Technical Report

O-Plan Project Evaluation Experiments and Results

Brian Drabble, Austin Tate & Jeff Dalton

This technical report enhances and replaces version 1 of the O-Plan Technical Report ARPA-RL/O-Plan/TR-23 entitled ”Applying O-Plan to the NEO Scenarios” dated 1st July 1995.

Artificial Intelligence Applications Institute
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
United Kingdom

November 10, 1995

ARPA-RL/O-Plan/TR/23 Version 2

Contents

1 Introduction

The aim of this paper is to describe the evaluation experiments conducted as part of the O-Plan project. The paper will show how each experiment can be related to the categorisation of experiment types defined in the ARPI Evaluation Handbook [4]. The O-Plan system aims to address three types of ARPI experiment: Programmatic, Demonstration and Scientific. This approach has been taken since the O-Plan project took a domain problem driven perspective to validate the approach of a specified planning architecture while allowing for the integration of new scientific ideas. The principal milestones comprised three annual demonstrations.

The O-Plan project has been targeted at a specific class of problems which include:

- project management for product introduction, systems engineering, construction, process flow for assembly, integration and verification, etc.
- planning and control of supply and distribution logistics.
- mission sequencing and control of space probes such as Voyager, ERS-1, etc.

These applications fit midway between the large scale manufacturing scheduling problems found in some industries (where there are often few inter-operation constraints) and the complex *puzzles* dealt with by very flexible logic-based tools. However, the problems of this type represent an important class of industrial relevance. From work on other projects in sectors such as aerospace, logistics, manufacturing, petroleum and business, we believe that this is a richly populated class of problems.

The structure of the paper is as follows. Section 2 describes the evaluation methods used and the experiment types carried out during the project. Section 3 describes a number of major experiments which included:

- Year 1 demonstration which showed a cut-down version of an ARPI Integrated Feasibility Demonstration IFD the IFD2 scenario running in the O-Plan system,
- Year 2 demonstration which showed how a rich model of resources could be used to improve the solutions provided by a planner
- Year 3 demonstration which showed how O-Plan could be employed in a command, planning and control scenario to deal with changes occurring on the environment and in the overall task
- linking of O-Plan with the EXPECT plan analysis tool from USC/ISI.

Section 4 describes a number of additional experiments which were used to evaluate the new functionality and capabilities being added to the system. The Appendix describes the algorithms used to implement the plan repair mechanism demonstrated in the Year 3 major demonstration.

2 Methods of Evaluation

The aim of this section is to describe the evaluation experiments conducted as part of the O-Plan project.

The O-Plan project has identified a number of demonstrations closely related to ARPI programmatic goals in a sufficiently simple domain which allows for the investigation of scientific goals. This domain is called PRECiS (Planning, Reactive Execution and Constraint Satisfaction) [24]. The PRECiS environment defines the data and hypothetical background for a demonstration domain related to logistics and transportation planning/scheduling problems and Non-combatant Evacuation Operations (NEOs). The definition of the PRECiS environment has drawn on work by:

- Brown (at MITRE) to describe a realistic NEO scenario for the Planning Initiative's IFD3
- Reece and Tate (at Edinburgh) to define an openly accessible fictional environment based on the island of Pacifica [23], suitable for enabling technology researchers interested in planning and reactive execution of plans,
- Hoffman and Burnard (at ISX) to produce a cut down demonstration scenario suitable for transportation scheduling research experiments.

Four primary needs of the ARPA/Rome Laboratory Planning and Scheduling Initiative are met by the PRECiS environment:

1. realistic scenarios can be explored from the data provided in the environment, for Course of Action (COA) generative planning, case based reasoning, transportation scheduling and the reactive execution of plans;
2. requirements of "tier-1" enabling researchers are sufficiently met by the data in order for them to pursue their individual research programmes;
3. entities in the environment are hypothetical and do not reflect actual peoples and locations. However, they are realistic in the types of data that would normally be available;
4. the scenario and domain descriptions are not confidential or military critical. They can be openly demonstrated and publications can be based upon them. This is important for enabling researchers.

Using the PRECiS domain as a base, the O-Plan project validated its vision of the component parts of the architecture for a responsive command, planning and control environment which is provided in a modular fashion. This allowed checks to be carried out on the components and pathways to ensure they were suitable and appropriate.

At the same time as validating the architecture, the O-Plan project made scientific progress in this integrated framework. This is addressed in the project's Year 2 and 3 demonstrations which addressed resource reasoning and integrated command, planning and control respectively.

The experiments carried out are covered by three main categories of the ARPI Evaluation Handbook and are as follows:

- **Programmatic**:
  The programmatic element aims to show the relevance of the O-Plan project to the goals of the ARPI and in particular its impact on the US military planning community. The impact was measured as follows:
  - Improved connectivity and consistency between command, planning, scheduling and control.
  - Open, inspectable, explainable and changeable plans.
  - Greater Scope for COA analysis and greater plan reliability.
- **Demonstration**:
  The demonstrations in Year 2 and Year 3 show O-Plan solving a series of problems from an ARPI relevant problem class. In order to help this process the project developed the PRECiS domain description with other members of the Initiative. This allows for the presentation of ideas to researchers outside the ARPI while maintaining the confidentiality of the target domain. In addition to the Year 2 and 3 demonstrations, other demonstration experiments were conducted. These including the linking of the O-Plan system with USC/ISI's EXPECT plan analysis tool [16] to allow plans generated by O-Plan to be evaluated against user provided domain dependent plan evaluation criteria. The evaluation matrix developed by EXPECT for a number of plans generated by O-Plan could allow the user to examine the quality of the solutions being generated by O-Plan on a series of problems from the PRECiS domain.
- **Scientific**:
  The scientific experiments showed how specific technical features of the O-Plan system have improved the quality of the solution presented to the user. The features examined in detail were as follows:
  - its ability to use resource reasoning in an activity planner framework,
  - its ability to explicitly represent authority in a command, planning and control environment.

The following sections describes the experiments carried out during the project and describes in outline terms the method of the experiment, the results obtained and the measures taken (where appropriate) to overcome any problems the experiment highlighted. Section 3 describes the major programmatic experiments conducted, and Section 4 describes a number of experiments conducted to evaluate the functionality and capabilities of the developing O-Plan system.

3 Experiments

The aim of this section is to describe some of the major experiments which have been carried out during the project. Each experiment will be described in outline and will be classified according to the taxonomy developed in the ARPI Evaluation Handbook [4].

3.1 Demonstration Experiment: Year 1 – 1993: Generation of Plans from the IFD-2 Scenario

One of the first year aims of the O-Plan project was to repeat the ARPI Integrated Feasibility Demonstrator Number 2 IFD-2 with O-Plan taking the place of SIPE-2. From the start of the experiment, it was recognised that SIPE-2 [32] was a more mature system than O-Plan and as such this could only be an approximation to IFD-2. However, using the Task Formalism (TF) (O-Plan's domain input language) then supported within O-Plan Version 2.1 it was possible to encode the SOCAP domain and to identify a number of shortcomings in O-Plan TF [7] [8]. The schema library for this domain contained 63 schemas which defined alternative missions, deployment and employment plans, sea and airlift resources, etc. The Courses of Action (COAs) generated contained an average of 150 actions and were developed in approximately 50 seconds. O-Plan was able to generate plans in the SOCAP domain for two tasks:

- **Task 1**: "Deter three threats"
  The task requires a plan to deter one army, one air force and one navy threat by specified dates. The threats are forces which have crossed the protected border.
- **Task 2**: "Deter three threats and counter a further nine"
  The task requires a plan to deter the same three threats as well as countering a further nine threats: three army, navy and air force respectively. These nine forces are threatening to cross the border but have not yet done so.

The experiment highlighted a number of problems in the way the system handled the satisfaction of `only_use_if` and `only_use_for_query` conditions and in the handling of plan objects (referred to as plan state variables). The experiment allowed these problems to be highlighted and fixed [31].

The experiment also allowed the testing of user defined functions for selecting the next agenda entry to run (instead of the default O-Plan mechanism). The agenda entry selection function was needed to force the planner to decide which military unit was to be used before developing the appropriate deployment plan for it. Off-line analysis showed that the problem could be solved with little or no search being involved. For example, many of the military units which could be chosen for a particular mission were similar and consequently the planner should have left the decision over which military unit to use until it was forced upon it, i.e., developing the military unit's employment plan. The agenda selection function allowed the user to provide this knowledge in a form which can be used by the system. The experiment showed that the user defined function reduced the search time from 10 hours to 50 seconds by removing many redundant paths from the search space. This analysis has proved useful in guiding work in Mixed Initiative Planning mechanisms for O-Plan [29]

3.2 Programmatic & Scientific Experiment: Year 2 – 1994: Use of a Rich Resource Model in an Activity Planner Framework

The aim of the Year 2 demonstration was to show O-Plan in a military-relevant resource-based scenario. The main aim of the demonstration was to show the benefits of using a richer resource model, e.g., the modelling of trucks, helicopters, cargo planes, passengers planes, air tankers, diesel and aviation fuel, storage tanks, runways, etc., within a generative activity planner such as O-Plan. In order to accomplish this a new Resource Utilisation Manager (RUM) [13] was designed which could deal with a number of different resource types and research was conducted into the ways in which the planner could make use of the domain information about resources to restrict its search. The resource type hierarchy defined for the RUM was consistent with that defined for KRSL [19] and extended the KRSL definitions in a number of ways. The demonstration also provided a check on the development of the functionality of the emerging O-Plan system and in particular the system's ability to reason with numbers and numerical ranges. The use of a rich model of resource management in an activity planner was one of the principal research themes of the project.

As part of the preparation for the demonstration a study was carried out into the different types of resources present in planning domains and into previous planning approaches to resource reasoning [9]. The results of this study were twofold.

1. It became possible to identify the type of resource reasoning support which should be possible with an activity planning framework.
2. It resulted in the design of a flexible Resource Utilisation Manager (RUM) for use in an activity planner such as O-Plan and SIPE-2.

The support provided by the new RUM design would allow a range of resources types to be represented and manipulated and went beyond those types supported by KRSL.

We set out to use a simpler Resource Utilisation Manager in O-Plan and existing planner features to deal with resources. The demonstration successfully showed that plans could be generated for a number of different resource constrained tasks specified in the PRECiS domain. A number of techniques were explored and validated which showed how resources could be defined and manipulated using a range of methods. These methods made explicit use of O-Plan's simple Resource Utilisation Manager to track consumable resources and O-Plan's World Condition and Effect (TOME and GOST) Manager to track reusable/sharable resources. Whilst these techniques allowed some of the coverage as was expected with the new RUM they do not have the same level of flexibility and support. In tasks where the resources were limited, e.g., small amounts of diesel fuel, the system was able to use knowledge of resources to rule out certain options as being impossible. In tasks where the choices were more extensive, e.g., use any transport type with no temporal restrictions, the system was still able to find a solution in an acceptable period of time.

The tasks described in the demonstration represent an interesting class of problems faced by the US military. They demonstrate how a series of transport assets (planes, helicopters, ground transports) and cargo (fuel) can be moved and coordinated between one location and another. The plans produced took into consideration many real world constraints on time and resources. However, in order to facilitate the demonstration a number of simplifications were introduced, e.g., fuel for the C5 and C141, crew schedules and maintenance periods were ignored. It would have been possible however, to introduce these resource and time constraints without drastically altering the performance of the system. The overall performance of the system showed that plan domains can be encoded with resource information and that plans can be generated which use this knowledge to restrict the options and choices to be considered.

While the new RUM has not been implemented in the current O-Plan prototype (Version 2.3), the research has investigated the underlying mechanisms necessary to support the various resource types and to integrate resource reasoning about each type into an activity planner. The approach adopted in the research – as stated in the original proposal – was to take an activity centred reasoning approach and to relate resource reasoning to this. The project does not claim this to be the best way to handle domains in which resource contentions dominate. Approaches such as in KIDS, [15] OPIS [26] or TOSCA [2] may be more appropriate. TOSCA is itself based on the O-Plan architecture but uses a resource centred representation and knowledge sources.

3.3 Programmatic & Scientific Experiment: Year 3: Coordinated Command, Planning and Control

This was a demonstration experiment which showed O-Plan solving a number of tasks from an integrated command, planning and control scenario. The aims of the demonstration were to show:

- O-Plan reacting to changes in the environment and identifying those parts of the plan which were now threatened by these changes.
- O-Plan reacting to changes in the overall task by integrating new plan requirements into the plan.

In both these cases the changes were to be made to an ongoing and executing plan.

The types of changes explored in this demonstration include failures of trucks due to blown engines and tyres and the inclusion of new objectives, e.g., pick up an extra group of evacuees. The PRECiS/Pacifica based example used for the demonstration has been deliberately simplified to allow a number of different aspects to be explored while keeping the plan to a manageable size. This is for viewing purposes only so that the user could follow what was happening in the demonstration. However, while being a simplification, the types of problem encountered and the solutions proposed by the planner are of relevance to military crisis action planning. Larger and more complex plans are available in other Pacifica domains.

The schema library for this domain contained 12 schemas which defined alternative evacuation methods, e.g., trucks or helicopters, fuel supplies, transport aircraft, etc. The COAs generated contained an average of 20 actions and were developed in approximately 40-60 seconds. Four different repair plans were used in the demonstration as follows:

- To repair a blown engine on a ground transport
  - The engine can only be fixed by a repair crew which is dispatched from the Pacifica airport at Delta with a tow truck. The ground transport is then towed to Delta for repairs. The evacuees remain with ground transport while it is being towed.
  - The failure of the transport occurs in a time critical situation and there is insufficient time to tow the broken transport to Delta. The evacuees are moved from the broken ground transport by helicopter to Delta and the transport is abandoned.
  - This is similar to the previous repair in that the failure occurs in a time critical situation except in this plan the evacuees are moved by another ground transport instead of by helicopter.
- To repair a blown tyre on a ground transport
  - The driver of the ground transport can fix the tyre by the side of the road. The effect of the repair action is to delay the ground transport by a fixed amount of time.

Details of the algorithms and methods used to implement the plan repair features are given in the Appendix.

Closely allied to the third year O-Plan demonstration showing the link between a proactive planner and a more comprehensive reactive execution agent, an associated Ph.D student project by Glen Reece showed a reactive execution agent [22] based on the O-Plan architecture. This has been used to reactively modify plans in response to operational demands in a simulation of the Pacifica island in the context of a NEO.

3.4 Programmatic & Demonstration Experiment: Linking of O-Plan and the EXPECT Plan Analysis Tool

This was a demonstration experiment conducted with USC/ISI in which the O-Plan system was linked with their EXPECT plan analysis tool [11],[12]. The ARPI Integrated Feasibility Demonstration Number 2 (IFD2) was used for IFD-2 was chosen for the evaluation domain. The schema library for this domain contained 63 schemas which defined alternative missions, deployment and employment plans, sea and airlift resources, etc. The Courses of Action (COAs) generated contained an average of 150 actions and were developed in approximately 40 seconds. The different COAs were generated using alternative mission profiles and force packages. EXPECT could allow military planners to analyse these alternative COAs generated by O-Plan against a number of user defined domain evaluation criteria creates an evaluation matrix for a number of chosen COAs. From the analysis, military planners would be able to identify aspects of the COAs which were acceptable (e.g., low numbers of support personnel) and those which were not (e.g., a closure date greater than 29 days). An EXPECT evaluation matrix from a series of different COAs generated by O-Plan for a logistics scenario is shown in Table 1. This information could

	COA 1	COA 2	COA 3	COA4
AIRPORTS				
- number of airports	1	1	1	2
- sorties per hour	315	315	315	480
- sq. ft. aircraft parking	2M	2M	2M	3M
SEAPORTS				
- number of seaports	1	1	1	2
- number of piers	6	6	6	15
- number of berths	6	6	6	16
- max. vessel size in ft.	600	600	600	765
- number of oil facilities	1	1	1	3
CLOSURE DATE	C + 29	C + 22	C + 23	C + 23
LOGISTICS PERSONNEL	1154	5360	5396	7362
LINES OF COMMUNICATION				
- number of locations	1	5	7	6
- max. distance in miles	20	99	140	120
- air and sea?	yes	yes	yes	yes

Table 1: EXPECT's evaluation of several alternative plans generated by O-Plan

then be used to impose addition requirements on the planning system to seek to provide a better quality solution.

4 Additional Experiments

In addition to the three main themes explored by the evaluation experiments, a wide range of other experiments were carried out during the research. These were intended to demonstrate the capabilities being added to the O-Plan prototype, to explore the characteristics of problems that we wished O-Plan to address, to explain to others how to encode domains in O-Plan, etc. These problems have been explored in two ways:

- by creating O-Plan Task Formalism domain descriptions to give to O-Plan or extending some of the domains already used with O-Plan such as House Building or Pacifica. Many of the TF domain descriptions created for these experiments are provided with the O-Plan release in the `demo/tf` directory.
- by creating a number of experiments which aim to show the implications on the search spaces of the use of different types of knowledge and search techniques.

The experiments are described in the following subsections.

4.1 Scientific Experiment: Evaluation of O-Plan Condition Types

The main aim of the experiment was to take a fresh look at condition types and in particular the need for each type and the ways in which they should be handled in future implementations of O-Plan. The reason for this reappraisal was to address our concerns that there was confusion and criticism in the technical literature [5] about the use of the various condition types, and that a great deal of effort was required in encoding some domains for what seemed to be a small gain in search efficiency or in the quality of plans being presented.

The use of domain knowledge to restrict the plan search space is vital in any large scale problem, as the use of syntactic information concerning a condition is inadequate. The O-Plan team believe that one effective way to provide this knowledge to a planner is via condition types. Some condition type information can be gathered by lexical analysis of a problem definition. However, at present AI planning researchers know of no way to automatically deduce some of the information we can gather from user-defined types and conditions.

The experiment was centred around the need for each particular condition types. The first point which was addressed was to provide three statements for each O-Plan condition type:

- **Purpose**: This describes the condition in domain terms for use by the domain encoder and describes the circumstances under which the condition should be used.
- **Definition**: This describes the condition in planner terms and describes in more detail how the planner goes about dealing with the condition type on behalf of the domain encoder.
- **Examples**: This clarifies of the use of a condition type.

In providing a description of each condition type in terms of its purpose and definition it became necessary to define further the meaning of a plan level and the plan circumstances in which a condition could be evaluated (i.e., when to trigger the agenda entry to have the condition satisfied). The original definition of a plan level was too loose and vague to be used with the emerging definition of condition types and as a result a cleaner and more precise definition of a level was produced. The time at which an agenda entry is released (or triggered) for processing is very important in the search for a plan. The function of the triggers is to ensure the agenda entry is released for processing when it is possible to process the agenda entry in the planning process. By developing a clearer understanding of the ways in which conditions can be satisfied and maintained it was possible to define a cleaner and more precise definition of triggers. Full details of this evaluation can be found in [14] and the results were published in [31].

At present, one of the current release demonstration domains cannot be encoded with the tighter definitions. This is a block stacking domain (`blocks-2.tf`) used to show examples in earlier planners such as Nonlin [27] and Noah [25]. On study it became clear that the domain is encoded in a way which limits the solution space artificially. This was necessary for earlier planners, but is not necessary for O-Plan. The encoding of (`blocks-1.tf`) is a general purpose and improved description of this problem, but it would not have been possible to use this with Nonlin and Noah. From O-Plan version 3.1, (`blocks-2.tf`) will be removed from the demonstration suite so that a tighter definition of condition types can be imposed in the future.

The result of the experiment has been a better understanding of the use of condition types and the re-engineering and testing of *all* O-Plan test domains to comply with the new condition definitions.

4.2 Programmatic & Scientific Experiment: Economy of Force

There had been discussion within the ARPI community during 1993-4 which indicated a belief that so-called "generative planners" were inherently incapable of finding to solutions to problems in which the choice of a single action (operator schema) was necessary to address two or more separate problem requirements. This is sometimes called "economy of force". It can be one of the domain elements of evaluation to guide choice of better plans. Ginsberg at the University of Oregon had provided a simple island evacuation domain description in which such a single action choice was necessary to find the shortest solution to a problem[1].

O-Plan does contain all economy of force solutions in its search space, as it is designed to be systematic in preserving search space completeness (modulo restrictions on the search space deliberately encoded by a domain writer through features provided for this purpose - such as condition typing). In order to prove this, the example from Ginsberg was coded in O-Plan TF and provided to O-Plan. As expected, this demonstration showed that O-Plan is easily able to find solutions to such problems.

The O-Plan condition satisfaction procedure (Question Answering) and the Operator Schema choice routines in O-Plan do in fact currently choose between open choices using a single criteria, but they preserve all choices systematically. These choices are available to the search space

[1]Personal Communication.

controller in O-Plan. The O-Plan design allows for the incorporation of heuristic prioritisation of choices made by the operator schema choice function and the pre-ordering of choices available via Question Answering to satisfy conditions. Such heuristic prioritisation is anticipated to support a range of domain dependent elements of evaluation (as described in [17]). This can include economy of force (or as we term it "kill-two-birds-with-one-stone") choice prioritisation. Note that it is not possible to build this heuristic in to a general purpose planner as a hard wired prioritisation routine, since in some domains economy of force is to be avoided. It can also run counter to other preferences such as robustness in plans.

4.3 Scientific Experiment: Missionary and Cannibals

Scientists at Rome Laboratory used the Missionary and Cannibals Problem during 1993-4 to compare O-Plan and SIPE-2 [20]. While the Missionary and Cannibals Problem is not in the problem class for which O-Plan is designed (since it is essentially a mathematical *puzzle*), we used a range of Missionary and Cannibals Problem descriptions in O-Plan TF to demonstrate features of O-Plan prior to support for numeric handling and compute conditions (for external function support [28]) being added. These experiments were done with version 2.1 of O-Plan – the first release to the ARPI CPE. This showed that the Missionary and Cannibals Problem could be encoded using successor arithmetic.

Following the addition of numeric and compute condition support to O-Plan in version 2.2 (the second release to the ARPI CPE in July 1994), the Missionary and Cannibals Problem was recoded to act as a test domain for these features and to show that improved handing of the domain was possible. The Missionary and Cannibals TF encodings for the early and later experiments are available in the O-Plan release within the `demo/tf` directory.

4.4 Scientific Experiment: Spanner

O-Plan includes handling for the satisfaction of conditions within operator schemas, where the introduction of actions to satisfy the conditions turns out to require the insertion of new actions into the plan before the temporal scope of the schema which contains the condition [31]. The O-Plan team have for some time explained that other planners designs are not allowing this possibility in their search spaces. This means that they are *designed* to be incomplete. Some plans that a domain encoder might expect to be possible will not be admitted. The benefits for these planners is that their search spaces can be significantly smaller.

O-Plan provides support which will allow all solutions to be found including those needing the introduction of actions into a plan which "span" the area from the start of the plan to the point at which the condition is needed within the operator schema expansion (rather than just being in the gap between the beginning of the operator schema expansion and the point where the condition is needed).

A simple domain description called `spanner.tf` has been written to describe a very simple domain in which the "spanning" capability is required in a planner. This domain will not be amenable to solution by other planners designed with the more restricted definition of the legal temporal scope for the insertion of new actions to satisfy an achievable condition. This domain when run on O-Plan shows that O-Plan correctly identifies solutions requiring this capability.

Adding this capability to O-Plan has some serious consequences for comparative trials of O-Plan versus other planning systems. As far as we are aware, O-Plan is the only planner to have identified and remedied this problem. The search spaces introduced by handling it make for larger numbers of choices for a range of domains, some of the worst being the "puzzle" orientated domains used by many researchers to test their systems. Even simple problems in large plans can lead to very many more open choices if this capability is added.

Up to and including version 2.3 of O-Plan, the final release to the ARPI CPE from the O-Plan project's work in July 1995, the default handling for achieve conditions assumed that the system should allow for "spanning" solutions to be found. It is anticipated that a future release of O-Plan will alter the default handling to limit solutions to the temporal scope of the operator expansion in which the condition is introduced. However, O-Plan will continue to provide the more comprehensive "spanning" solution as necessary, and this will be able to be switched on by simply altering a default to the TF Compiler – using `achieve_after_point`.

4.5 Scientific & Demonstration Experiment: Proof of Concept for Mixed Initiative Planning

This experiment provided an early proof of concept demonstration for Mixed Initiative Planning (MIP) within the O-Plan framework [29]. The demonstration was based on the PRECiS/Pacifica domain and showed how the user could provide manual control over some of the choices being taken by the planner. Interaction with the system was via the KS-USER knowledge source and one of the aims of the experiment was to demonstrate that the improved functionality within this knowledge source could provide basic MIP support.

The choices which the user could control were:

- **Choice of schema**:
  The evacuation plans available for a given trip (i.e., city to city) were via helicopter or via ground transports.
- **Choice of variable binding**:
  Once the evacuation plan had been defined a particular transport asset had to be allocated, e.g., a ground transport plan could use any of the ground transports available.
- **Choice of back track point**:
  If the developing plan became invalid, e.g., due to unsatisfiable time or resource constraints then the user could choose which alternative partial plan was to be used as an alternative candidate. Planning would then continue from this partially generated plan.

The schema library for this domain contained 25 schema which defined alternative missions, transport plans, fuel (diesel and aviation), transport assets (helicopters and trucks). The COAs generated contained between 15 and 25 actions and were developed in approximately 40 seconds (discounting time taken for decision making by the user).

The demonstration showed that the O-Plan system could provide a framework within which the user could experiment with MIP related issues and that the support provided by the KS-USER knowledge source was sufficient for basic experimentation with MIP. The demonstration was shown to the ARPI Programme Managers during the O-Plan project review meeting in May 1994.

4.6 Scientific Experiment: Dealing with Plan State Variables

The aim of this experiment was to investigate the use of maintaining tighter information on the possible values for different plan state variables. A Plan State Variables (PSV) can be restricted to disallow a certain value or to require that its value be different from that of another PSV. The latter case is called a "not-same" constraint in O-Plan. The problem of dealing with the co-designation/non-codesignation of constraints has been a topic of research for many years. Others researchers [3] has developed schemes which attempt to solve parts of this problem. The work of the O-Plan system aims to develop research in this area further.

Each PSV has a "possibles-cache" that lists the values the PSV might take. Restrictions can remove values from the possibles-cache. If the PSV is left with only one possible value, it must be bound to that value; if it's left with zero, the plan is invalid.

The PSV Manager was changed to extract more information from *combinations* of not-same constraints. This was done only when a restriction is added to a PSV, leaving two or more values in PSV's possibles-cache. Then the PSV Manager looks at the PSVs listed in variables' not-sames constraints to check how many of the variables possible values they might take in combination.

The basic idea can be explained by an example. Suppose P-1, P-2, and P-3 all have A and B as their only possible values. Suppose P-2 and P-3 must have different values and that we then restrict P-1 to be different from both P-2 and P-3. Clearly, with three variables and only two possible values, there aren't enough values to go around. The aim is to detect cases of this sort and force the planner to abandon invalid plans earlier.

This change to the PSV Manager did not result in a noticeable increase in overall run-time and significantly reduced the number of O-Plan problem solving cycles required for certain tasks. For instance, the Pacifica task Blue Lagoon went down from 259 O-Plan cycles to 124. A more significant effect was that some tasks became practical (in terms of acceptable run time) for the first time.

4.7 Scientific Experiment: Agenda Choice

The aim of this series of experiment was to investigate the impact on the search space of different schemes for choosing ready to run entries from the agenda. The agenda contains "issues" that must be resolved in order to construct a complete plan: actions to be expanded, conditions to be satisfied, variables to be bound, etc. The order in which the issues (agenda entries) were processed can affect how quickly a plan is found and which plan is found. The different schemes tried were aimed at significantly improving the planner's performance by paying attention to plan levels and by exploiting heuristics that had been developed in earlier Edinburgh planning work. These heuristics include Branch1/BranchN factors which are used to select the most constraining choice next [6]. These have also been studied recently as the "least cost flaw repair" heuristic by Joslin/Pollack [18] who showed that it does lead to search space reductions. Recent work reported, for example at IJCAI-95 in Montreal, has begun the refinement of these methods.

The outcome and results of the experiment were mixed. As anticipated, no simple fixed prioritisation scheme improves performance on all problems. An adaptive and opportunistic scheme is what we are seeking which makes use of constraints in the plan and domain knowledge. The experiments also highlighted problems with individual techniques and these were as follows:

Plan Levels

One problem with using levels is that many of the current O-Plan TF domain definitions were written without a clear hierarchical model and a certain amount of rethinking concerning their encoding is required. In addition we are aware that the O-Plan agenda choice priority mechanism is being used to ensure the planner follows a certain planning "algorithm" (e.g., expands are done before certain types of condition are satisfied). It would be better if these mechanisms were separated from other types of choice to allow improved search control. This is the subject of future O-Plan research.

Branch-1 and Branch-N Estimators

O-Plan maintains two estimators of the branching factors for agenda entries – Branch-1 and Branch-N. Branch-1 is the number of "top level" possibilities that will be considered when an issue is processed. Branch-N is an estimate of the possible final number of alternatives for this issue. When branch-1 is 1, there is only one possibility and the planner has a single committed choice. There is an intuitive and informal argument to the effect that the Planner should prefer committed choices and process them first. All issues on the agenda will have to be processed. Some will create branches in the search space, and all remaining issues will have to be processed in all branches. If committed choices are done first, they will be processed only once. Moreover, they may constrain the plan, thus making things easier (in a sense) when processing other issues. There is also some empirical evidence that it helps to prefer forced moves [18].

However, the experiment showed that preferring committed choices can make things worse. Here, it is important to distinguish between two questions:

1. Is it better overall to process forced moves first?
2. Does it always make things better, or are there some cases where it makes things worse?

The aim of the experiment was to address the second question, not the first. Joslin's work [18] addressed the first problem.

To see that things *can* get worse locally, consider the following example. Suppose the agenda includes items A and B, that A has branch-1 = 1 while B has branch-1 > 1, and that A and B are independent in the sense that processing one will not affect how we process the other. Now suppose that when we process B we will always reach a dead end, so that the Planner must backtrack, and that processing A will not reach a dead end. If we process A before B, the time spent processing A will be wasted. This lead to the consideration of the cost involved in the processing of an agenda entry and that processing some agenda entries was more "costly" (in terms of the amount of effort expanded) than others. Earlier versions of O-Plan did try to maintain knowledge source activation estimates for this reason. Further experimentation is required to resolve this question within O-Plan.

4.8 Scientific Experiment: Alternative Choice

The aim of this experiment was to validate a number of different schemes for chosing alternative plan states to backtrack.

When the Planner cannot continue in the plan state it is currently considering, or chooses not to continue, the Agenda Manager (AM) is asked to pick one of the available alternatives so that the planner can continue from there. That is, the planner returns to some earlier decision and makes a different choice. Alternatives represent such decisions as which schema to use when expanding an action or which value to give to a variable.

Since alternatives are data structures, rather than being expressed procedurally in the Planner's code, the AM can employ a number of different search strategies when deciding which alternative to try. This is done, in part, by assigning each alternative a cost. Both the cost function and the AM's choice method can be redefined.

Initially, a very simple cost function for an alternative was used by counting the number of actions in its associated the plan state, and the choice method was to choose the lowest-cost alternative. If more than one alternative had the lowest cost, the most recent one created was taken. (This was done implicitly by the way the list of alternatives was sorted.)

A number of different cost functions and choice methods were tried and evaluated by planning for a range of O-Plan demonstration tasks. The different schemes investigated were as follows:

- Split the cost function into two parts that would be added, one to measure the work done so far, and one to estimate the work still to be done. This is similar to algorithms such as A* [21] and had some benefits in a number of domains.
- Add a depth-first element by choosing the most recently created alternative, rather than the one with lowest cost, in certain cases. This would provide part of the "local best than global best" strategy. We are seeking to use this in O-Plan. (The rest would be provided by knowledge sources that did some local search on their own.) Of the different schemes used this was by far the most effective change of all the ones described.
- Limit the amount of work done before seeing if an alternative might be better. After a given number of O-Plan problem solving cycles, the AM would switch to a better-rated

alternative even if it was still possible to continue from the current plan state. This had no significant improvement in performance in the set of test domains and in some cases made things worse.

In addition to these schemes a number of different cost functions were tried, stopping once an improvement was found.

5 Summary

This paper describes the evaluation experiments conducted as part of the O-Plan project. Each of the experiments conducted has been categorised according to the ARPI Evaluation Handbook. The O-Plan system has addressed three types of ARPI experiment: Programmatic, Demonstration and Scientific. A number of experiments have been described from each of these categories detailing the aims of the experiments, the method used and the conclusions and results which were found. This approach has been taken since the O-Plan project took a domain problem driven perspective to validate the approach of a specified planning architecture while allowing for the integration of new scientific ideas. The principal milestones for the project comprised three annual demonstrations and one TIE as follows:

- Year 1 demonstration which showed a cut-down version of an ARPI Integrated Feasibility Demonstration IFD the IFD2 scenario running in the O-Plan system,
- Year 2 demonstration which showed how a rich model of resources could be used to improve the solutions provided by a planner
- Year 3 demonstration which showed how O-Plan could be employed in a command, planning and control scenario to deal with changes occurring on the environment and in the overall task
- linking of O-Plan with the EXPECT plan analysis tool from USC/ISI.

A number of additional experiments are also described which were used to evaluate the new functionality and capabilities being added to the system. The Appendix describes the algorithms used to implement the plan repair mechanism demonstrated in the Year 3 major demonstration.

References

[1] Arentoft, M.M., Parrod, Y., Stader, J., Stokes, I. and Vadon, H., Optimum-AIV: a Planning and Scheduling System For Spacecraft AIV, in *Telematics and Informatics* Vol.8, No.4, pp.239-252, 1991.

[2] Beck, H., TOSCA: A Novel Approach to the Management of Job-shop Scheduling Constraints in Realising CIM's Industrial Potential: Proceedings of the Ninth CIM-Europe Annual Conference, (eds. C. Kooij., P.A. MacConaill and J. Bastos), pp138–149, Amsterdam, 12-14 May, 1993.

[3] Choueiry, B.Y. and Faltings B., Interactive Resource Allocation by Problem Decomposition and Temporal Abstractions, in *Current Trends in AI Planning*, (eds. Backstrom C. and Sandewall, E.), (the proceedings of the Second European Workshop on Planning), Vadstana, Sweden, 1993.

[4] Cohen, P., Dean, T., Gil, Y., Ginsberg, M. and Hoebel, L. *Handbook of Evaluation for the ARPA/Rome Laboratory Planning Initiative*, February, 1994.

[5] Collins, G. and Pryor, L. On the Misuse of Filter Conditions: A Critical Analysis, in *Current Trends in AI Planning*, (eds. Backström, C. & Sandewall, E.), IOS Press, 1993.

[6] Currie, K.W. and Tate, A., O-Plan: the Open Planning Architecture, *Artificial Intelligence Journal*, Volume 51, No 1, 1991. Also available as AIAI-TR-67.

[7] Drabble, B., Conversion of SIPE-2 Domain Descriptions to O-Plan Task Formalism, O-Plan2 Technical Report ARPA-RL/O-Plan2/TR/1 Version 1, March 1993.

[8] Drabble, B., Applying O-Plan2 to Military Logistics Planning, O-Plan Technical Report ARPA-RL/O-Plan2/TR/9 Version 1, September 1993.

[9] Drabble, B., Comparison of the CAMPS system with O-Plan2: Architecture and Reasoning Capabilities, O-Plan Technical Report ARPA-RL/O-Plan2/TR/12 Version 1, January 1994.

[10] Drabble, B., NEO Scenarios for O-Plan2 Demonstrations, O-Plan Technical Report ARPA-RL/O-Plan2/TR/4 Version 2, March 1994.

[11] Drabble, B., Gil, Y. and Tate, A., *Acquiring Criteria for Plan Quality Control*, Proceedings of the AAAI Spring Symposium Workshop on Integrated Planning Applications, June 1995. Stanford Univerisity, CA, USA. Publishers the American Association for Artificial Intelligence, Menlo Park, California.

[12] Drabble, B. and Gil, Y., Yes, but why is that plan better?, Proceedings of the International Conference on Artificial Intelligence in the Petroleum Industry, 13th-15th September 1995, Lillehammer, Norway.

[13] Drabble, B. and Tate, A., The Use of Optimistic and Pessimistic Resource Profiles to Inform Search in an Activity Based Planner, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), AAAI Press, Chicago, USA, June 1994.

[14] Drabble, B., Tate, A., Dalton, G. and Reece, G., Condition Types in O-Plan2: A Discussion, O-Plan2 Discussion Note ARPA-RL/O-Plan2/DN/1 Version 1, January 1994.

[15] Fowler, N., Cross, S.E. and Owens, C., The ARPA-Rome Knowledge-Based Planning and Scheduling Initiative, *IEEE Expert: Intelligent Systems and their Applications*, Vol. 10, No. 1, pp. 4-9, February 1995, IEEE Computer Society.

[16] Gil, Y., Refinement in a Reflective Architecture, Proceedings of the Twelfth International Conference on Artificial Intelligence, Seattle, WA, USA. August 1994. Published by AAAI Press/The MIT Press Menlo Park, CA, USA.

[17] Gil, Y., Hoffman, M. and Tate, A., Domain-Specific Criteria to Direct and Evaluate Planning Systems, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. M.Burstein), Morgan-Kaufmann, 1994.

[18] Joslin, D. and Pollack, M.E., Least Cost Flaw Repair: A Plan Refinement Strategy for Partial Order Planning, Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, USA, August 1994.

[19] Lehrer, N., (ed.), ARPI KRSL Reference Manual 2.0.2, February, 1993. ISX Corporation.

[20] Ludlow, C., Looking at O-Plan2 and Sipe-2 Through Missionaries and Cannibals, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop, (ed. M.Burstein), Morgan-Kaufmann, 1994.

[21] Nilsson, N.J. *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill Book Company, New York, USA, 1971.

[22] Reece, G.A. and Tate, A., Synthesizing Protection Monitors from Causal Structure, Proceedings of the Second International Conference on AI Planning Systems (AIPS-94), pp. 146-152, AAAI Press, Chicago, USA, 1994.

[23] Reece, G.A.. and Tate, A., The Pacifica NEO Scenario, O-Plan2 Technical Report ARPA-RL/O-Plan2/TR/3 Version 1, March 1993.

[24] Reece, G.A., Tate, A., Brown. D. and Hoffman, M., The PRECiS Environment, Presented at the ARPI Workshop as part of AAAI-93 in Washington, DC. Also available as O-Plan2 Technical Paper ARPA-RL/O-Plan2/TP/10.

[25] Sacerdoti, E.D., *The Structure of Plans and Behaviour*, American Elsevier, New York, 1977.

[26] Smith, S.F., Ow, P.S., Potvin, J-Y., Muscettola, N. and Matthys, D., An Integrated Framework for Generating and Revising Factory Schedules, in *Journal of Operational Research Society*, pp539-552, Vol 41, No 6, 1990.

[27] Tate, A., Generating Project Networks, Proceedings of the Fifth International Joint Conference on Artificial Intelligence, William Kaufmann Inc, 1977.

[28] Tate, A., Planning and Condition Monitoring in a FMS, Proceedings of the International Conference on Flexible Automation Systems, Institute of Electrical Enginers, London, July, 1984.

[29] Tate, A., Mixed Initiative Planning in O-Plan2, Proceedings of the ARPA/Rome Laboratory Planning Initiative Workshop at Tucson, Arizona, USA, (ed. M. Burstein), Morgan-Kaufmann, 1994.

[30] Tate, A. and Whiter. A., Multiple Resource Constraints and an Application to a Naval Planning Problem, Proceedings of the First Conference on Artificial Intelligence Applications, pp410-416, AAAI, Denver, Colorado, USA, December, 1984.

[31] Tate, A., Drabble, B. and Dalton, J., The Use of Condition Types to Restrict Search in an AI Planner, Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, USA, August 1994.

[32] Wilkins, D. *Practical Planning*, Morgan-Kaufmann, 1988.

6 Appendix O-A – Plan Repair Algorithm

The aim of this Appendix is to describe the approach taken in the O-Plan third year demonstration to the problem of repairing an ongoing plan in the face of changes from the environment and in the task itself.

The aim of the demonstration was to show O-Plan in a integrated command, planning and control environment in which a number of changes were occurring in the domain and in the overall task requirement. The demonstration showed how O-Plan could integrate a number of pre-assembled repair plans, e.g., repairing a blown engine, repair a flat tyre, etc; into an ongoing and executing plan. For the purposes of the demonstration the integration of the repair plans into the ongoing plan was accomplished via the planning agent. However, the techniques and methods used could easily have been integrated into the capabilities of a separate execution agent.

The plan repair algorithms deal with two types of plan state entity as follows:

- **Table of Multiple Effects (TOME) Entry**
  A TOME entry is created for each effect asserted in the plan and is of the form `pattern = value at node-end`, i.e., `p = v @ n`. For example, `(colour_of ball) = green at end_of node-1`

- **Goal Structure Table (GOST) Entry**
  A GOST entry is created for each protected range in the plan and is of the form `condition_type pattern = value at condition-node-end from contributor-node-end`, i.e. `p = v at c from [e]`. This specifies that the pattern is asserted at the contributor-node-end and is required at the condition-node-end. For example, `unsupervised (colour_of ball) = green at (begin_of node-1-2) from (end_of node-1)`. Multiple disjunctive constraints are possible.

Each of these entities is maintained by the O-Plan TOME and GOST Manager (TGM). A plan failure occurs when one or more of the GOST entries are broken, i.e. a contributor of a GOST entry is not asserted as expected or an external world event occurs which asserts extra effects into the plan which breaks the protected GOST range.

Plan repairs are dealt with by a number of knowledge sources. The knowledge sources are responsible for deciding what action to take when a plan failure has been detected by the TGM and for making a repair to the effected plan. A number of knowledge sources have been defined (in addition to those required for plan generation and agent capabilities) to allow the system to repair broken GOST entries as follows:

- KS-EXECUTE starts off the execution of the plan and initialises the datastructures needed by the other execution support knowledge sources.

- KS-EXECUTION-SUCCESS updates the list of executed node-ends to indicate the node-end has executed successfully.

- KS-EXECUTION-FAILURE decides which GOST entries have been effected and posts a number of KS-FIXs to deal with each broken GOST entry.
- KS-UNEXPECTED-WORLD-EVENT deals with the consequences of the occurrence of an unexpected world event and posts a number of KS-FIXs to deal with each broken GOST entry.
- KS-FIX deals with the resatisfaction of a GOST entry broken by a plan failure.
- KS-CONTINUE-EXECUTION decides which node-ends can begin execution after the plan has been repaired.

The main problems dealt with by the repair mechanisms are as follows:

- **Execution Failure**:
  An execution failure occurs when one or more of the expected effects of a node end fail to be asserted. For example, the node end corresponding to the end of the action `Check_out_ground_transport` should assert that the status of the engine and tyres was fine, e.g., `(engine_status gt1) = working` and `(tyre_status gt1) = working`. These may not in fact be satisfied after the plan failure. This type of failure may cause problems if the expected effects of the action are needed to satisfy the preconditions of a later action. For example, the evacuation of people from an outlying city can only precede if the tyres and engine of the ground transport continue to function correctly.
- **Unexpected World Event**:
  Unexpected events cause effects in the world which make planned actions fail. For example, a landslide event may have the effect `(road_status Abyss_to_Barnacle) = closed` and this would interfere with any action requiring the road to be open.

The description of the algorithms of the execution and plan repair system is divided into three main sections. Section 1 describe how the system maintains an execution fringe of those node ends which have been executed and those which are awaiting execution. Section 2 describes how the system deals with plan failures and Section 3 describes how unexpected world events are dealt with.

1 Maintaining the Execution Fringe

An activity, dummy node or event is represented in an O-Plan plan as a node with two ends, a `begin_end` and an `end_end`, each of which is represented via a time point. Conditions and effects can be attached to either end of a node and monitored by the execution system. The execution system reasons purely in terms of node-ends and not in terms of activities or events. The system is driven by the success and failure messages from the model of the world. The system calculates an "execution fringe" of the activities in the partially ordered plan which are currently being executed. The reason for maintaining the execution fringe is to provide a context within which replanning can take place and to provide a focus point when considering where to insert repair actions, i.e., after all node-ends which have executed and before any node-ends waiting to execute. This point is known as the plan's *neck point* and a single dummy node can be added to the plan by the repair algorithm to neck the plan at this point on need.

The execution fringe instantiated as a list of the node-ends which are currently ready for execution, i.e., node-end E is ready for execution when all node-ends that are linked before it have successfully executed and the required condition contributor are available. This ensures that all explicit ordering constraints (i.e., links from the orderings clauses of O-Plan TF schemas) are satisfied and that all node-ends that provide effects needed by conditions at E have provided those effects. The actual "ready to execute" check considers only whether all the node-ends linked before E have been executed, regardless of whether the execution was successful. It assumes that any problems due to execution failures or world events have been fixed. (It is the responsibility of other parts of the system to ensure that this is so.)

The "ready to execute" check also ignores the temporal constraints in the plan. Temporal constraints are handled by putting node-ends that are ready to execute in a "departure queue" and not dispatching them, (i.e., sending them to the world simulator) until all node-ends with earlier due-times have been executed. This means that node-ends can be dispatched to the world simulator in advance of their actual execution time. The system must obey temporal constraints to this extent because it sometimes need to link a node-end after all the node-ends that have been executed and before all node-ends that have not. If the system looked only at links when deciding when to execute a node-end, it might execute E1 before E2 when the temporal constraints indicate they had to be the other way around. For example, if E1 has executed, E2 had not and the system trys to link something after E1 and before E2 the temporal constraints will not allow it.

During execution, node-ends take on execution status values in the following order:

- **:not-ready**
  Not ready for execution, either because the system has not examined it yet or because some precondition has not yet been met.
- **:ready**
  Assigned when the system determines that all preconditions have been met and the node-end has been queued for execution.

- **:sent** Assigned when the system sends an the appropriate execution message to the execution support system.
- **:finished** Assigned when the system receives a success or failure message from the execution support system.

A node-end with status `:ready` can have its status set back to `:not-ready` when KS-CONTINUE-EXECUTION rebuilds the departure queue. This happens when the system has finished changing the plan after an execution failure or an unexpected world event occur.

2 Dealing with Execution Failures

When an execution failures occur at a particular node-end some of the effects due to be asserted may not occur. They're returned from the execution monitoring system to the planning agent as a list of failed-effects. The task of the planning system is to fix the plan so that any condition that needed one of the failed effects as a contributor is satisfied in some other way. The fix can be relatively simple, e.g., there is already another contributor in the GOST entry or there is a suitable alternative contributor already present in the plan. If these simple fixes cannot be applied then the system will attempt to add a new action to the plan through which a repair plan can be introduced. However, if there are no conditions requiring the failed effects then the execution "failure" can be ignored.

The main algorithm used by the system to track execution and initiate repairs is as follows:

- Mark the node-end as having been executed.
- If there are no failed effects, then a repair is not needed.
- If there are failed effects then remove the TOME entries that correspond to them
- Determine which GOST entries are affected by the failed (removed) effects. If there are none, then a repair is not needed.
- By reaching this point there is a definite failure and the system needs to instigate a repair activity. The search for a repair plan is as follows:
  - Search through the affected GOST entries in turn.
  - If a GOST entry has more than one contributor, check if any are still valid (It is also possible that more than one is from the same node-end, so that a failure might take out more than just one).
  - If there are still some valid contributors, reduce the contributor list; otherwise record the GOST entry as truly broken.
  - If no GOST entries are truly broken, then the repair is complete.
- By reaching this point some GOST entries are truly broken, and the planner will need to post agenda entries for each of the broken GOST entries. For each broken GOST entry, the planner posts a KS-FIX agenda entry which has very similar functionality to the knowledge source KS-ACHIEVE. Its function is to satisfy an `achieve` type condition [31] at a specified node-end in the plan either by:
  - finding an existing alternative contributor in the plan.
  - bringing in additional actions (a repair plan) which asserts the appropriate effect.
- The `achieve` condition for the broken GOST entry will be *after* the *neck* point that is linked after all node-ends that have been executed so far (the execution fringe).

- If there is only one node-end in the execution fringe, use it as the neck point. Otherwise, add a new neck node, link it after all members of the execution fringe.
- Once the neck point has been identified the system can carry out the lower level detail of the repair. The algorithm is as follows: (Where A is the end of the neck node)
  - Step through each of the truly broken GOST enties
  - If the condition is not an `unsupervised` condition type indicate that the contributors for it are not yet defined. Post a KS-FIX to re-establish the condition at the required point, i.e. to "achieve p = v at e after A".
  - If g is a supervised condition the `p = v` must be established over a range, rather than just at a point.
  - Create a new dummy node `d` to act as the "delivery point" and Link `d` after the neck point, before the effect and before all node-ends that are spanned by the condition and have not yet been executed.
  - Change the conditions value to have `d` as the contributor and give `d p=v` as an effect in the TOME.
  - Post a KS-FIX to re-establish `p=v at d`, i.e., to "achieve p = v at d after A"

  The system must be consistent in its use of a single end of d for both conditions and effects to avoid "gaps" in the goal structure which would effect the meaning of the plan. `d` is linked into the plan like this:

  end_of after-node –> begin_of d –> end_of d –> spanned node-ends

  To ensure no gaps are left the system uses the default value for condition_contributor_node_end (obtained from the TF compiler). This is referred to as node-end "ccne_of d" The systems needs to re-establish p=v at the same end of d, so the `achieve` condition should "achieve p = v at ccne_of d after A".
- The current list of alternatives is altered so that the system does not backtrack over alternatives before the fix.
- Post a KS-CONTINUE-EXECUTION to continue execution after the fixes have been made.

3 Dealing with Unexpected World Events

If an unexpected world event occurs in the world that is not anticipated in the current plan. The event is reported as a time, an event pattern, and a list of effects (`pattern` and `value` pairs). For instance, the occurrence of a landslide event would be reported as:

```
event {landslide} with effects
   {status road-a} = blocked,
   {status road-b} = blocked;
```

Events are treated the same way as plan activities except they are not placed in the plan until they have occurred. The effects may break GOST ranges in the plan and if so, the planner must try to satisfy those conditions some other way. However, even if no GOST entries are broken, the planner needs to add a node to represent the world event. This is because, even if the event's effects don't make any difference now, they may matter later on.

The new node represents something that has definitely and already happened. So it must be linked after all node-ends that have already been executed and before all node-ends that have not yet been executed. The new node's effects can't be added until we've removed any broken GOST entries. Otherwise the TOME and GOST Manager might try to preserve those entries when we put in the effects.

The algorithm for dealing with unexpected world events is as follows:

- Add an event node, E, to represent the world event. Link it after the execution fringe. Mark E as having already been executed.
- Edit the GOST to remove any contributors that can no longer contribute, and get a list of the truly broken GOST entries.
- For each truly broken GOST entry g:

  Set up for and post a KS-FIX agenda entry as in the case of an execution failure using end_of E as the neck point.
- Add the world event's effects at end_of E.
- If there were no truly broken GOST entries, then the repair is complete.
- Otherwise, the current list of alternatives is altered so the system does not backtrack over alternatives before the fix. Post a KS-CONTINUE-EXECUTION to continue execution after the fixes have been made.