A REPRESENTATION FOR ROBOT PLANS

Philip J. Hayes
Institute for the Study of Semantics and Cognition
6976 Castagnola, Switzerland

Abstract

A representation for robot plans is proposed. The representation of a given plan reflects the structure of the process which produced the plan. This information is useful in both the original construction of the plan and its subsequent modification if unforeseen events cause execution failures. A programmed system that constructs and executes (in simulation) plans for journeys using a large system of public transport is described to illustrate the advantages of the representation. It is also shown how the representation could be used for more typical robot-planning worlds.

1. Introduction

Since most robots must function in a world whose behaviour they cannot hope to predict exactly/ they must be prepared for plans they make to fail during execution. (As usual, a plan is a linear sequence of actions or operations intended to transform some initial state of the robot's world into some goal state.) Such failures may occur because some action in the plan fails to have its expected effect or because of some unpredictable event outside the control of the robot. In any case, after such a failure the robot must be able to modify its plan to cope with the unexpected turn of events.

Rather than throw away the original plan and start again from scratch, it is clearly desirable that as much as possible of the work that went into constructing the old plan should be reused in producing a modified version. One previous system, STRIPS[1,2], that dealt with replanning after execution failures, tried to save work by making arbitrary subsequences of operations from the original plan available to the replanning process as primitive operations. No attention, however, was paid to the way in which such subsequences were originally intended to contribute to the fulfillment of the goal of the plan, thus, at times, leading to rather arbitrary uses of them. Such a scheme does not try and use any of the problem analysis that went into constructing the original plan but only its results.

More hierarchically structured systems have proposed that plans should be sketched out at some level of detail abstracted from that of the robot's primitive actions, and each step fully detailed only as it is executed. Such an approach, while giving an inbuilt flexibility with regard to details, runs the risk of disaster if some of the unelaborated steps turn out to be impossible because of some complication at the more detailed levels.

The scheme of plan representation presented in this paper is primarily designed to facilitate the reconstruction of detailed plans after failure in execution. It does this by explicitly recording in the representation of a particular plan the structure of the process which produced that plan. The information thus recorded includes the choices made during the construction of a plan, how they advanced the construction of the plan in terms of sub-goaling and refinement of details, and how they are logically related to each other. After failure this information enables that part of the development of a plan due to decisions invalidated by the circumstances of failure to be precisely identified and discarded. The resulting data structure represents that part of the original problem analysis which is still applicable after the failure*. Use of this structure by the replanning mechanism will avoid the corresponding part of the original planning effort being duplicated during replanning.

While the representation makes as few assumptions as possible about planning processes, it is basically oriented to processes based on hierarchical levels of detail[3,4,5]. By making information about all steps of a plan constantly available, the representation facilitates plan making for worlds in which the effects of operators tend to interfere with each other to a significant extent. It also permits different parts of the execution sequence to be developed in an order and to relative levels of depth, dependent only on the constraints of the problem domain. Such an ability is useful when there are different certainties as to whether the different steps of an undetailed skeleton plan can be successfully developed to a detailed level.

Use of the representation forms the basis of a working robot planning and (simulated) execution system described in section 3. Some aspects of the system implemented are atypical of robot planning domains in general, and so a discussion of how the representation would be used with a more usual robot world follows in section 4.

2. The Representation

A plan in the proposed representation consists of two interlinked data structures: a tree which represents the subgoal structure of the plan and a graph which represents the logical relationships of the decisions taken in constructing the plan. This representation and its uses are described below in general terms; for detailed examples see section 3.

*In fact it does not always represent all of it, since there can be decisions that are still appropriate after failure, but are discarded because they were originally based on one of the decisions invalidated by the failure.

Each node of the subgoal tree of a plan (subsequently inode) corresponds to a goal and the action necessary to achieve it. (These two concepts are sufficiently close that it is often useful to blur any distinction between them.) Subgoaling is represented by the branching of the tree (with a left to right time ordering). Thus

would indicate that goal X has been split into subgoal Y and subgoal Z (or equivalently that the action required to achieve X can be split into an action that will achieve Y followed by an action that will achieve Z). The root of the tree corresponds to the top-level goal of the plan, while in a complete plan the tips of the tree correspond to primitive actions. For a system using a hierarchy of levels of detail, progress along a branch from root to tip would thus be naturally accompanied by an increasing amount of detail.

Besides just representing a goal or action, each jnode can contain any other information about that goal or action that is helpful to the plan construction process. In particular, information about the expected state of the world before or after the action could be thus represented. Communication and co-operation between processes constructing different steps in the same plan can be greatly facilitated by having such information for every step in the plan constantly available.

The nodes of the decision graph of a plan (subsequently dnodes) are in one-one correspondence with the decisions made during the construction of that plan. The parent-child relation of the graph indicates logical dependence of the child on the parent. Logical dependence of one decision on another here means that the process which made the second decision was influenced by some direct consequence of the first. A graph structure is necessary to represent such logical dependencies, because the effects of two quite independent decisions can influence the making of a third.

Each dnode has two-way pointers from it to those jnodes created as a direct consequence of its decision. These pointers can be used in conjunction with a graph structure to precisely identify all the effects both direct and indirect of a decision on the development of a plan. The process of removing the effects (thus identified) of a decision from a plan is known as UNDOING the decision and consists of:

a) removing the decision from the graph and all the jnodes pointed to by the decision from the tree;

b) UNDOING the children of the decision in the graph.

This UNDO mechanism forms the basis of the method for reconstructing plans after failure in execution. The basic idea is to identify the most logically senior decisions inappropriate to the unexpected situation, UNDO them, and then use what is left of the original plan representation as a basis for plan reconstruction. To this end, each decision must have a resumption point of the plan construction process associated with it. Use of this resumption point should result in the plan construction process being reentered to remake the decision in the light of all currently available information.

In more detail the replanning mechanism works as follows. When an execution failure occurs, the execution monitoring system is assumed to designate a set of jnodes as unexecutable, these jnodes being the most senior that could be so designated. Then:

1. all portions of the subgoal tree successfully executed are discarded, together with any dnodes directly responsible only for discarded jnodes,
2. all information in the tree, made inaccurate as a result of the circumstances of failure, is updated,
3. for each jnode, A, of the unexecutable set
   a) if A has already been removed from the tree nothing more is done, otherwise
   b) the dnode, D, directly responsible for A is UNDONE,
   c) the plan construction process is reentered through the resumption point of D,
   d) if the reentered plan construction process terminates successfully nothing more is done, otherwise
   e) if A is not the root of the tree, it is replaced by its parent and step b) is looped to, otherwise
   f) replanning is assumed to be impossible.

If this mechanism is always to produce a complete plan, the resumption point associated with a decision must not only remake its own decision, but also any other dependent decisions needed to produce a complete plan. Such an arrangement may not always be feasible. In such cases those jnodes on the frontier of the subgoal tree that were not primitive at the end of the above process would have to be found and developed until they were. This problem does not, however, arise with the system to be described in the next section.

3. A Working System Based on Use of the Representation

The representation described above forms the basis of a working system for the construction and (simulated) execution of plans. The plans concern the journeys of a (robot) traveller through a network of rail, sea and air public transport services. The system contains a considerable amount of knowledge in procedural form about how to make plans in such a domain. It is not, however, tied to a particular network, but accepts the definition of such a network in tabular form as initial data. This definition comprises lists of connections, timetables for those connections and some geographical data (concerning the positions of towns and the relative positions of countries and seas).

The network used for example purposes is extensive. It connects 84 European centres using 43 rail services, 7 sea services and 55 air services giving in excess of 500 town to town connections and more than 2000 primitive journeys (journeys between two centres using one service at a particular time). The services used are all taken from real world timetables- The geographical area covered by the network is shown in figure 1. For the sake of readability, only a small subset of the network is shown, but this subset includes all the routes used in the examples.

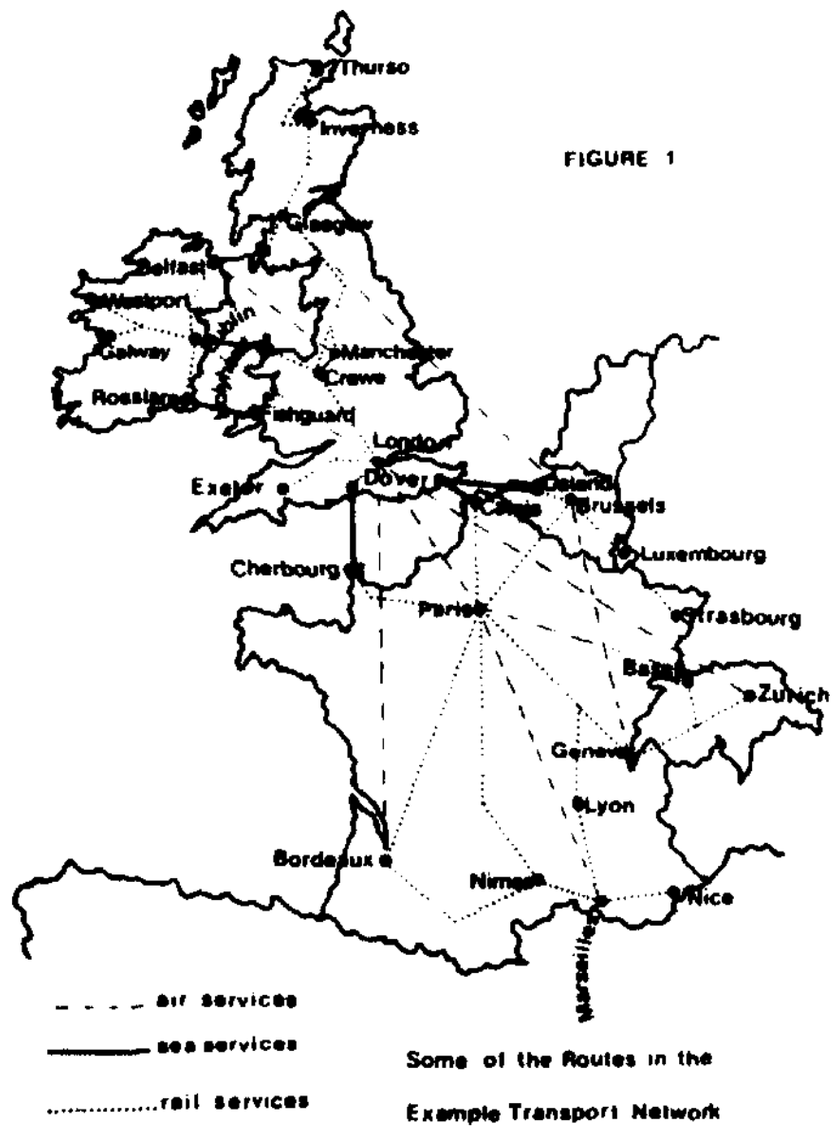

FIGURE 1

Some of the Routes in the Example Transport Network

The system can construct plans subject to certain constraints of time and cost. These constraints can be both global (e.g. on the overall cost of a journey) and local (e.g. on the time of arrival at a particular place). The system can also cope with the occurrence of "unexpected" events during the simulated execution of a journey. It checks their relevance to the journey and if necessary modifies the plan using the replanning mechanism described in the previous section.

The system uses its domain dependent knowledge to good effect and a planning time of around 20 seconds is typical for a journey involving six primitive journeys. The plan produced is "reasonable" though not necessarily optimal. The system was coded in POP-2 on an ICL 4130 computer.

Figure 2 shows the subgoal tree and decision graph of a plan produced by the system for going from Manchester to Nice by train and boat. (This plan is, in fact, optimal with respect to both time and cost for a journey in the example network from Manchester to Nice using surface transport.)

Because of the nature of the domain, all jnodes correspond to journeys of varying degrees of specification. The correspondence between increase in detail and increase in depth of the tree will be clear. Times are represented by a 24-hour clock time plus a day, thus 17:30 2 means 5:30 p.m. on day 2 of the journey. The list of numbers in brackets associated with each dnode indicates which jnodes were created as a direct result of each dnode.

The function of each decision shown in figure 2 (b) is as follows:

Decision 0 is a dummy decision which is notionally responsible for all the unchangeable features of the world.

Decision 1 chooses surface transport as the mode of travel. Because a sea, the English Channel, lies between Manchester and Nice, it results in subgoaling the original goal into three subgoals: to get to the Channel, to cross it, and to get to Nice.

Decision 2 chooses the route for the Channel crossing and thus refines jnodes 7, 3 and 4 into the more detailed jnodes 6, 5 and 7 respectively.

Decision 3 fixes a time for the sea crossing, refining jnode 5 still further. Since the sea crossings are rarer than the rail connections, unnecessary waiting is minimized if rail times are fitted to sea times rather than the other way round. The situation after this decision is an example of the concurrent use of differing levels of detail for different steps of the plan.

Decision 4 fixes the route for the train journey as far as the Channel crossing, but is independent of decision 3 even though made after it. Decision 7 is similar.

Decision 5 fixes a time for the train journey immediately before the Channel crossing. It depends on a routing decision (4) and a timing decision (3) which are independent, but an alteration in either would require the reconsideration of decision 5. Decision 8 is similar.

Decisions 6, 9 and 10 are other timing decisions which depend on a routing decision and its dependent timing decision.

The domain dependent expertise of the planning system is encoded in a number of special functions called dfunctions. Each dfunction has its own special area of knowledge and is associated with the making of a particular sort of decision. Df unctions are parameterless and are expected to extract all the knowledge they require from the decision graph and subgoal tree representing the existing state of the plan.

There are seven main dfunctions:

TRYSORP which chooses between air and surface transport,

TRAINS, BOATS, and PLANES which choose routes for rail, sea and air journeys respectively,

TRAINTIME, SEATIME and PLANET1ME which choose times for journeys by rail, sea and air respectively.

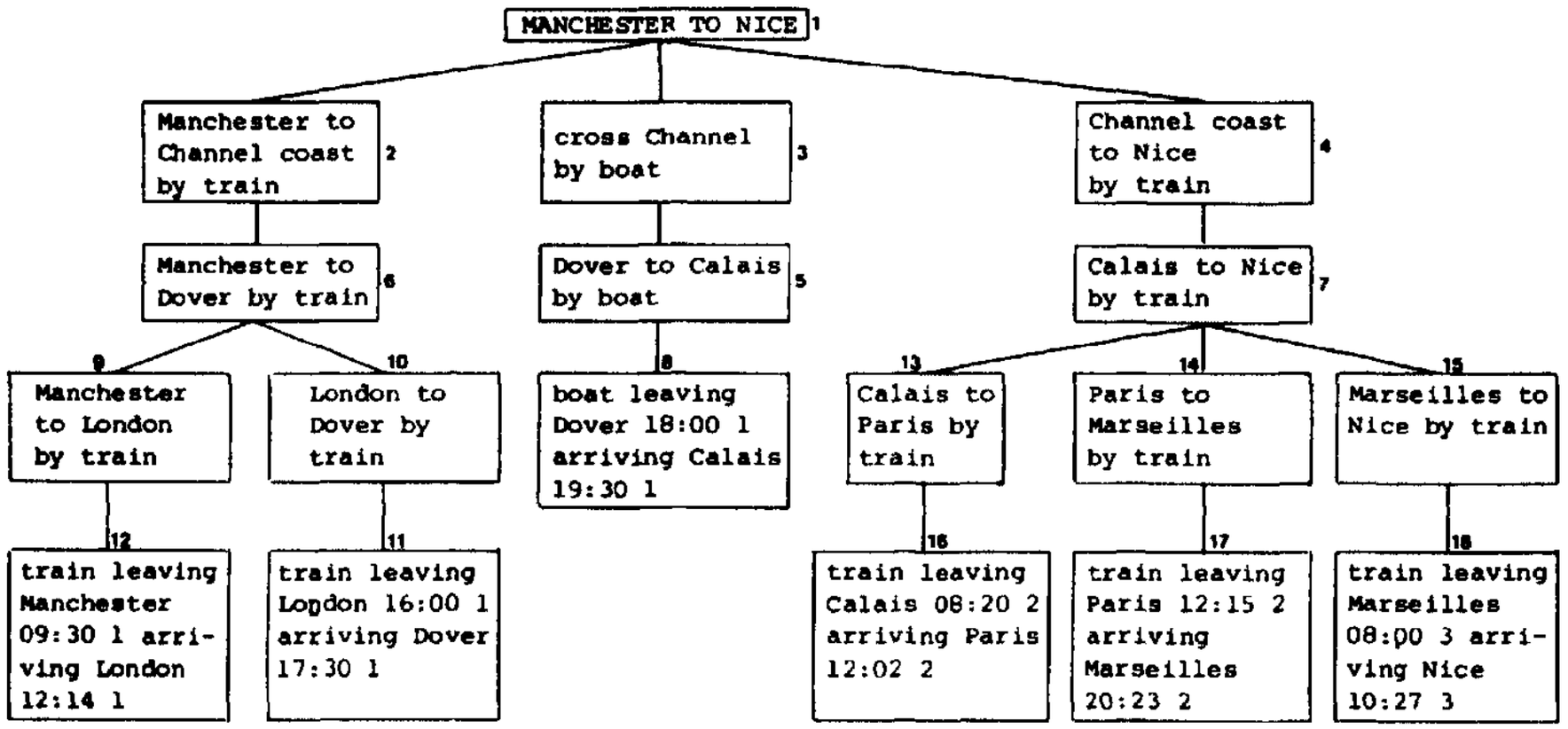

FIGURE 2 (a) Subgoal tree of plan for journey from Manchester to Nice

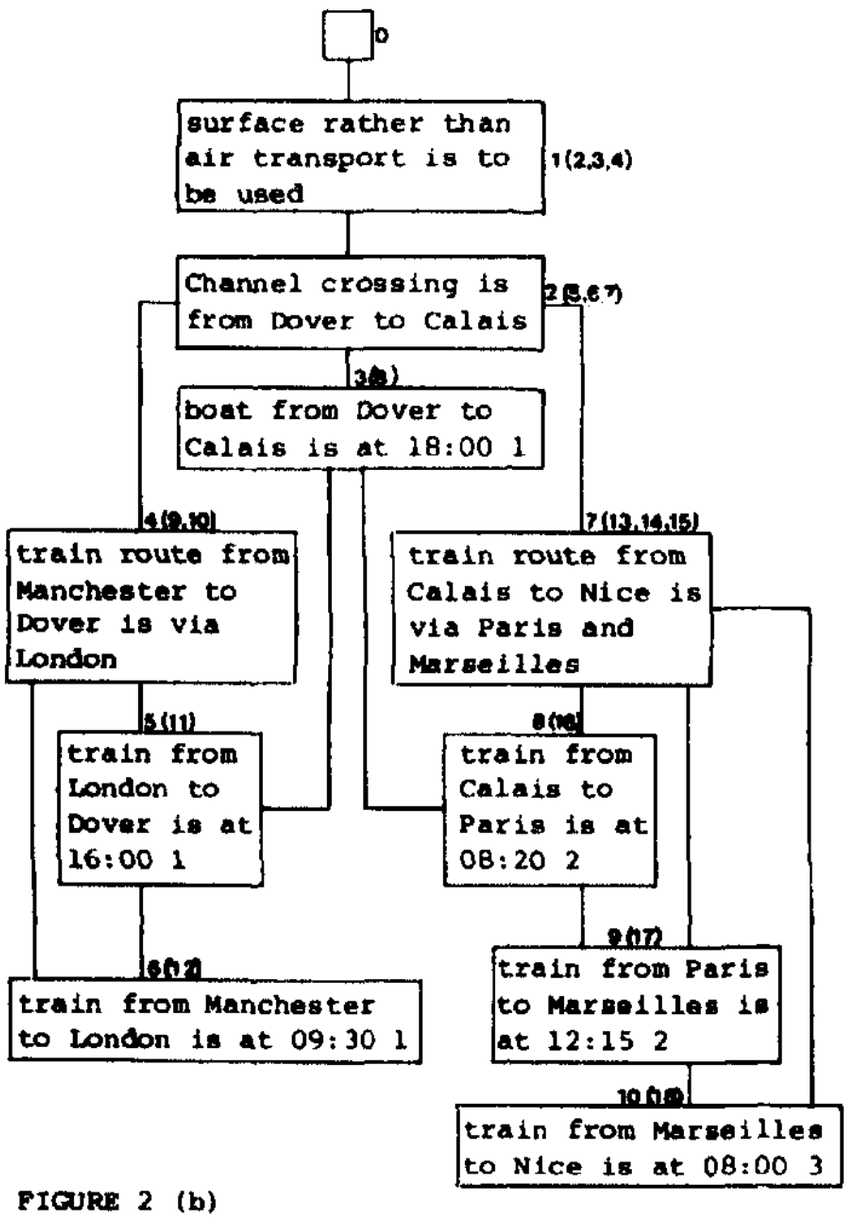

FIGURE 2 (b)

Decision Graph of plan for journey from Manchester to Nice

Since dfunctions take all their information from the plan representation existing at the time they are called, they make idealresumption points for use with the replarming mechanism described in the previous section. Indeed, this is the reason they were so designed.

The entire planning system runs under a simple backtracking regime. The problems with automatic backtracking are well known[6], and this feature of the system is undesirable. However, since emphasis is placed on trying to make good decisions at the first attempt, the bad points of backtracking do not have as much impact as they might. See reference 8 for a discussion of how the representation can be used to ameliorate backtracking for those plan construction processes using it. The unimplemented scheme proposed there is based on the fact that the representation enables a decision to be remade without remaking chronologically subsequent but logically independent decisions.

Use of the representation can also help the plan construction process in handling the global constraints of time and cost that can be imposed on a journey. The system deals with these constraints by associating with each jnode an estimate of the time and cost of the journey it represents. The more specified the journey, naturally the more accurate the estimate. At any time the most up-to-date estimate for the whole journey can be obtained by summing the individual estimates of the jnodes on the current frontier of the tree, making allowance for waiting times and overnight stops. If this estimate is much outside the prescribed limit, then the development of the plan along the current lines can be discontinued immediately, thus wasting as little effort as possible. Such an approach leads to little time being spent on journeys for which the allowance is grossly inadequate, but more on journeys for which the allowance falls just short.

The computer output produced by a simulated execution of the example plan described above is:

```
          AT MANCHESTER WAITING FOR A TRAIN TO
          LONDON
09:30 1   LEAVING MANCHESTER ON WAY TO LONDON BY
          TRAIN
12:14 1   ARRIVING IN LONDON WAITING FOR A TRAIN
          TO DOVER
14:00 1   TRAIN FROM MARSEILLES TO NICE AT 08:00
          CANCELLED

                  **REPLANNING**

          AT LONDON WAITING FOR A TRAIN TO DOVER
16:00 1   LEAVING LONDON ON WAY TO DOVER BY TRAIN
16:30 1   BOAT ACROSS THE CHANNEL FROM DOVER TO
          CALAIS AT 18:00 CANCELLED
17:30 1   ARRIVING IN DOVER

                  **REPLANNING**

          AT DOVER WAITING FOR A BOAT TO CALAIS
22:00 1   LEAVING DOVER ON WAY TO CALAIS BY BOAT
23:30 1   ARRIVING IN CALAIS STAYING IN CALAIS
          FOR THE NIGHT
23:45 1   STORMS IN THE CHANNEL
08:20 2   LEAVING CALAIS ON WAY TO PARIS BY TRAIN
10:00 2   ALL TRAINS BETWEEN PARIS AND MARSEILLES
          CANCELLED
12:02 2   ARRIVING IN PARIS

                  **REPLANNING**

          AT PARIS WAITING FOR A TRAIN TO NIMES
09:03 3   LEAVING PARIS ON WAY TO NIMES BY TRAIN
20:25 3   ARRIVING IN NIMES WAITING FOR A TRAIN
          TO MARSEILLES
21:53 3   LEAVING NIMES ON WAY TO MARSEILLES BY
          TRAIN
23:15 3   ARRIVING IN MARSEILLES STAYING IN MAR-
          SEILLES FOR THE NIGHT
13:00 4   LEAVING MARSEILLES ON WAY TO NICE BY TRAIN
15:20 4   ARRIVING IN NICE JOURNEY FINISHED
```

During this execution, four unexpected events occur and replanning is necessary for three of them. Following the execution in detail will make clear the workings of the replanning mechanism (step numbers refer to the description in section 2).

Execution proceeds normally until 14:00 1 when the train from Marseilles to Nice at 08:00 is withdrawn. This means that the final step in the plan, jnode 18, is unexecutable and so replanning commences. The successfully executed part of the plan i.e. the journey from Manchester to London (jnodes 9, 12 and dnode 6) is discarded (step 1). The starting points of jnodes 1, 2, 6 are changed to London and the time and cost limits for the journey decremented by the amount already used (step 2). The decision, dnode 10, responsible for the unexecutable step in the plan is UNDONE resulting in the removal of jnode 18 and dnode 10 (step 3b). The plan construction process is then reentered through the dfunction, TRAINTIME, associated with dnode 10 (step 3c). TRAINTIME chooses another time for the Marseilles-Nice journey and a complete revised plan results. This new plan uses all the original plan (minus the Manchester-London journey) except that jnode 18 is replaced by jnode 19, a journey from Marseilles to Nice at 13:00 3, and decision 10 is replaced by decision 11, the choice of that service.

Execution of the revised plan is then resumed, but has not continued for long when the boat service that was to be used is cancelled. The decision responsible for the choice of that service is decision 3. After removing the successfully completed part of the plan, i.e. everything involved in getting as far as Dover, dnode 3 is therefore UNDONE. Its UNDOING leads also to the UNDOING of dnodes 8, 9 and 11, i.e. all the remaining timing decisions (for they were chosen to fit in with it) and consequently to the removing of all the timed jnodes (8, 16, 17, 19). Note that decision 7, the rail routing decision, though made chronologically after decision 3 is logically independent of it and therefore unaffected. The planning process is then reentered through the resumption point of decision 3, the dfunction SEATIME, and produces the retimed plan shown in figure 3.

Execution of this newly revised plan then continues with the retimed Channel crossing. The next event, even though it causes all boat services across the Channel to be cancelled, does not affect execution of the plan since, by the time it occurs, the Channel has been successfully crossed.

The final event of the journey (which refers to the direct line from Paris to Marseilles via Lyons) is handled similarly to the first two. This time very little of the original plan can be saved because the routing decision for France, dnode 7, is rendered invalid, and all of the journey that is left is the trip through France. Invocation of TRAINS, the dfunction of dnode 7, produces the plan which finally leads without further interruption from Paris to Nice.

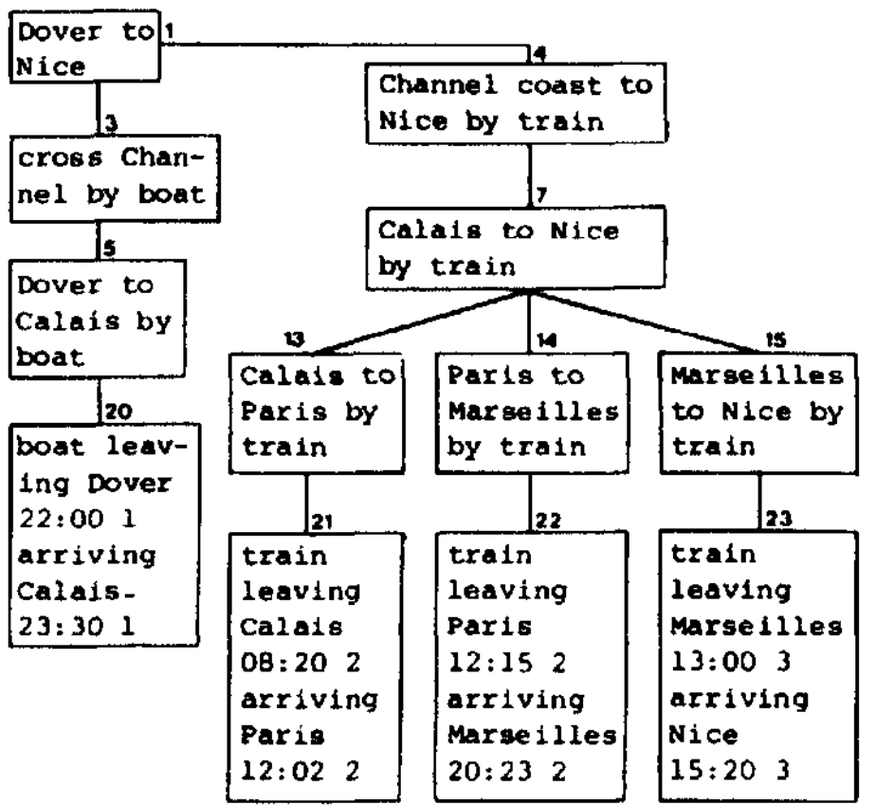

Figure 3 (a) Subgoal tree of a plan reconstructed after an execution failure

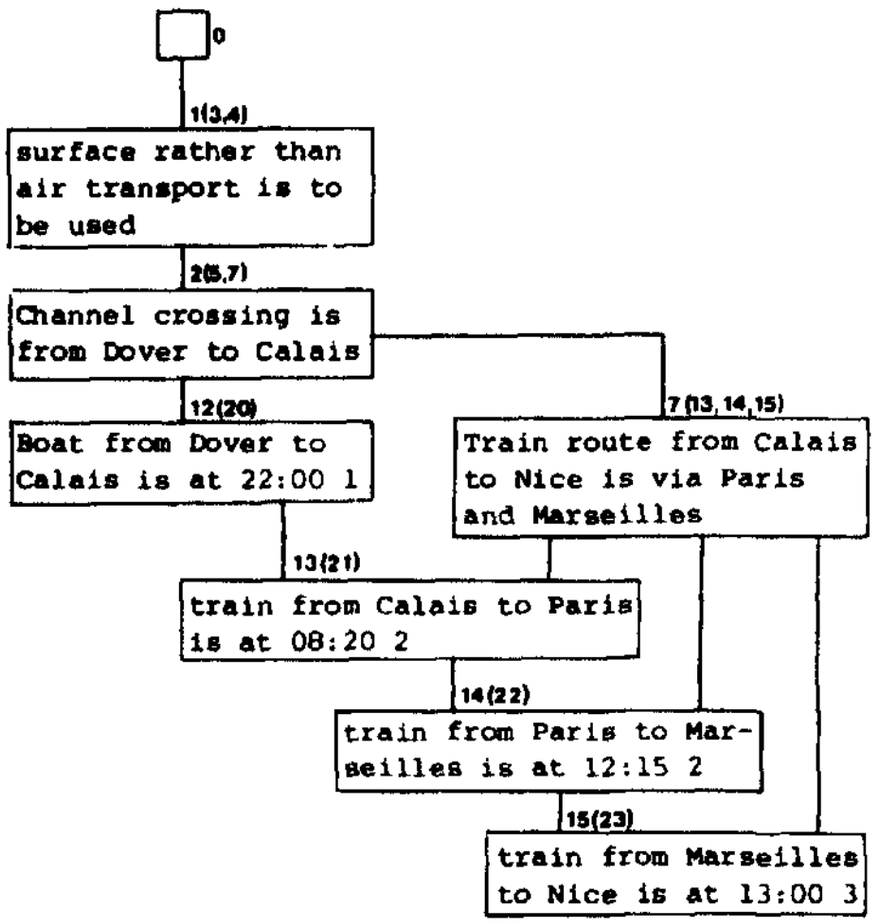

Figure 3 (b) Decision graph of a plan reconstructed after an execution failure

4. Representing Plans for More Typical Robot Worlds

The system described above is atypical of robot planning systems. Most other systems are concerned with worlds in which e.g. a mobile robot pushes blocks around through a network of rooms[1,3,5], or a one-handed robot moves blocks about on a table top . The essential difference between these worlds and the transport world lies in the number of features of the world that change or can be changed in a predictable way. For the transport world such features are few and fixed (only the robot's position and consequently the amounts of time and money available to it), while for the other worlds they are numerous and variable (e.g. the position of every block).

The systems mentioned above that deal with such domains all use some sort of data base to model the changing features of their worlds. This data base contains a record of all changeable features of the world and is typically updated in a simulation of execution as plan construction proceeds. If the representation is to make explicitly available at all times information about every step in a plan, one data base is insufficient for use with it. There must be several, one for each jnode in the plan. To provide for each jnode a separate data base containing mention of all changeable features of the world is, however, unnecessarily wasteful. It is possible and even advantageous to use one global data base plus a linked incremental system of alterations to it.

In this scheme, each jnode has attached to it an alteration, i.e. a record of the changes expected to be brought about by the action associated with it. The presence or absence of any piece of information in the data base representing the expected state of the world after the execution of some jnode can then be found by looking back along the frontier of the tree, starting with that jnode, for the first mention of that item in an alteration. An item can be mentioned as being either present or absent, but the first mention of it discovered determines its status. If the item is not mentioned in any of the alterations, its status is the same as its status in the global data base describing the initial state of the world. An example will make this clearer.

Consider the simple world shown in figure 4 with a robot, two boxes, four rooms and four connecting doors. Figure 5 shows the tree and graph of a plan to transform it from the configuration shown in figure 4 to one in which the two boxes are next to each other. *The* primitive actions shown in the plan are similar to those used by STRIPS[1] for a similar world. No assumptions are made, however, about the type of process which produced the plan or about the order in which its several steps were elaborated.

The global data base representing the initial state of the world might contain the following items:

(IN ROBOT R0OMI) (IN BOXI R00M2) (IN B0X2 ROOM3)

(CLOSED DO0R1) (OPEN D00R2) (OPEN DOOR3)
(OPEN D00R4)

plus other information describing static features of the world such as D00R1 connects ROOMI and R00M2 or that the two boxes are pushable.

The alterations associated with each jnode might be as follows:

```
1.   (NEXTO BOX1 BOX2)
2.   (IN ROBOT ROOM2) (NEXTO ROBOT BOX1)
     -(IN ROBOT $) -(NEXTO ROBOT $)
3.   as 2, but without (NEXTO ROBOT BOX1)
4.   (NEXTO ROBOT DOOR1) -(NEXTO ROBOT $)
5.& 7.  same as 3.
6.   -(CLOSED DOOR1) (OPEN DOOR1)
8.   (NEXTO ROBOT BOX1) -(NEXTO ROBOT $)
9.   (IN ROBOT ROOM3) (IN BOX1 ROOM3)
                      (NEXTO BOX1 BOX2)
     -(IN ROBOT $) -(IN BOX1 $) plus A
10. as 9, without (NEXTO BOX1 BOX2)
11. (NEXTO BOX2 DOOR2) plus A
12.& 13.  same as 10.
14. (NEXTO BOX1 BOX2) plus A

where A is (NEXTO ROBOT BOX1) -(NEXTO ROBOT $)
           -(NEXTO BOX1 $) -(NEXTO $ BOX1)
      - denotes the absence of that item
      $ is a free variable that stands for anything
```

Thus -(IN ROBOT $) denotes the absence of all items of that form, i.e. it cancels any specification of which room the robot is in. The positive items of an alteration take precedence over its negative items when conflicts arise.

In this complete plan, the room the robot is expected to be in after the execution of jnode 13 is found directly from the alteration of jnode 13,

since it contains (IN ROBOT ROOM3). For jnode 11, however, it is necessary to follow the frontier back until jnode 7, to find this information, since the alterations of neither jnode 11 nor jnode 8 contain an item of the form (IN ROBOT room). Similarly the expected position of B0X2 after the execution of every jnode will always be found from the global data base since no mention of it is made in any alteration either positively or negatively.

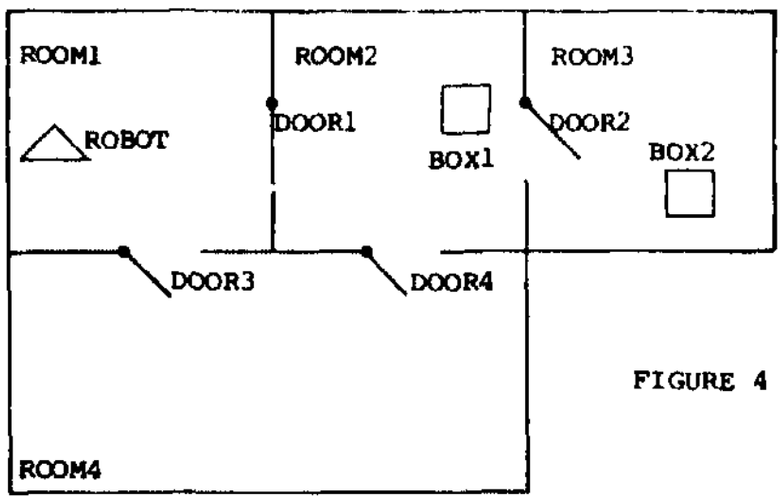

FIGURE 4

Initial world in more typical robot planning domain

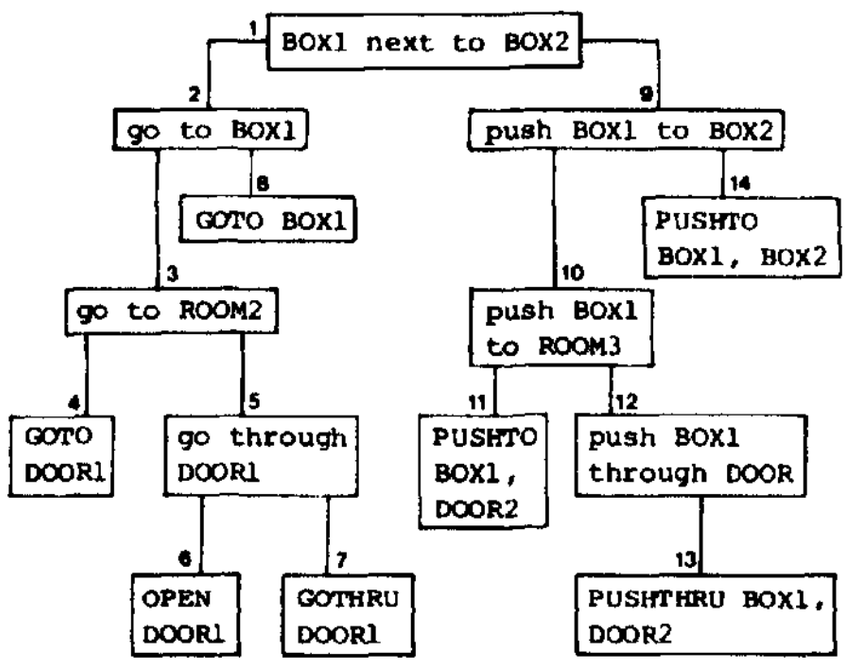

FIGURE 5 (a) Subgoal tree of plan for pushing the two boxes in figure 4 together

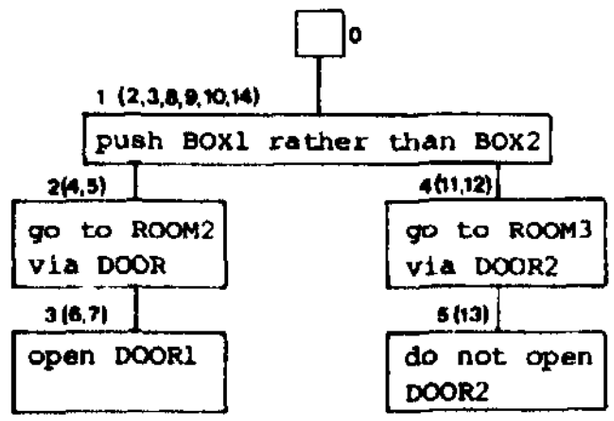

FIGURE 5 (b) Decision graph of plan for pushing the two boxes in figure 4 together

Using such a scheme with predicates which give essentially the same information at differing levels of detail can lead to difficulties when different steps of a plan are at different levels of detail. Suppose, for example, that jnode 8 was further developed to specify (AT ROBOT X), X being the position the robot must be in to push B0X1 in the proper direction. Then the data bases of all subsequent jnodes in the execution sequence would contain this (for them) erroneous information. One solution is to ensure that whenever a new peice of information is established, all other information both at the same and lower levels of detail that could conflict should be erased. Thus including (IN ROBOT room) in an alteration would also entail the inclusion of -(IN ROBOT *$)*, -(NEXTO ROBOT $) and -(AT ROBOT $) in the same alteration.

An important advantage of the incremental scheme is related to this problem. Any details filled in about one step of a plan become immediately available in the data bases of all subsequent steps of the plan, providing, of course, no intervening alteration contradicts the detail. Thus the data base of one jnode can be updated by development of a completely different step of the plan. For example, in the above plan, jnode 14 is already in existence immediately after decision 1 is made. (CLOSED DOOR1) is present in its data base, since the item is present in the original data base and no intervening alteration existing at that time mentions it. However, after decision 3 has been made, jnode 6 is present on the frontier of the tree and its alteration removes (CLOSED DOORI) from the data base of jnode 14 and inserts (OPEN DOORI) in its place.

This data base scheme makes the construction of the decision graph straightforward. A decision directly depends exactly on those other decisions which first established the information influencing its formation. Thus if information from the alteration of a jnode influences a decision, that decision depends on the decision responsible for the creation of the eldest ancestor of that jnode whose alteration contains the same information.

The advantages of having information about the entire plan constantly available are more apparent in a world where different steps of the same plan are more likely to interfere with each other. An example of such a world is the well known BLOCKS world of Winograd[7]. The robot in this world cannot lift up a block which has other blocks on top of it. To manipulate such a block, it must first clear off its top by putting the supported blocks somewhere else, usually on the table. Such tactics can lead to problems when one block occupies the space needed for another. Such problems are resolved rather inefficiently by backtracking in Winograd's system. If the above representation were used, a plan could be developed as far as possible down each branch of its tree until it was necessary to choose definite places on the table *in* which to put the blocks. A place for each block could then be chosen using the information in the tree about all the other objects which had to be fitted in, thus eliminating the need for trial and error backtracking search.

The method of plan reconstruction after execution failure, baaed on use of the representation and described in section 2, works in exactly the same manner as illustrated in section 3. For example, suppose in the execution of the present plan, the robot discovered when it got to DO0R1 that it was not only CLOSED but LOCKED. It would apply the replanning procedure to jnode 5, as the most senior unexecutable ancestor of jnode 6, the unexecutable primitive action of opening D00R1. Firstly, jnode 4 would be discarded since it had already been successfully executed. Then the global data base would be updated with the current position of the robot and the fact that D00R1 is LOCKED. Dnode 2, the choice of the robot's route from ROOH1 to R0OM2, is the decision directly responsible for jnode 5, so it would be UNDONE, removing jnodes S, 6 and 7 from the tree and dnodes 2 and 3 from the graph. Since the knowledge that D0OR1 is locked is available, reentering the planning process should then result in the route from ROOM1 to R00M2 via R00M4 being chosen. In the new plan, thus produced, the tree representing the robot's going from R00M1 to ROOM2 would grow from the original jnode 3. All the rest of the plan about pushing B0X1 through DOOR2 would, of course, remain intact throughout this process.

5. Conclusion

A representation for robot plans has been presented which can assist in both the construction and execution of such plans. Use of the representation was illustrated by a description of a working planning and execution system concerned with the journeyings of a robot traveller in a network of public transport services. The problem domain of this system is atypical of robot planning systems in general, but it was also shown how the representation could be used for more typical worlds. The examples of the use of the representation indicated a number of features of it:

When a failure occurs during execution, the representation enables the discarding of those parts of the plan invalidated by the failure and thus reconstruction of the plan based on the selective reuse of the analysis made by the original planning process.

The structure of the representation is oriented towards planning systems using hierarchical levels of detail, and enables details to be filled in in an order unrelated to execution order. It also facilitates the use of information about one part of a plan in constructing other parts of the same plan.

The representation makes these contributions to plan construction and execution, because, for a particular plan, it reflects the structure of the process which produced the plan. The present representation is able to reflect the structures of currently typical planning processes. For more advanced processes with more structure a representation capable of reflecting the extra structure would probably give further benefits similar to the ones mentioned above.

Acknowledgements

This paper is based on a University of Edinburgh Master of Philosophy thesis[6]. As in that, I would like to thank my supervisors Harry Barrow and Donald Michie for their guidance and encouragement. My financial support at Edinburgh was provided by the Science Research Council.

References

1. Fikes, R. E. and Nilsson, N. J. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", Artificial Intelligence, vol. 2, 1971, pp. 189-208.

2. Fikes, R. E., Hart, P. E. and Nilsson, N. J., "Learning and Executing Generalized Robot Plans" Artificial Intelligence, vol. 3, 1972, pp. 251-288.

3. Sacerdoti, E., "Planning in a Hierarchy of Abstraction Spaces", Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, California, 1973., pp. 412-422.

4. Nilsson, N. J., "A Hierarchical Robot Planning and Execution System", AIC Technical Note 76, SRI Project 1187, Stanford Research Institute, California, 1973.

5. Siklossy, L. and Dreussi, J. "An efficient Robot Planner which Generates its own Procedures", Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, California, 1973, pp. 423-430.

6. Sussman, G. J. and McDermott, D. V., "Why Conniving is Better than Planning", AI Memo 255A, Massachusetts Institute of Technology, Cambridge, Mass. 1972.

7. Winograd, T., "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language", Project MAC TR-84, Massachusetts Institute of Technology, Cambridge, Mass., 1971.

8. Hayes, P. J., "Structuring of Robot Plans by Successive Refinement and Decision Dependency", M. Phil. Thesis, School of Artificial Intelligence, University of Edinburgh, Edinburgh, 1973.