describes successive states by a set of true and false conditions, independently of the form of the operators that transform one state into another.

An example will help to illustrate the working of DISPROVER. We take the world of fig. 2 (as used in [1,3,7]), and change it so that if the robot goes through the door to the room containing the location G, then the light turns off. To turn the LIGHTSWITCH1 on, the robot must be (ON ROBOT BOX1) and the BOX1 must be (NEXTTO BOX1 LIGHTSWITCH1). We wish to show that the task: (STATUS LIGHTSWITCH1 ON) (AT ROBOT G), is impossible, i.e., we wish a disproof of this goal. (The goal description may include some negated predicates.)

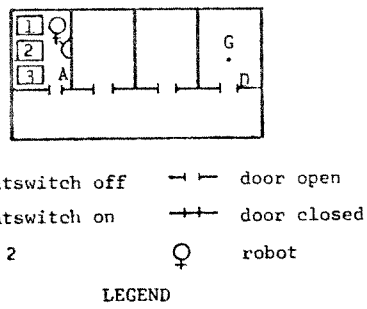

Fig. 2. A robot world.

DISPROVER uses two sets of predicates for disproofs:
a) anchor predicates, which are always the predicates (made positive if necessary) in the statement of the problem to be disproved. Here the anchor predicates are P1 = (STATUS LIGHTSWITCH1 ON); P2 = (AT ROBOT G).
b) refinement predicates, which are additional predicates used in the disproof.

The basic idea of DISPROVER is to disregard everything in the world except the anchor and refinement predicates. In this way, the world is significantly reduced from the initial world. This reduced world is often so small that it is feasible to enumerate all possible states that can be reached from the initial reduced world by the application of all operators (appropriately reduced also). Each reduced state that can be obtained in this way is called a partition. Since the set of partitions is finite, two cases can occur:
-the goal is not a subset of any partition. This means that the goal cannot be reached (from the original partition) in the simplified, reduced world. Hence, the goal cannot be reached in the complete world from the original world. DISPROVER succeeds.
-the goal is a subset of some partition. This means that the goal can be reached in the simplified world, but says nothing about whether it can be reached in the complete world.

Two examples will help to clarify the workings of DISPROVER.

4.1 Disproof attempt with one set of refinement predicates

First let us assume that the set of refinement predicates is empty. Let us demonstrate how the

disproof fails. The initial state is reduced to the initial partition, and is described by -P1 -P2 (since neither P1 nor P2 is true in the initial state). The robot can go to G, and thus we obtain a new partition -P1 P2. The conditions -P1 P2 do not violate the preconditions of the operator turnonlight(lightswitch), which could be (in some axiomatization): (TYPE lightswitch LIGHTSWITCH) (ON ROBOT BOX1) (STATUS lightswitch OFF) (NEXTTO BOX1 lightswitch) (INROOM ROBOT rm) (INROOM lightswitch rm).
Hence in the reduced world, we can apply the operator turnonlight(LIGHTSWITCH1) to the partition -P1 P2, to obtain the partition P1 P2. Since this partition contains the goal to be disproved, the disproof fails: we do not know whether the goal can be achieved or not.

4.2 Disproof attempt with another set of refinement predicates

Let us now attempt the disproof with the refinement predicates P3=(INROOM ROBOT ROOMA), P4=(INROOM ROBOT ROOMD). The initial partition is -P1 -P2 P3 -P4. The complete disproof is too long to be inserted here, but it should be clear that it will succeed (assuming a decent axiomatization of the robot). In particular, the disproof can no longer fail as before. If the robot goes to G, it will be in the partition -P1 P2 -P3 P4. The -P3 violates the preconditions for turning on the light, so that the turnonlight operator may not be applied to this partition.

4.3 DISPROVER's difficulties

The above example makes obvious the difficulties of DISPROVER. A disproof may fail because the set of refinement predicates is too small, or inadequate, to correctly partition the world to yield a disproof. On the other hand, if there are too many refinement predicates, the enumeration of partitions can become too time consuming and require too much memory, and the difficulty is due to the lack of resources.

5. LAWALY AIDED BY DISPROVER

Now that we have examined LAWALY and DISPROVER, we turn to examples of their collaboration.

5.1 Large class of subgoals with the same hierarchy

When LAWALY encounters a problem containing a large class of subgoals (more than three subgoals), it tries one permutation of the subgoals (for luck). If LAWALY fails to find a solution, the problem is passed to DISPROVER, without refinement predicates.

If the problem has a solution, DISPROVER will fail, since it cannot disprove a problem which has a solution. The unsuccessful disproof indicates that there is a path from the initial partition to the goal. Each path represents a certain order in which the subgoals have been added to the partitions, until they are all present in the partition containing the goal. Each path therefore represents a permutation of the subgoals in which the reduced problem can be solved. Let us call such a permutation a feasible reduced permutation, while a permutation of the subgoals in which the original, not reduced, problem can be solved is called a feasible permutation. Clearly, every feasible permutation is a feasible reduced permutation, but not vice-versa. LAWALY will attempt to find a solution for the order given by every feasible reduced permutation.

(i) An example

Usually, the use of hierarchies of subtasks splits subgoals into levels of the same hierarchy such that each level has few subtasks. To illustrate the help offered by DISPROVER (or heuristic DISPROVER, section 5.1(ii)) in the case of a hierarchic level containing many subgoals, we must manufacture a somewhat artificial example. The example must exhibit multiple interactions among the subgoals. Indeed, if the subgoals did not interact strongly, they could be placed in separate hierarchical levels.

In a certain robot world, rooms A, B and C (denoted (A ROOM), (B ROOM) and (C ROOM)) are on floor 1 and communicate through Hall 1 (denoted (1 HALL)). Rooms D and E are on floor 2; they communicate through Hall 2. An elevator joins both floors. We consider the goal: 1-(INROOM TRASHCART (B ROOM)), 2-(HOLDING BROOM), 3-(STATUS (1 HALL) CLEAN), 4-(STATUS (1 ELEVATORDOOR) OPEN), 5-(STATUS (1 TRASHBASKET) EMPTY), 6-(STATUS (D DOOR) OPEN).
In the particular representation, all six subgoals belong to the same hierarchy. The axiomatization of the world is such that of the 6! = 720 permutations, only four are feasible. The subgoals must be performed in the order: 5, 2, [3,6 or 6.3], [4,1 or 1,4]. One of the solutions is 32 steps long. To guarantee no other solutions, certain operators have added unusual side effects. For example, to assure that 4 will not be accomplished before 5, 2, 3 and 6, the operators which would accomplish 5, 2, 3 or 6 would also undo 4. That is, the elevator door on floor 1 closes when trashbasket 1 is emptied, or when the robot picks up the broom, or when Hall 1 is swept, or when Door D is opened!!

DISPROVER fails, but from its failure four feasible reduced permutations are extracted, each one allowing LAWALY to solve the problem. The disproof took 130 seconds in interpreted LISP on the CDC 6600. This time is significantly smaller than the time needed by LAWALY to try all 720 permutations.

(ii) Heuristic DISPROVER

A modification of DISPROVER, called Heuristic DISPROVER, accelerates the finding of feasible reduced permutations, although it is no longer a disprover, nor does it guarantee that all feasible reduced permutations will be found.

The idea is simple. From any partition we only build those partitions which represent a positive advance towards the goal. For example, from the initial partition Par1: -P1 -P2 -P3 -P4 -P5 -P6, we might be able to build partition Par2: -P1 P2 -P3 -P4 -P5 -P6. If an operator op1 applied to Par2 results in a partition Par3: -P1 -P2 P3 -P4 -P5 -P6, then Par3 will not be kept. On the other hand; if another operator op2 applied to Par2 yields the partition Par4: -P1 P2 -P3 -P4 P5 -P6, then Par4 will be kept.

With the same problem as in section 5.1(i), heuristic DISPROVER generated the same four feasible reduced permutations in 22.8 seconds.

(iii) Failure of the cooperation

The cooperation will fail if DISPROVER returns too many feasible reduced permutations, all of which LAWALY cannot try due to time limitations.

5.2 Stubbornness

Section 3.3(ii) described a solvable problem which LAWALY fails to solve. The problem is given to DISPROVER, which fails, as it should since the problem is solvable. DISPROVER generates a partition 4: (NEXTTO ROBOT BOX) (CLOSED DOOR), by applying the operator gonext(object) to the partition 1: -(NEXTTO ROBOT BOX) (CLOSED DOOR).

The preconditions of the operator are: (ONFLOOR) (INROOM ROBOT room) (INROOM object room). In the operator, "object" is bound to BOX, and "room" to B, hence the preconditions represent a state sti: (ONFLOOR) (INROOM ROBOT B) (INROOM BOX B). DISPROVER suggests to LAWALY that the original problem could perhaps be solved by splitting it into two successive problems. The first problem is to go from the initial state to a state containing the intermediary state sti above; the second problem is to go from there to the final state. LAWALY does in fact solve the original problem in this way.

(i) Failure of the cooperation

The collaboration could continue, in the sense that DISPROVER could keep suggesting additional intermediary states. The collaboration will fail when DISPROVER can suggest no new intermediary state, or suggests an unreachable intermediary state.

6. DISPROVER AIDED BY LAWALY

When LAWALY tries to find a solution to an impossible task, she will fail. She keeps track of the preconditions of all of the operators that she has applied on each of her unsuccessful attempts at a solution. Thus, for each attempt, we obtain a candidate set of refinement predicates. A disproof is attempted with each candidate set. If all these disproofs fail, all the candidate sets are merged into one (huge) set of refinement predicates, and another DISPROOF is attempted.

In the example of section 4, when attempting to solve the problem in the order (STATUS LIGHTSWITCH1 ON), (AT ROBOT G), LAWALY fails. On her path she picks up enough predicates to provide an adequate, but certainly not minimal, set of refinement predicates.

6.1 Failure of the cooperation

LAWALY may not provide DISPROVER with an adequate set of refinement predicates. Our experience has been that LAWALY finds the refinement predicates necessary for a disproof, since she picks up just about everything along a promising solution path. The problem is that generally she picks up far too many refinement predicates, which slow down DISPROVER to the point where a disproof is sometimes impossible with the available resources of time. We must teach LAWALY to be more selective in the help that she gives DISPROVER.

7. CONCLUSION

We have given examples of the difficulties that an optimistic problem-solver, LAWALY, and a pessimistic problem-solver, DISPROVER, experience when tackling problems in simulated robot worlds. In many cases the optimist can help the pessimist and the pessimist can help the optimist, in such a way that the two systems working together can solve problems which neither system alone can solve.

If the robot worlds considered are appropriately enlarged, it is not possible to design an algorithm which can decide, for each problem, whether it has a solution or not. Therefore, it cannot be expected that the collaboration between LAWALY and DISPROVER (or their descendants) will ever be perfect. Some of the weaknesses of their present modes of collaboration, however, point to areas where improvements can be achieved. It would be useful, moreover, to develop a more continuous collaboration: presently, the collaboration must wait for one system to terminate with a partial or total failure. The failure of DISPROVER can be ascertained only when the system has run its full course. On the other hand, LAWALY may encounter difficulties on the way to a solution. These difficulties could prove helpful to DISPROVER, even though LAWALY has not yet exhausted all its resources.

REFERENCES

[1] Siklóssy, L. and Dreussi, J., An Efficient Robot Planner which generates its own Procedures, Third International Joint Conference on Artificial Intelligence, Palo Alto, California, 1973.

[2] Siklóssy, L. and Roach, J., Proving the Impossible is Impossible is Possible: Disproofs based on Hereditary Partitions, Third International Joint Conference on Artificial Intelligence, Palo Alto, California, 1973.

[3] Fikes, R. E. and Nilsson, N. J., STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, Artificial Intelligence, vol. 2, no. 3/4, 1971, 189-208.

[4] Fikes, R. E., Hart, P. E. and Nilsson, N. J., Learning and Executing Generalized Robot Plans, Artificial Intelligence, vol. 3, no. 4, 1972, 251-288.

[5] Fikes, R. E., Hart, P. E. and Nilsson, N. J., New Directions in Robot Problem Solving, in: Michie, D. and Meltzer, B. (Eds.), Machine Intelligence 7, Edinburgh University Press, Edinburgh, Great Britain, 1972.

[6] Sacerdoti, E., Planning in a Hierarchy of Abstraction Spaces, Third International Joint Conference on Artificial Intelligence, Palo Alto, California, 1973.

[7] Siklóssy, L., Modelled Exploration by Robot. Technical Report TR-1, Computer Sciences Department, University of Texas, Austin, 1972.

[8] Siklóssy, L. and Dreussi, J., Simulation of Executing Robots in Uncertain Environments, Technical Report TR-16, Computer Sciences Department; University of Texas, Austin, 1973.

[9] Siklóssy, L. and Roach, J., Model Verification and Improvement using DISPROVER, Technical Report TR-26, University of Texas, Austin, July, 1973.

[10] Hendrix, G., Beyond Omnipotent Robots, Technical Report NL-14, University of Texas, Austin, 1973. (To appear under the title: Modeling Simultaneous Actions and Continuous Processes, in Artificial Intelligence Journal.)

[11] Siklóssy, L. and Dreussi, J., Robot problem-solving with negative goals, Technical Report TR-23, University of Texas, Austin, 1973.

[12] Winston, P. H. Scene Understanding Systems, in: Watanabe, S. (Ed.) Frontiers of Pattern Recognition, Academic Press, New York, 1972.

[13] Winograd, T. Procedures as Representation for Data in a Computer Program for Understanding Natural Language, Technical Report AI TR-17, M.I.T., 1971.

[14] Simon, H. A., On Reasoning about Actions, in: Simon, H. A. and Siklóssy, L. (Eds.) Representation and Meaning, Prentice-Hall, Englewood Cliffs, N. J., 1972.