Associating A.I. Planner Entities with an underlying Time Point Network

Brian Drabble and Richard Kirby
Artificial Intelligence Applications Institute
University of Edinburgh
Edinburgh EH1 1HN

November 15, 1990

Abstract

This article outlines the design rationale behind the entity and time representation and reasoning system incorporated into the O-PLAN2 planner. The paper shows the advantages of splitting the planning entities (such as activities, events and time orderings) from the underlying temporal information of the domain. This allows a single architecture to be tailored to act as planner, scheduler or controller with only minor changes in the knowledge required. The paper concludes with a description of the current O-PLAN2 system together with our early results and future research direction.

1 Background to the Research

The earlier O-PLAN project at Edinburgh [2], 1984-1988, focussed on the techniques and technologies necessary to support the informed search processes needed to generate predictive plans for subsequent execution by some agent. The current O-PLAN2 [1] project continues the emphasis placed on the formal design of a Planning Architecture in identifying the modular functionality, the roles of these modules, and their software interfaces.

O-PLAN2 is incorporated within a blackboard-like framework; for efficiency reasons we have chosen an agenda driven architecture. Items on the agendas represent

[1]This research is supported by the US Air Force/European Office of Aerospace Research and Development by grant number EOARD/88-0044 monitored by Dr Nort Fowler at the Rome Air Development Centre

outstanding tasks to be performed during the planning process, and they relate directly to the set of *flaws* identified as existing within any non-final state of the emerging plan. A simple example of a *flaw* is that of a condition awaiting satisfaction, or an action requiring refinement to a lower level. An agenda controller will choose on each planning cycle which flaw to operate on next.

2 Architecture

A generalised picture of the architecture is shown in figure 1. The main components are:

The Domain Information. Domain descriptions will be supplied to O-PLAN2 in a structured language, which will be compiled into the internal data static structures to be used during planning. The description will include details of actions which can be performed in the domain, goals to describe the planning requirements, and incremental updates or re-specifications of knowledge sources. The structured language (we call it Task Formalism) is the means through which a domain writer or domain expert can supply the domain specific information to the O-PLAN2 system, which itself is a domain *independent* planner. O-PLAN2 will embody many search space pruning mechanisms (strong search methods) and will fall back on other weak methods, if these fail, in order to preserve completeness of the search space. The task formalism is the mechanism that enables the user of the system to supply his domain dependent knowledge to assist the system in its search. This information is not updated by the operation of the system, so the information flow is depicted outwards from this static data block to show the support offered to the functional modules.

The Plan State. In contrast to the static information outlined above, the plan state (on the left of the figure) is the dynamic data structures used during planning and houses the emerging plan. There are many components to this structure, the principal ones being:

- the plan itself. This is based on a partial order of activities, as originally suggested by the NOAH planner. In O-PLAN2 the plan information is concentrated in the "Associated Data Structure" (ADS). The ADS is a list of node and link structures noting temporal and resource information (indirectly), plan information and a planning history.

Figure 1

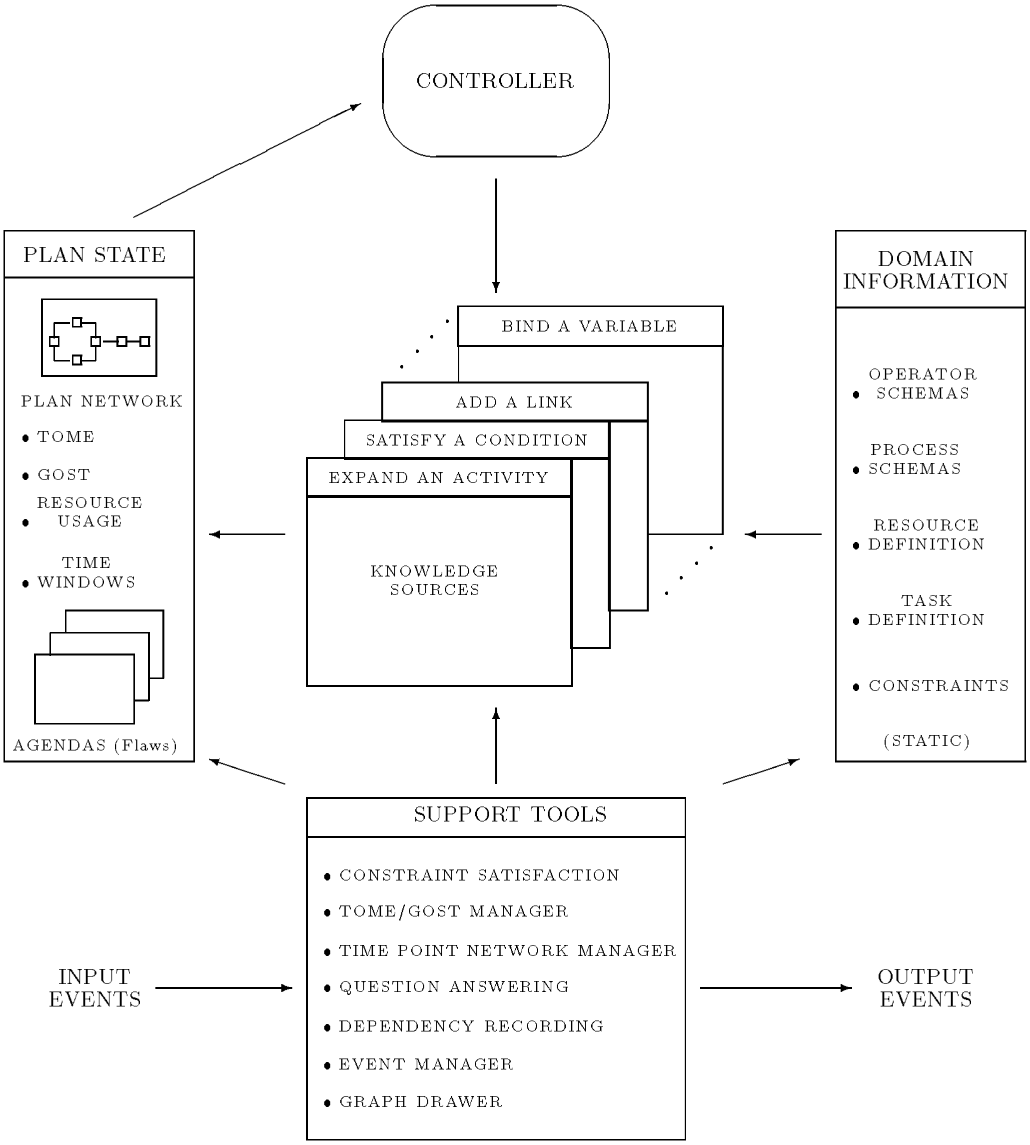


Figure 1: Figure 1: O-Plan2 Architecture

- the plan rationale. Borrowing from Nonlin and O-PLAN, the system keeps explicit information to "explain" why the plan is built the way it is. This rationale is called the Goal Structure and, along with the Table of Multiple Effects (GOST and TOME), provides an efficient data structure for the underlying condition achievement procedure used in O-PLAN2 (*c.f.* Chapman's Modal Truth Criteria [1]).

- the agenda list(s). O-PLAN2 will start with a complete plan, but one which is "flawed", hence preventing the plan from being capable of execution. The nature of the flaws present will be varied, from actions which are at a higher level than that which the executing agent can operate, to linkages necessary in the plan to resolve conflict.

The plan state is a self-contained snapshot of the state of the planning system at a particular point in time in the plan generation process. It contains all the state of the system hence the generation process can be suspended and this single structure rolled back at a later point in time to allow resumption of the search[2].

The Knowledge Sources. These are the processing units associated with the processing of the flaws contained in the plan and they embody the planning knowledge of the system. There are as many knowledge sources (KSs) are there are flaw types, including the interface to the user wishing to exert his influence on the generation process. The KSs draw on information from the static data (*e.g.* the use of an action schema for purposes of expansion) to process a single flaw, and in turn they can add structure to any part of the plan state (*e.g.* adding structure to the plan, inserting new effects or further populating the agenda(s) with flaws).

2.1 Support Modules.

In order to efficiently support the main planning functionality in O-PLAN2 there is a layer of support modules separated out from the core of the planner. These modules have carefully designed functional interfaces and declared function in order that we can both build the planner in a piecewise fashion, and in particular that we can experiment with and easily integrate new planning ideas. The following sections defines in detail the motivation and structure of the main support modules.

[2]Actually this assumes that the task formalism and the knowledge sources used on re-start are the same "static" information used previously.

2.1.1 Time Point Network (TPN)

The need for a separate time network system was identified during the earlier research programme for O-PLAN. O-PLAN's ability to represent the delays between actions and to interact with events occurring in the real world relied on data structures which were considered inefficient and cumbersome. O-PLAN could interact with fixed events such as the delivery of a resource but could not handle dynamic events occurring in the target domain. The delays between actions were handled by incorporating delays for each arch leaving a node which made the representation inflexible and inefficient.

A separate time representation scheme will allow a speed up in operations such as Question Answering (QA) and Temporal Coherence (TC) and will allow plan structures based on different temporal constructs, *i.e.* points, intervals and events. The time representation scheme needs to represent a variety of information which includes:

- The ability to represent actions and events from some given absolute beginning of time. This can be relative to the time of planning, execution or some known external events.
- The temporal constraints which limit the "position" in time at which an action or event can occur and will be represented as a upper and lower value pair. The action of events constraint will be specified relative to absolute beginning of time or relative to some other action or event.
- The dependency information which create a "belief" for the continued existence of an action or event within the plan.
- The creation of a time-line which contains those actions and events which have known start time (*i.e.* the lower and upper values are the same). This will allow easy comparison of actions and events as well as aid in the analysis of resources used during a plan execution.

2.1.2 Time Network data structures

The choice of which temporal construct to use as the basic building block was a very important consideration to be made. It was decided to use a *point based* construct for several reasons, these being:

- A time point representation directly supports standard Operational Research/PERT type algorithms.

- If a knowledge source requires to reason about the plan state in terms of intervals these can be recovered from a point based network However, points cannot be recovered from an interval based representation.
- Constraint satisfaction algorithms already exist which can operate efficiently on a point based network.

The data structures which are used to implement the temporal point network (TPN) will now be described.

2.1.3 Time point network

The time point network consists of a set of points each of which has an upper and lower bound on its temporal distance from the start of time. The format of each point is as follows:

```
(pt<n> lower_bound upper_bound)
```

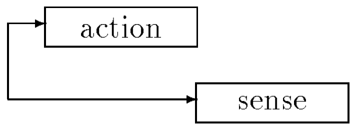
The points will be associated with actions, links and events, but this association will only be made at the Associated Data Structure (ADS) level. The point number is used as the index giving a constant retrieval time for any number of points. This structure will allow points to be retrieved, deleted and compared easily with a minimum of overhead. Points in this network which have an upper and lower bound the same will be marked with a tag so that they can easily be retrieved from the landmark line (Section 2.1.6)

In later versions of the TPN it my be possible to contextually divide the points of the TPN according to their *type* or *use*. The resource reasoning modules in QA for example may require only those points involving resources creation, deletion or usage. This may also apply to the world model and execution reasoning modules within O-PLAN2. This would be an extra level of context information between the Associated Data Structure and the information held in the "clouds" of effect format.

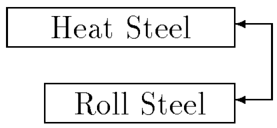
2.1.4 Managerial constraints

The *managerial* constraints are used by the TPN to handle explicit constraints between points within the network. For example the begin point and end points of an action must be constrained to be equal to the lower and upper values of its duration. The managerial constraints are also used to handle user specified constraints such as:

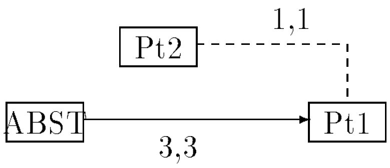
- The start time of one action being a prescribed distance before the start of another. For example, if our plan contains a sensing action then we would like the sensing action to begin a certain time after the action to be sensed has begun. This allows us to check that the action is progressing and that the required effects are now in present in the real world. The term "Action Progression" is used to describe this situation.



- The finish time of one action being a prescribed distance before or after the finish time of another. An example of this comes from a steel rolling plant domain. Here the user does not know the time taken to roll the steel or heat the steel but the user does require the end of the heating phase to be as close as possible to the end of the previous rolling phase. The term "Action Continuity" is used to describe this sort of situation.



A problem which may arise when using managerial links is that of loops occurring with the TPN. To avoid this, checking will be performed when a new managerial constraint is added. A managerial link will be maintained by a dependency check that the temporal distance between two points is maintained, *i.e.* it lies between the links upper and lower values. The managerial links will add greater flexibility to the representation by allowing points with no explicit positional constraints to be placed in time. In the following example



We can calculate on demand the position constraint for `Pt2` by knowing the position constraint for `Pt1` and the managerial constraint between `Pt1` and `Pt2`. The adding of new positional constraints will only be done as the result of a request to the TPN and *not* in order to create a closure of all relations.

2.1.5 Position constraints

The positional constraints are used to fix a point in time by specifying an upper and lower bound on its temporal "distance" from some absolute time value ABST. The value of ABST can be either time of plan generation, plan execution or some external event the plan should create or avoid. If a constraint between two points is not specified then a default of $(0, \infty)$ is assumed. The constraints are held in a table format with the index mechanism provided by the point number. The format of the table entries is as follows;

$$< Point-number >< List-of-constraints >< Current-glb-and-lub >$$

The GLB specifies the greatest upper bound and the LUB specifies the least upper bound. This gives us the *intersection* of all constraints specified for a point. The list of constraints is further divided into those which are specified as positional constraints and those which are managerial constraints. Any positional constraints which are introduced as a result of using managerial constraint information will be specially marked so that if at a later date a positional constraint is added, then only the necessary conflicts will be identified. The access times for this table is linear and tests have been carried out on tables exceeding 500 points to check on the maximum retrieval time.

2.1.6 Landmark line

The landmark line contains those points which have an upper and lower value with the same value *i.e.* co-incident. The points are sorted into ascending order from left right. The landmark line allows:

- Points to be easily compared for relations such as before, after, etc.
- Resources which have known usage times can be profiled thus aiding the resources reasoning required within the planner.

When a point is inserted into the landmark line it is *not* kept as a separate entry elsewhere within TPN. The landmark line will carry a dependency check so that if at a later date the upper and lower values of the point differ the point can be removed from the landmark line and reinstated as a point in the TPN. For example, if I change my mind about going shopping from 9.00am to sometime between 9.00am and 9.30am the corresponding end point will have to be removed from the landmark line. Each point in the land mark line is unique in that no two points can have exactly the same upper and lower values and be equal. If such a condition occurs then the points are unified and a message is sent to the ADS to inform it of the change which has been made. The use of the landmark line for resource management will have to be carefully investigated as only those resources requirements with exactly known times of occurrence. *i.e.* GLB = LUB will be entered there. Times which do not have this relation will have to be reasoned about in a similar way to that used in the EXCALIBUR system [3]

2.1.7 Time Network Manager (TNM)

The TMM is the collection of routines which interface the TPN to the ADS. These routines query the network and maintain its integrity in response to queries from the ADS. The main routines of the TNM consists of;

- **Query Routines** Checks whether a given action or constraint can be asserted in the TPN. Any failures are reported back to calling modules for its consideration
- Add or delete a point from the TPN
- Add or delete a constraint from the TPN
- **Constraint propagation and checking** Checks that a new constraint does not cause a violation within the current TPN.

2.2 Associated Data Structure (ADS)

The ADS provides the *contextual* information used to attach meaning to the contents of the TPN, and the data defining the emerging plan. The main elements of the plan are nodes and links, with ordering information as necessary to define the partial order relationships between these elements. The links between the actions and events of the plan are themselves held as nodes. This allows us to represent delays between actions cleanly and means that the reasoning mechanism does not have to decide between links and actions/events. Higher level support modules (such as QA, TOME and GOST Management (TGM), *etc.*) will rely heavily on the detail held in the ADS and on the functionality provided by the TPN. The ADS consists of information held in *two* different parts of the planner, namely the plan network (held within the Plan Manager (PM)) and the TOME and GOST entries (held within the TOME and GOST manager TGM). Figure 2 describes the relationships and data flow paths between the ADS and the TPN. The plan manager holds the graph of action, events and dummies which are in the “current plan”, as well as the symbolic time constraints. These symbolic constraints are computed as metric time positional constraints or managerial constraints within the TPN.

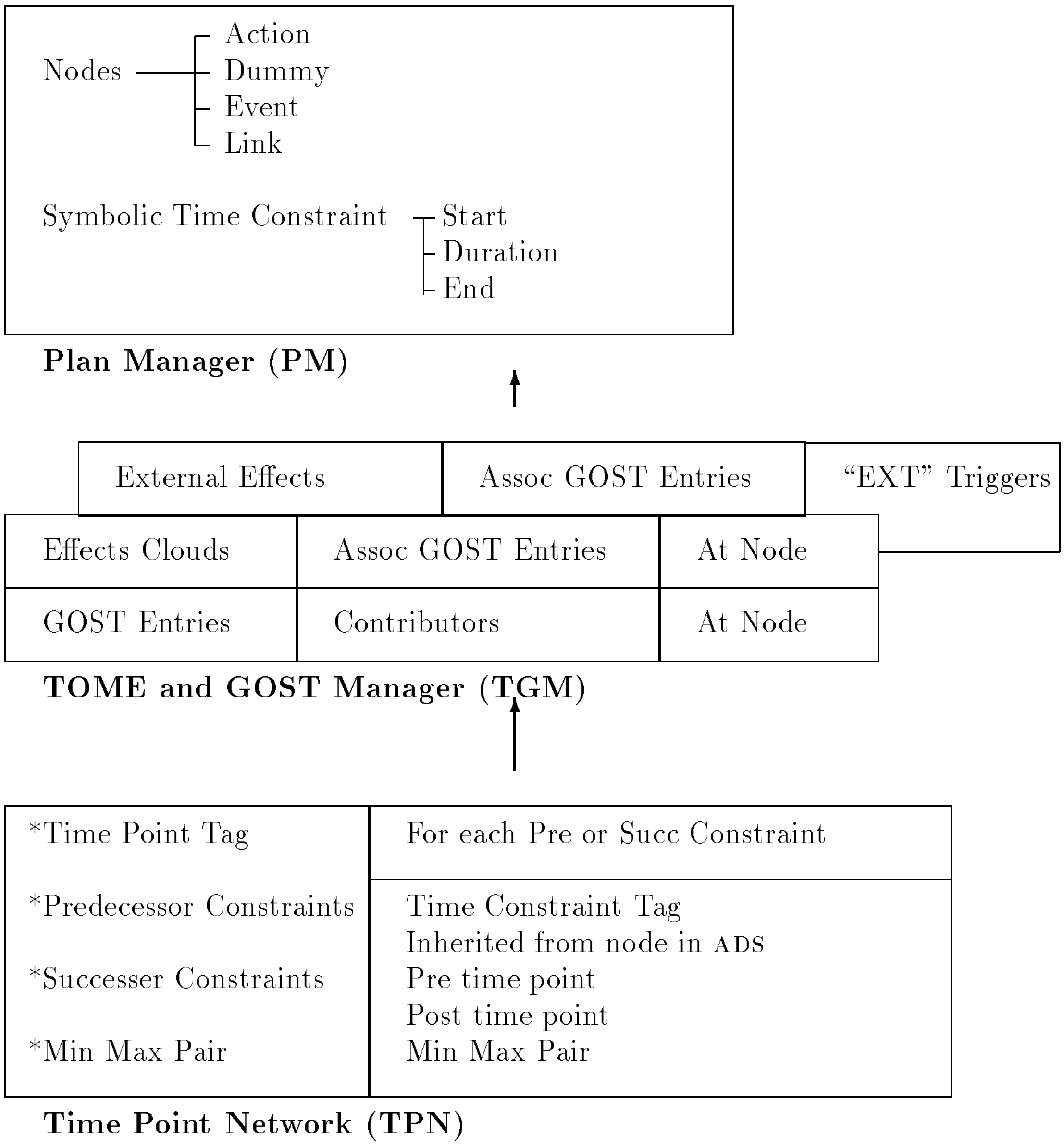


Figure 2.

The TOME and GOST Manager handles both plan effects required within the plan and external effects occurring in the real world which are also required within the plan. The entries of the TOME and GOST are partitioned into those requiring external events and those satisfied from within the plan. This is because the external events will in most cases have *triggering* conditions which must be met for the event to occur. For example, the bank opening at 9.30am requires no triggers where as a kettle boiling will require external and internal (to the plan) triggers such as the kettle being plugged in and the plug being switched on. External information will be provided and maintained by the World Model Manager (WMM) from a qualitative model built up as the plan is being generated. As O-PLAN2 develops for use in continuous command and control applications the need to predict and recover from situations becomes much more demanding. The earlier work of Drabble [3] will provide a focus for how this will be achieved. The idea behind this form of control is that shallow reasoning such as TC [4] will need to be increased with deeper models in more complex domains. The different ways we envision using qualitative reasoning are as follows:

1. If an aircraft has problems with its hydraulics then it would be a good idea to place the flaps down and lock the undercarriage before carrying out any further planning.
2. The qualitative model may also be used for generating extra constraints between actions of the plan due to newly discovered interactions. For example in making steel try to have the finish of the heating process as close to the rolling as possible.
3. During execution of a plan the qualitative reasoning can be used to check for
   (a) the unexpected side effects of actions
   (b) undesirable changes/events occurring within the domain
   (c) the successful outcome of a set of plan actions whose collective effects are designed to bring about a change in the domain.

3 Summary

The O-PLAN project took an architectural approach to the problem of representing decision making within a planning system. O-PLAN2 goes further in that it attempts to split the representation components required in the decision making processes identified. The advantages of this approach have been outlined in the paper. The first version of the TPN and ADS has been implemented and tested. These tests involved creating networks of points in excess of 500 points with a constraint set in access of 200. This has provided us with timing information for plan networks of the order of 200 nodes which is the maximum size of problem we wish to tackle with early versions of O-PLAN2. Further research is continuing on the partitioning of the network and in the dependency/triggering required for the knowledge sources and World Model Manager.

References

[1] Chapman, D. Planning for conjunctive goals. *Artificial Intelligence Vol. 32, pp. 333-377, 1987.*

[2] Currie, K. & Tate, A. O-Plan: the Open Planning Architecture. *To appear in the AI Journal. Also* AIAI-TR-*67. 1989.*

[3] Drabble, B. Planning and reasoning with processes. *Procs. of the 8th Workshop of the Alvey Planning* SIG, *The Institute of Electrical Engineers, November, 1988.*

[4] Drummond, M. & Currie, K. Exploiting temporal coherence in nonlinear plan construction. *Procs. of* IJCAI-*89, Detroit.*

[5] Hayes-Roth, B. & Hayes-Roth, F. A cognitive model of planning. *Cognitive Science, pp 275 to 310, 1979.*

[6] McDermott, D.V. A Temporal Logic for Reasoning about Processes and Plans In *Cognitive Science, 6, pp 101-155, 1978.*

[7] Sacerdoti, E. A structure for plans and behaviours. *Artificial Intelligence series, publ. North Holland, 1977.*

[8] Stefik, M. Planning with constraints. In *Artificial Intelligence, Vol. 16, pp. 111-140. 1981.*

[9] Tate, A. Generating project networks. *In procs.* IJCAI-77*, 1977.*

[10] Vere, S. Planning in time: windows and durations for activities and goals. IEEE *Transactions on Pattern Analysis and Machine Intelligence Vol. 5, 1981.*

[11] Wilkins, D. Practical Planning. *Morgan Kaufman, 1988.*