U-Plan: An Approach to Planning Given Uncertain and Incomplete Information

T.M. Mansell

DSTO-RR-0103

DEPARTMENT ◆ OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

U-Plan: An Approach to Planning Given Uncertain and Incomplete Information

T. M. Mansell

Maritime Operations Division
Aeronautical and Maritime Research Laboratory

DSTO-RR-0103

ABSTRACT

This paper describes research into planning in an uncertain environment. In particular, it describes U-Plan, a planning system that constructs quantitatively ranked plans given an incomplete description of the state of the world. Information acquisition operators are then applied to choose between plan alternatives. U-Plan has been trialled in a simulated one-on-one air combat domain. Results of the evaluation in this report included number of plans produced, sensitivity and timing data.

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Victoria 3001

Telephone: (03) 9626 7000
Fax: (03) 9626 999

AR No. 010-205
April 1997

APPROVED FOR PUBLIC RELEASE

U-Plan: An Approach to Planning Given Uncertain and Incomplete Information

Executive Summary

A key aspect of intelligent activity is the ability to determine a course of action that is likely to achieve a desired goal. Within the field of artificial intelligence (AI) research, determining such a course of action is known as planning. Planning, whether for naval platforms or aircraft, typically requires a careful balance between the execution of reactive actions and achieving long term goals. At present manoeuvre planning is best performed by human operators. However, due to the volume of data produced from modern sensor systems, the potential exists for the human planner to succumb to information overload, thereby increasing the chance of missing superior strategies/tactics. One solution is to provide the operator with suitable decision aids that improve his ability to make decisions in a timely manner. The development of an AI planning system capable of operating in a dynamic and imprecisely described environment would provide the operator with a tool capable of producing novel, alternative plans of action. These plans could be included with the operator's own plans for consideration before a final decision is made. Decision aids that improve the command's ability to perform situation awareness and generate a timely response can only improve the operational effectiveness of platforms.

Examined and analysed in this report is U-Plan, a planning system that constructs quantitatively ranked plans generated from an imprecise description of the state of the world. The planner takes as input what is known about the world, and constructs a number of possible initial states with representations at different abstraction levels. A plan is constructed for the initial state with the greatest support, and this plan is tested to see if it will work for other possible initial states. All, part or none of the existing plans may be used in the generation of the plans for the remaining possible worlds. Planning takes place in what is termed an abstraction hierarchy, where strategic decisions are made before tactical decisions. A super-plan is then constructed, based on merging the set of plans and the appropriately timed acquisition of essential knowledge (which is used to decide between plan alternatives). U-Plan has been trialed in a simulated one-on-one air combat domain. Results of the evaluation in this paper included number of plans produced, sensitivity and timing data. U-Plan usually produces a super-plan in less time than a classical planner would take to produce a set of plans, one for each possible world.

This report addresses both knowledge representation and implementation issues in planning under uncertainty. The approach presented seems to suit a particular class of application domain, which we refer to as emergency response domains. In addition to a detailed analysis of a particular domain (the air combat domain), preliminary work on a submarine manoeuvre planning domain is also described in this report. U-Plan is not a real time planning system as it has been developed for concept demonstration. However, with suitable development a near real time system is achievable for application to domains where response time is measures in seconds (eg, the submarine manoeuvre planning domain), rather than milliseconds (eg, the air combat domain).

Author

Todd Michael Mansell
Maritime Operations Division

Todd Mansell graduated with a Bachelor of Science degree from Deakin University Geelong in 1988, majoring in Physics and Electronics. He continued on to obtained a first class Honours working in the field of digital image processing. In 1989 Dr. Mansell joined the Defence Science and Technology Organisation (DSTO) as a Cadet Research Scientist, and after a years service, began his Ph.D.. In 1994 he completed his Ph.D. in artificial intelligence (AI) with The University of Melbourne. While with DSTO, Dr. Mansell has worked on the application of information fusion techniques to Naval problems (including mine warfare and combat systems). His main research interests are information fusion, AI planning, and tactical decision aids.

Contents

1. Introduction

Planning has been a core area of AI research as it explores a key aspect of intelligent activity: the ability to determine a course of action that is likely to achieve a desired goal. Much work has been devoted to understanding various aspects of the planning problem, such as appropriate representations or computationally effective search strategies. Generally, the problem has been approached from two quite distinct perspectives. On the one hand, the problem has been presented as one of finding a 'provably correct' plan for achieving a stated goal in a static world for which there is a complete description. On the other hand, the problem has been presented as one of determining the next action to take in a dynamic world that defies complete description and for which the 'correct' action is determined by matching characteristics of the world to preconditions of predetermined responses. We wish to plan in circumstances that are intermediate to these two extremes. We want to determine a plan that may include alternative actions to achieve, at least to some degree, a desired goal in a world for which an incomplete description exists.

The worlds for which we wish to plan lack a complete description, and consequently alternative plans may need to be constructed when the description is insufficient to eliminate alternatives. We assume that additional information about a world may be acquired by knowledge acquisition activities. These activities are likely to incur a cost. We wish to find a plan that attains the desired goal by acquiring only necessary supplementary information. In addition, we only want to acquire supplementary information at the point in the plan where that information is needed to select from alternative actions.

The world in which we wish to apply our techniques is dynamic. Agents in our world may change the environment in unpredictable ways. However, we assume that the incomplete description of a world is static and begin planning by selecting our first action based on this description. We will reassess the world after an action is chosen, and determine whether it has changed sufficiently to require altering the current plan. The notion adopted is that the world at its most detailed may change during planning, but the more abstract concepts are unlikely to change significantly. Hence, the tactical detail of a plan may need modifying, but the plan strategy will still be relevant most of the time.

At present manoeuvre planning is best performed by human operators. However, due to the volume of data produced from modern sensor systems, the potential exists for the human planner to succumb to information overload, thereby increasing the chance of missing superior strategies/tactics. One solution is to provide the operator with suitable decision aids that improve his ability to make decisions in a timely manner. The development of an AI planning system capable of operating in a dynamic and imprecisely described environment would provide the operator with a tool capable of producing novel, alternative plans of action. These plans could be included with the operator's own plans for consideration before a final decision is made. Decision aids

that improve the command's ability to perform situation awareness and generate a timely response can only improve the operational effectiveness of platforms.

1.1 The Problem Description

This report specifically deals with the case where an uncertain and/or incomplete description of the environment is a likely burden to planning. It also assumes that the effects of actions are fully and accurately represented. As a result, unforeseen problems or errors in action execution are not considered during the planning process. The central problem dealt with in this report is the following:

GIVEN:

- limited information about the initial state of the world
- a set of goal states
- a suitably represented application domain where imprecise information is likely to be a restriction on the planner
- a domain that also sustains a number of planning solutions for most possible worlds used to characterise the environment
- a set of actions that can be applied to the given domain
- the ability to acquire information about the domain.

OBJECTIVE:

- to produce a plan that achieves the goals of the system by outlining a course of action to be taken in the domain. Where appropriate, this course of action should differentiate between a number of alternative actions by acquiring specific knowledge.

A number of difficult problems are encountered when developing a theory for planning, often resulting from the requirement to achieve specified goals through forecasting the consequence of actions. The evolution of planning has seen the successful development of many techniques for dealing with such problems. Thus, it is the intention of this research to build on previous planning methods.

1.2 U-Plan

U-Plan is a planning system that constructs quantitatively ranked plans given an incomplete description of the state of the world. U-Plan uses Dempster-Shafer (Shafer, 1976) theory to characterise uncertain and/or incomplete information about the state of the world. Planning takes place in an abstraction hierarchy where strategic decisions are made before tactical decisions. The planner takes as input what is known about the world, and constructs a number of possible initial states with representations at different abstraction levels. A plan is constructed for the initial state with the greatest

support, and this plan is tested to see if it will work for other possible initial states. All, part or none of the existing plans may be used in the generation of the plans for the remaining possible worlds. A super-plan is then constructed, based on merging the set of plans and the appropriately timed acquisition of essential knowledge, which is used to decide between plan alternatives. Figure 1 gives a general overview of the activities carried out by U-Plan and how they are used in relation to each other.

1.2.1 Possible Worlds

A major problem when planning given imprecise information about the environment is that it is not possible to construct one initial state that precisely and unambiguously represents the world. U-Plan assumes that an incomplete model of the world is all that is available, and uses a set of initial possible states (P-states) to describe what might be true of the world (discussed in detail in section 2). A P-state is a complete description of one possible world using propositional statements. Each P-state is described hierarchically with n levels of abstraction, where n is domain dependent and selected during knowledge engineering. Each of these levels is a complete description of a world at a specified abstraction level. The highest level of abstraction gives a coarse description of the world. The lowest level gives a detailed view of the world. Intermediate levels provide the description required to make a smooth transition between both extremes.

Initially, information sources provide U-Plan with a set of propositional statements that represent distinct aspects of the domain. This initial information, along with mapping functions (Shafer, 1976) (defined at knowledge engineering time and domain dependent) are used to construct a set of P-states that represent the possible worlds (as demonstrated in the top portion of figure 1). U-Plan uses the support and plausibility measures (defined by Dempster-Shafer theory) to calculate the weight of evidence attributed to each P-state, where support represents the degree to which the evidence advocates the P-state, and the plausibility representing the degree to which the evidence fails to refute the P-state.

1.2.2 Plan Representation

Associated with the problem of representing the world is the question of selecting actions, given that critical information relating to applicability (and effect) of that action will be unknown. This will depend largely on how the world is described, the representation of actions and the type of planning system used (hierarchical, or non-hierarchical, linear, etc.).

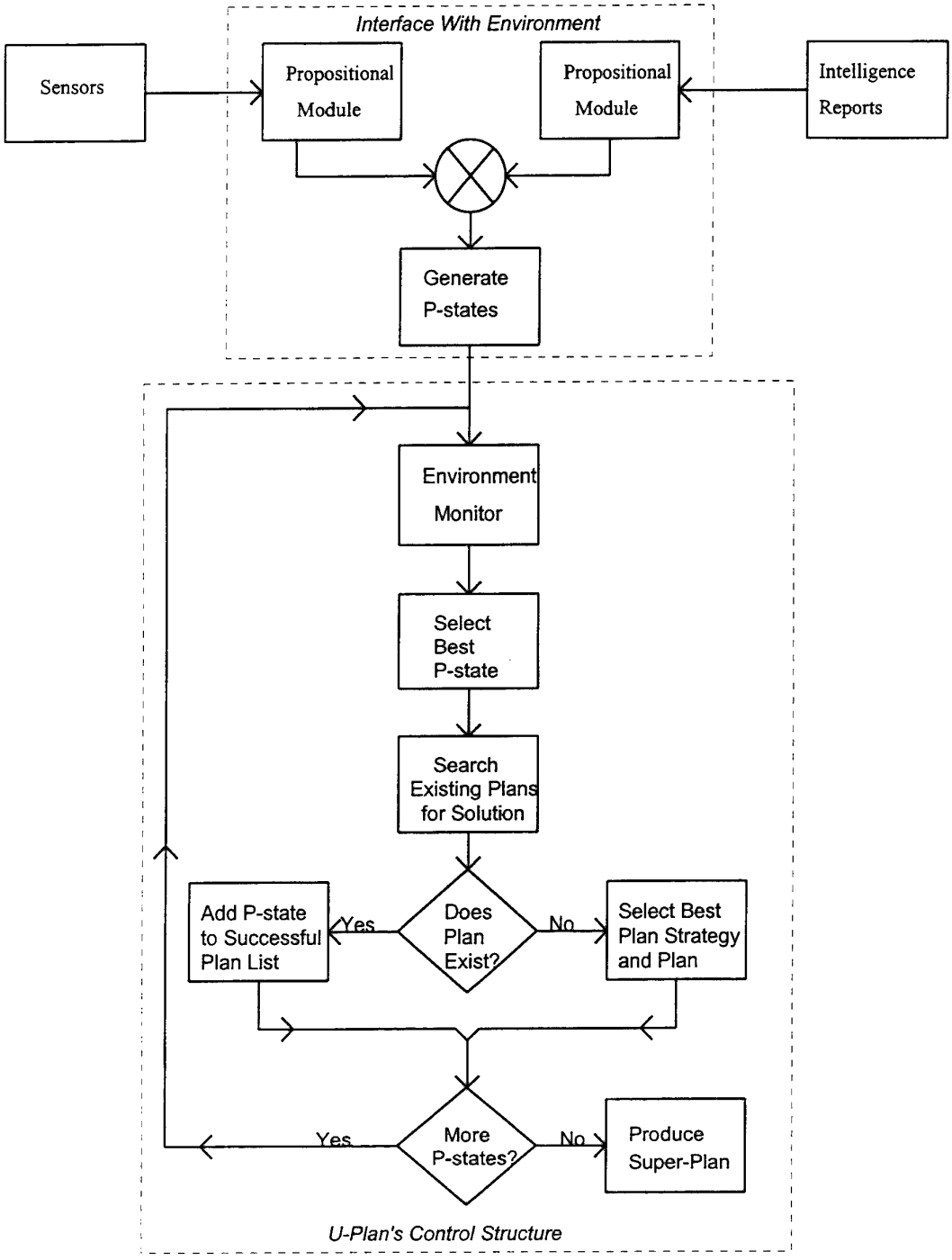

Figure 1: A flow diagram linking the activities pursued by U-Plan.

U-Plan uses a hierarchical approach to planning, as it significantly reduces the search space by first planning at abstract levels, and then expanding these abstract plans into more detailed plans (Chapman, 1987). At the highest abstraction level strategic decisions are made, while at the lowest levels of abstraction, tactical decisions about how best to implement the strategy are made. The representation of P-states at differing levels of abstraction was contrived in support of hierarchical planning, allowing decisions to be made using a P-state representation at an equivalently detailed level of abstraction. That is, strategic decisions are based on strategic information; and conversely, tactical decisions are based on tactical information.

The role of an action is to change the state of the world; the aim of an operator is to represent how applying that action will change the system's view of the state of the world. U-Plan uses reduction operators to give alternative methods for achieving the goal at a more abstract level or, at the tactical level, it describes the direct effects of an action on the P-state. To do this, information pertaining to when the operator should be applied, how it should be applied, and what effect it has on the world, must be encoded in each reduction operator. Fundamental to a U-Plan operator is its *plot*. This provides step-by-step instructions on how to perform the action represented by the operator. The description of the reduction operators also includes the specific representation of the conditions required before the operator may be applied. This includes what must already be true of the world (*necessary preconditions*), and what can be made to be true of the world (*satisfiable preconditions*).

Each operator includes a function for calculating the probability of succeeding given the current P-state. The availability of such a function is domain specific and may be obtained empirically (based on historical data), or subjectively (based on expert opinion). The probability of success does not provide sufficient information to select a reduction operator as it does not take into account the goals of the system. It is for this reason that, associated with each reduction operator, there is a measure of fulfilment representing the degree to which the reduction operator achieves its intended goal. The reduction operator's expected fulfilment (product of probability of success and fulfilment measure) is calculated during planning and utilised in reduction operator selection (see section 4.1).

In addition to the above information, U-Plan reduction operators include in their description: (1) an abstraction level corresponding to the P-state abstraction level on which they operate; (2) a list of postconditions; and (3) instructions to follow if the operator fails. This information is domain specific and often difficult to characterise.

1.2.3 Plan Generation

Many classical planning systems use a state-based search strategy to solve planning problems. To find a solution one applies operators to a state description until an expression describing the goal state is found. U-Plan uses a decision theoretic process (i.e. measure of expected fulfilment) to select the most appropriate action within an abstraction hierarchy. U-Plan's planning algorithm constructs totally ordered

nonlinear plans[1] which describe a sequence of actions that, when applied in a particular P-state, have a given probability of producing the goal state.

U-Plan provides a plan (by generation or reapplication) for all P-states with support and plausibility above a specified (domain specific) threshold. The plan generation algorithm constructs a plan for one possible world at a time, the first plan being constructed for the possible world with the greatest likelihood of representing the true world. A flow chart representing the process of generating a plan for a given P-state is presented in figure 2.

The hierarchical planning process involves selecting a strategy that achieves the goals of the system, reducing them to a set of executable operators. As in many hierarchical planning systems, this process will involve determining the goal, then: (1) selecting the reduction operator(s) that best achieves this goal at the next level of abstraction; (2) ensuring it may be applied to the prevailing P-state; and (3) applying that operator. This process is repeated from the highest level of abstraction to the lowest, producing a course of action that should convert the initial P-state to the goal P-state.

Goals, in many applications, are not precise requirements. Many general goals can be fulfilled to various degrees by achieving alternative subgoals. However, not all subgoals are equally likely to be achieved. It may be desirable for the planning system to include a method for selecting which subgoal best fulfils the goals of the planner. This selection process may be qualitative (as in Wellman, 1990) or quantitative. U-Plan's planning algorithm bases selection of each reduction operator on the calculation of the expected degree to which the goal should be achieved, termed *expected fulfilment*.

The domains in which many planning activities take place are dynamic. Other agents in the world may change the state in unpredictable ways. When developing an approach for planning under uncertainty, one should include a mechanism for dealing with the dynamic nature of an environment. This requires a method that balances a degree of long term planning (giving the planner long enough to develop and implement a strategy) and reactivity (giving the planner the ability to survive and represent the up-to-date state of the world). U-Plan expects to carry out the planning and execution of actions in a dynamic environment. To plan in a dynamic world, U-Plan uses a separate module to monitor changes in the environment, and will abort the planning process if there are changes to critical aspects of the environment.

[1]See Mansell (1994) or Veloso (1992) for an explanation of why totally ordered plans are not necessarily linear.

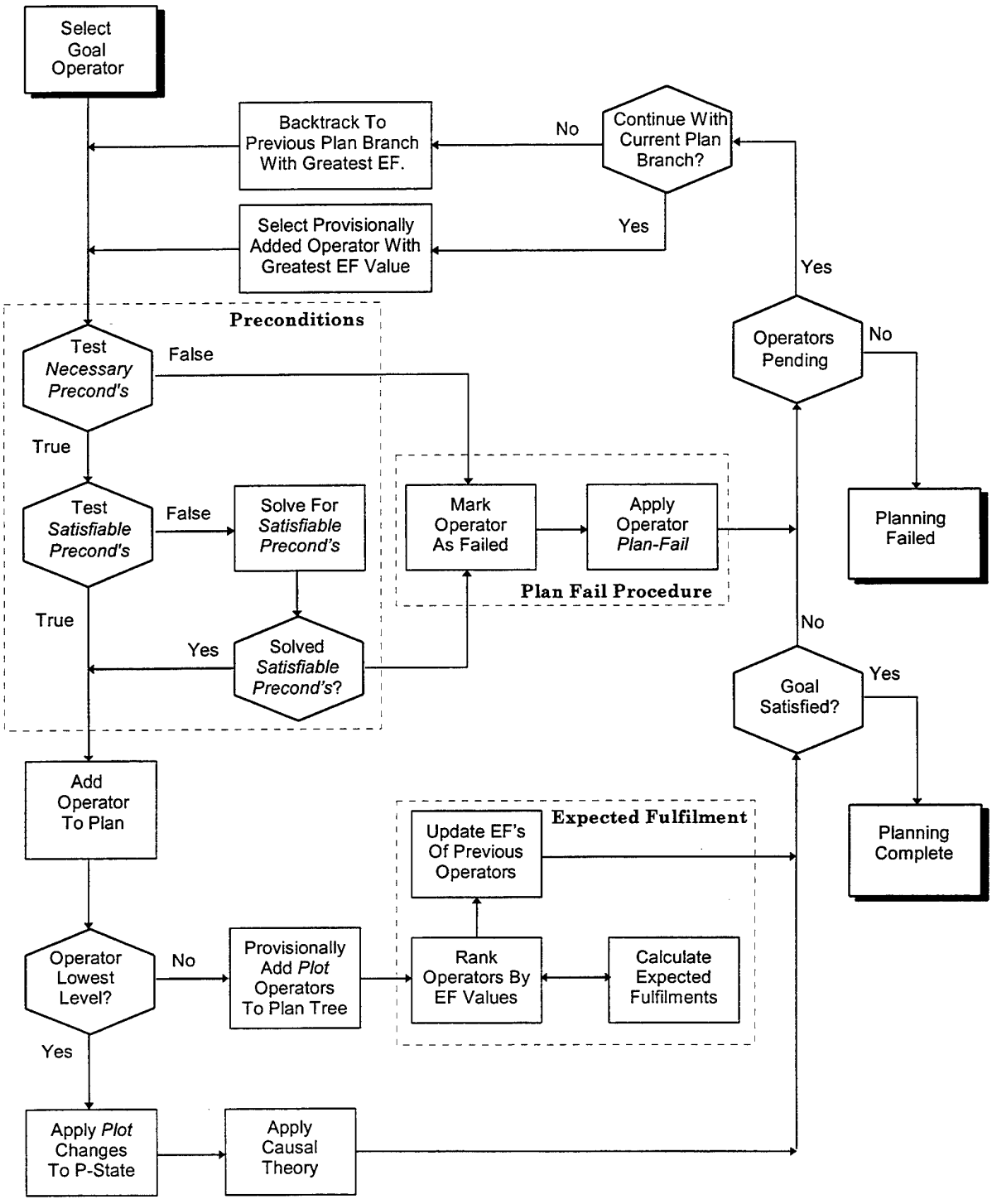

Figure 2: The flow diagram for the generation of a single plan.

1.2.4 Plan Reapplication

U-Plan implements plan reapplication in an attempt to determine if a plan, generated for one initial P-state, can be adopted for another initial P-state. The desired result is for U-Plan to produce fewer plans than the number of initial P-states. This is accomplished by first generating a plan for the initial P-state with the greatest support (as discussed in the previous section). This plan is then added to a list of existing plans. The next highest supported P-state is then selected and the existing plans examined to determine if any may be reapplied to the alternative P-state. The process of determining the suitability of existing plans, before generating a new plan for each P-state, continues for all P-states with a support and plausibility over a given threshold.

A plan is reapplicable to another P-state if all the reduction operators in the plan (that are not redundant) are able to be applied to the new P-state, and when applied, result in the goal state being achieved. That is, each reduction operator in the totally ordered plan is applied to the new P-state in order, and if all operators succeed and the goal state is reached, the plan has been successfully reapplied.

If a plan, during reapplication, fails due to the unsuccessful application of an operator, that plan is not entirely discarded. U-Plan may attempt to use the part of the plan that was successful and continue planning from the point where the plan failed. The intent is to construct plans with the same or similar strategies by reusing at least part of the plan at the highest level of abstraction.

The overall effect of reapplying existing plans to numerous P-states is to produce fewer plans than P-states. The reapplication of part of an existing plan results in a plan set that uses a smaller number of strategies than would be the case if a unique plan were produced for each P-state. The net effect of plan reapplication is a simplified plan merging process, resulting in a more workable super-plan (discussed in the next section).

1.2.5 Super-Plans

When planning given uncertain and incomplete information, the assumption (requirement) that additional information about the world can be acquired by knowledge acquisition activities may need to be made. The ability to plan to acquire additional information may alleviate some of the problems associated with the limitation of having only partial information. However, these activities are likely to incur a cost. It would be beneficial to find a plan that attains the desired goal by acquiring only necessary supplementary information. In addition, it would be desirable to acquire the supplementary information at the point in the plan when that information is needed to select among alternatives.

U-Plan intends to acquire knowledge at a logical point; that is, where information is required to continue operator execution. This is achieved by constructing a *super-plan* that combines the set of existing plans and knowledge acquisition operators. When a

plan exists for every possible world, the operator order of all the plans is combined to obtain a single planning tree that branches when the operator execution order differs. At this point the ability to acquire additional knowledge is used. At each branch, a knowledge acquisition operator can be inserted to determine which action in the planning tree to carry out next.

The circumstance may arise in which the required information is not available and an action must be selected, based on the support for each branch of the super-plan. Simply stated, the evidence supporting each plan is equal to the sum of the evidence for the initial P-states that use the plan; this includes the evidence attributed to the disjunction of any of the initial P-states in the set. The support for each branch of the super-plan is, therefore, equal to the sum of the evidence supporting the plans included in that branch.

1.3 U-Plan's Domain Of Application

Planning under uncertainty covers a broad range of issues involving generating a course of action for situations containing imprecision. U-Plan has resulted from the general intention to develop an approach to planning, where imprecise information about the world is likely to be all that is available at plan time. Additional information might be accessible before plan execution, and should be included as an option in the planner's final output. The specific domain types in mind were named by the author *emergency response* problems: that is, domains in which action would be required before complete and precise information could be obtained, but where a general course of action that achieved the goals was produced.

This research examines how one might plan in an imprecisely described environment by building on current planning theory, focusing on the type of output one would desire from the planner. When considering how an agent might act in the *emergency response* problems, the following attributes of a plan were identified.

- The number of plans should be fewer than the total number of possible worlds.
- A number of alternative plans are likely to be needed to cover the sets of possible worlds used to describe the environment. The final plan produced by U-Plan should include a method for choosing between these alternatives, or give a measure of the evidence attributed to the plan alternatives.
- Action sequences common to a group of alternative plans should be applied before the choice to select between alternative plans is required.
- To produce a set of plans that maximised the number of common action sequences, one should attempt to reapply all or part of some plans generated for other possible worlds.

To achieve these ends, the *emergency response* problems should include the circumstances in which a number of planning solutions were likely to exist for each possible world. The problem would then be to generate a plan for one possible world

that could be applied, in part or whole, to a number of other possible worlds. This will allow U-Plan's reapplication module to produce fewer plans than possible worlds, and maintain a number of common action sequences in the differing plans.

Domains where only one solution exists for each possible world can be addressed by U-Plan. However, the number of decision nodes in the final super-plan would reduce the usefulness of the plan, as the plan would acquire near complete knowledge to execute it. The time required to produce a super-plan under these conditions would be substantially higher than a traditional planner would take to produce one plan for every possible world.

The problem domain should not be prone to high levels of incomplete data (i.e. evidence being attributed to a set consisting of more than one element) or conflicting data, and when knowledge acquisition is doubtful or impossible. The reason for this is more an understanding of the uncertainty calculi used by U-Plan to represent imprecise information (i.e. Dempster-Shafer theory). When evidence is attributed to a set containing more than one element, the evidential interval produced is likely to demonstrate low support for or against the possible worlds. As the evidence gathered for the possible worlds is used to rank the plans generated by U-Plan, there may be no clear preferences between the plan alternatives. When additional knowledge acquisition is unlikely, the application of a super-plan produced by U-Plan depends on the information provided in the evidential intervals.

U-Plan has been applied to a simulated air combat domain (appendix 1) and hazard action response domain (Mansell 1995). In addition investigations have begun on the application of U-Plan to a submarine manoeuvre planning domain (described in appendix 2). The air combat domain (appendix 1) is used throughout this report as a demonstrator, and it is believed that many of the results reported here are directly transportable to the submarine planning domain.

2. State Representation

Classical planning systems assume the availability of complete information at the time of planning. Unfortunately, in real environments, an agent must often deal with incomplete information (due to, for example, the sensory limitations of knowledge sources). When formally representing a domain that is described using imprecise information, two approaches can be taken. One is to use possible worlds (i.e. an exhaustive set of unique world states are used to represent the agent's environment). Alternatively, the representation of the world can be ambiguous, allowing the disjunction of statements in the representation. The selection of a representation will both guide and limit the development of the entire planning mechanism. This section describes U-Plan's unique approach to representing the environment in which it operates.

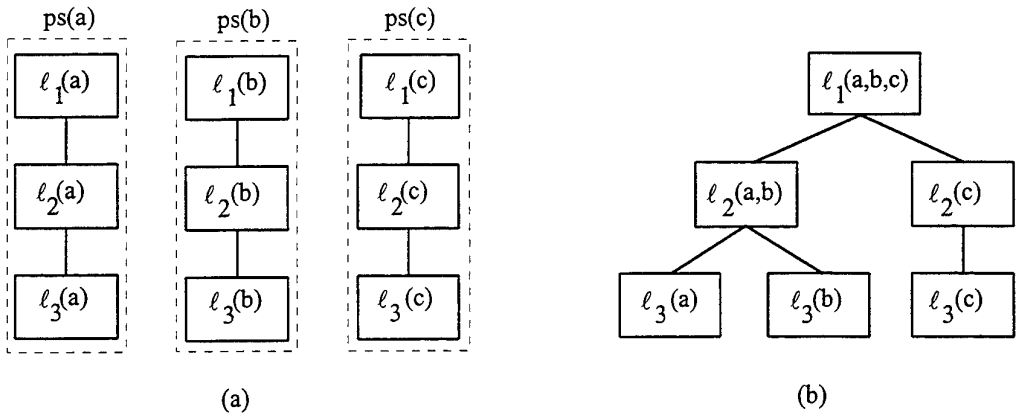

Figure 3: (a) Depicts how 3 sample P-states have representations at 3 abstraction levels. $\ell_n(x)$ represents P-state, x, at abstraction level, n. (b) An example of how 3 initial P-states may be grouped in tree form.

2.1 P-States

When an incomplete model of the world is all that is available, a set of initial states can be used to describe the alternative environments. U-Plan employs a set of initial possible states (P-states) to describe what might be true of the world. A P-state, ps(a), is a complete description of one possible world using propositional statements. Each P-state is described hierarchically with n levels of abstraction: (ps(a)=$\{\ell_1(a) \ldots \ell_n(a)\}$) where n is domain dependent and selected during knowledge engineering (fig. 3(a)). The level $\ell_i(a)$ is a complete description of a world at the ith level. The highest level of abstraction gives a coarse description of the state of the world. The lowest level gives a detailed view of the world. Intermediate levels provide the description required to make a smooth transition between both extremes.

Information sources provide U-Plan with a set of propositional statements that represent distinct aspects of the domain. Each propositional statement has associated with it a measure of certainty. (U-Plan uses a Dempster-Shafer mass distribution for reasons discussed in section 2.3.) The propositional statements are then mapped to the lowest level of abstraction, where they are used to generate a set of detailed P-states. (For example, in figure 3(a) the initial information is used to construct $\{\ell_3(a),\ell_3(b),\ell_3(c)\}$, the set of P-states described at the lowest level of abstraction.)

2.2 Compatibility Relations

Compatibility relations (Shafer, 1976; Lowrance *et al.*, 1991) are a part of Dempster-Shafer theory used to describe which elements from two frames can be true simultaneously, allowing propositional statements to be addressed jointly. U-Plan uses compatibility relations to specify the interrelationships between a piece of information

at one level of abstraction and other levels of abstraction. The restriction is that a compatibility relation can only link one level to the levels directly above or below it. To ensure that each level is a complete representation of the possible world, a compatibility relation must exist for every element of the frame.

U-Plan uses compatibility relations (defined when knowledge engineering the problem domain) to construct a representation of each P-state at every level of abstraction. This is a bottom-up process, beginning with the most detailed description, mapping to the intermediate descriptions, and concluding with the most abstract description. For example, the detailed description ℓ_3(a)={Alt(1-2), type(f1), ...}), may be used to produce ℓ_2(a)={Alt(v-low), type(fighter), ...}), which in turn is used to generate ℓ_1(a)={Alt(known), Intent(fighter-cover), ...}). The resulting P-state, ps(a), is represented by three descriptions of the environment at different abstraction levels, {ℓ_3(a),ℓ_2(a),ℓ_1(a)}.

The cost of using this possible worlds representation is the first time penalty of generating each abstracted P-state, and the space required to store the additional P-states. One benefit provided by this representation is the ability to write operators that use a description of the world equivalent to their level of abstraction (see section 1.2.2). Another benefit is that the operator descriptions are simplified, as abstracted propositions can be used to represent numerous detailed propositions. For example, an intermediate reduction operator that is conditional on the aggressor being a fighter will use the proposition type (fighter), instead of listing all the fighter aircraft.

2.3 P-State Grouping

U-Plan groups together equivalent initial P-states according to their hierarchical levels; i.e. the P-states with the same state description at a particular abstraction level are grouped together.

Figure 3(b) demonstrates how initial P-states may be grouped in tree form. In this example the set of P-states from figure 3(a) are used. At the lowest level of abstraction the set of possible worlds are distinct, represented as the leaf nodes of the tree, {ℓ_3(a),ℓ_3(b),ℓ_3(c)}. Let us assume, when viewing the world in a more coarse light, i.e. at a higher level of abstraction, ℓ_2(a) and ℓ_2(b) are identical. In this case they would be grouped together to give ℓ_2(a,b). At the highest level of abstraction ℓ_1(a,b) and ℓ_1(c) might also be identical resulting in the state ℓ_1(a,b,c).

2.4 P-State Ranking

Information acquired in a real-world situation provides evidence about the possible states of the world. This information is typically uncertain and incomplete. Dempster-Shafer (D-S) Theory (Shafer, 1976; Lowrance *et al.*, 1991.) is one way of handling such evidence, using an interval to explicitly capture what is known as well as what is not

known (i.e. uncertainty). This is achieved by allowing belief to be associated directly with a disjunction of events. The D-S theory was chosen as it is well suited to dealing with information represented at different levels of abstraction (through compatibility relations).

Dempster-Shafer reasoning is used to assess the effect of all pieces of available evidence on a hypothesis, making use of domain-specific knowledge. Fundamental to D-S theory is the frame of discernment (or frame), Θ_A, the set of mutually exclusive, exhaustive propositional statements representing what is known of the world. For example, frame A might represent the various headings a target aircraft may take. Propositional statements are represented by disjunctions of elements of the frame (Θ_A). Additional aspects of the domain can be incorporated into the system by the inclusion of new frames of discernment.

U-Plan represents a body of evidence about the environment as a set of propositional statements within a frame of discernment. Assigned to all possible propositional statements in a frame of discernment is a belief value (summing to unity); these values, $m_A(A_i)$, are known as *masses*, and the process is called a *mass distribution*.

To interpret a body of evidence relative to a propositional statement A_j, the *support* and *plausibility* are calculated to derive what is termed the *evidential interval*. Stated simply, the *support* for a hypothesis A_j is the sum of the masses of all propositions that are subsets of A_j (including A_j itself), while the *plausibility*, Pls(A_j), is the degree to which the evidence fails to support its negation. The difference between support and plausibility represents the residual ignorance, or uncertainty, $U_\theta(A_j)$=Pls(A_j)-Spt(A_j).

U-Plan calculates a measure of support and plausibility for each initial P-state at every level of abstraction based on the mass distributions of the initial evidence. This is used to determine the order in which possible worlds have plans generated.

The methodology, provided by D-S theory, for handling the interrelationships between propositions in different frames of discernment gives U-Plan the ability to handle state spaces at multiple levels of abstraction (Shafer, 1976). When sufficient evidence is available for the mass functions to represent Bayesian probabilities,[2] the evidential interval calculated for a P-state is identical to the updated Bayesian probability. (An implementation of U-Plan that uses Bayesian probabilities to characterise the environment is described in Mansell, 1994c.)

[2]A Dempster-Shafer mass distribution is identical to a Bayesian probability distribution when complete and precise probabilities exist for all relevant propositions (i.e. when there is no residual uncertainty).

2.5 P-state Selection

The selection of the initial P-state to begin planning involves choosing the P-state with greatest support[3] at the highest level of abstraction (for example ℓ_1(a,b,c)). The node in the P-state tree that is a child of this initial P-state with the greatest support is then selected (e.g. ℓ_2(a,b) or ℓ_2(c)). This selection process continues from highest to lowest levels of abstraction. The result is an initial P-state with a description at all levels of abstraction.

The P-states are chosen in this manner in an attempt to allow the possible world with the greatest support to be planned first. This does not guarantee the plan will have the greatest support when planning is complete, or that the best plan will be constructed first. The usefulness of this strategy becomes apparent in section 6 when attempting to use all, or part of, previously constructed plans during planning for other P-states. The effectiveness of this approach relies on a suitable representation of the domain and the reduction operators.

3. Reduction Operator

Planning operators represent actions that the system may perform in the given domain. The role of an action is to change the state of the world; the aim of an operator is to represent how applying that action will change the system's view of the state of the world. U-Plan uses reduction operators to give alternative methods for achieving the goal at a lower level of abstraction, or at the tactical level it describes the direct effects of an action on the P-state. These are SIPE-like operators (Wilkins, 1988), used where hierarchical planning takes place in what is assumed to be a closed world.[4]

---

[3]The selection of the initial P-state is based on the selection of the best Dempster-Shafer interval. A variety of techniques dealing with interval based decision making exists; they are currently under evaluation.

[4]The world is assumed to be closed only for plan generation. An environmental monitoring module is constantly ensuring the world does not change significantly from the description held by U-Plan.

Name:	Attack
Level:	1
N-Precond:	Weapons(¬nil) AND Fuel(>1000)
S-Precond:	TargetLocation(known)
Plot:	(OR (BVR_Attack (1.0)) (VR_Attack (0.8)))
Probability:	(((Intent$_1$(Air-super))) 0.70) ((Intent$_1$(Fighter-cover)) 0.80) ((Intent$_1$(Bomb)) 0.95))
Postconditions:	Nil
Planfail:	Backtrack

Name:	Acquire_Target
Level:	2
N-Precond:	
S-Precond:	TargetLocation(known)
Plot:	(OR (Radar_Lock (1.0)) (Visual_Lock (1.0)))
Probability:	(((Type$_2$ (Fighter)) 0.80) ((Type$_2$ (Fighter-Bomb)) 0.85) ((Type$_2$ (Bomb)) 0.95))
Postconditions:	Nil
Planfail:	((Fail Attack) (Backtrack))

Name:	Radar_Lock
Level:	3
N-Precond:	Target(located)
S-Precond:	Radar(Tracking) OR Radar(on)
Plot:	(Radar(?) $\Rightarrow$ Radar(lock))
Probability:	(((Alt$_3$ (0 1)) 0.60) ((Alt$_3$ (2 3)) (Type$_3$ f1)) 0.70) ((Alt$_3$ (2 3)) (Type$_3$ fb1) 0.75) ((Alt$_3$ (2 3)) (Type$_3$ b1) 0.80) ((Type$_3$ f1) 0.80) ((Default 0.90))
Postconditions:	Radar(lock)
Planfail:	((Fail BVR_Attack) (Backtrack))

Figure 4: A simplified example of the Attack, Acquire_Target, and Radar_Lock operators from the air combat domain.

In addition to effects, operators hold information about the objects involved in the action, their relationship to each other and the domain, the goal the action is intended to achieve, the conditions necessary for the action to be performed, and the likelihood of the operators' success. The way in which operators are described will be presented with the aid of three simplified sample operators given in figure 4. The *Attack* operator is of a fairly high abstraction level, describing when and how to attack an aggressor aircraft; the *Acquire_Target* is a medium level operator; and the *Radar_Lock* operator works at the lowest level of abstraction, physically changing the state of the world (as shown in figures 13 and 14). What follows is an explanation of what is defined for the typical reduction operator.

Abstraction Level

The P-states used by U-Plan are described at a number of abstraction levels, one of which corresponds to the level the operators are designed to work upon. Reduction operators alter the P-state according to how an action is expected to change the state of the world. To synchronise the level of abstraction at which the active P-state should be modified, each operator designates the level of abstraction on which it operates. Any changes that are made at one level are instantly upgraded for the entire P-state to maintain consistency. For example, the *Attack* operator is designated to base its selection and act on the P-state's most abstract description, while the *Radar_Lock* operator will alter the P-state at abstraction level 3.

Necessary Preconditions

An operator's necessary precondition must be true in the P-state before the operator can be applied. The system will make no attempt to make these preconditions true; a false precondition simply means the operator is inappropriate at this time. Necessary preconditions are useful in domains where one does not wish to allocate resources to change the world to allow an action to be performed outside the prevailing conditions. That is, it may or may not be possible to find a series of actions that will allow the operator to be performed, but one would not wish to work to achieve this. In the *Attack* example, it is deemed worthless to attack another aircraft if you have insufficient fuel or no weapons.

The situation may also arise in which one wants to base the action on what is true of the P-state at the present time, without changing the state of the world. For example, a *BVR-Attack* should only be attempted if the aggressor is already beyond visual range. If the defender was in visual range when testing the *BVR-Attack* preconditions, the defender would not want to try to attain beyond visual range status so as to continue with the *BVR-Attack*.

Satisfiable Preconditions

Satisfiable preconditions represent conditions that must be true of the world before the operator can be applied. U-Plan will work to alter the P-state so that the operator can

be applied. This involves finding an operator (or sequence of operators) that, when executed, produce a P-state which satisfies these preconditions. Initially, the postconditions for the set of all operators are searched to determine if any one operator makes one or more of the necessary changes to the P-state. If the sequence of actions produced by this process does not achieve all the changes required, a random walk algorithm is used to search the set of operators to make the final alterations to the P-state. The criteria used to decide when to discontinue attempting to find a solution is domain dependent, and should be entered at knowledge engineering time. In the *Radar_Lock* operator case the radar must be on, or in tracking mode, before the operator can be applied.

If an operator or sequence of operators are found to be capable of transforming the P-state in the desired manner, they are applied and the relevant changes to the P-state made. The operators are included in the plan, and if the operators have a probability less than 1, the probability of the parent operator is updated as discussed in section 4.4. For example, the *Set-Alt* operator includes the satisfiable preconditions $Vel_3(\neg\ low)$, meaning the defender aircraft may not have low velocity when changing altitude. If the defender does have a low velocity, a *Set-Velocity(med)* operator can be applied to increase the velocity.

Plot

The plot provides step-by-step instructions on how to perform the action represented by the operator. This includes a description of the goal reducing operators that are applied at the next level of abstraction, and the degree to which they achieve the goal of the parent operator (i.e. fulfilment). Or, at the lowest level of abstraction, how applying the operator changes the P-state. The plot can be described as nodes in a procedural network: OR nodes, demonstrating that one of a given set of operators must be applied; and AND nodes, requiring that a sequence of operators be applied.
The *Attack* operator may be achieved by either a Beyond Visual Range (BVR) Attack, or a Visual Range (VR) Attack. The *BVR_Attack* completely fulfils the goal of the *Attack* operator, where the *VR_Attack* is expected to only partially (80%) fulfil this goal. The *Radar_Lock* operator changes the radar predicate in the active P-state to locked on to the target (regardless of its prior value).

Probability

Each operator includes a function for calculating the probability of the reduction operator being successfully applied to the current P-state, given the description of the world held in the P-state. In this report the term *successful* when referring to the application of an operator should be interpreted as the condition arising when the reduction operator is applied to the P-state (or expanded as outlined in definition 3), directly resulting in the goals of that operator being achieved, the method for achieving these goals being outlined in the plot of the reduction operator. Alternative definitions of successful application (e.g. definitions that allow accidental or

coincidental goal achievement) are not used as they complicate the process of obtaining conditional probabilities.

When the action to be applied is a reduction operator, the probability indicates the likelihood of that reduction operator successfully accomplishing its intended goal. Alternatively, one could consider the probability function as illustrating the likelihood that the course of action outlined in the plot of the reduction operator can be successfully implemented.[5]

The meaning of an operator's probability of success depends on the abstraction of the operator. At the lowest level of abstraction, operators physically change the P-state; hence probability is a measure of the likelihood that the world will change as intended. At higher levels of abstraction, probability is a measure of confidence in the ability of the agent (in this case, the pilot) to successfully select and execute a sequence of lower-level actions.

For example, figure 4 describes the *Attack* reduction operator and *Radar_Lock* operator. The probability of the defender successfully performing any of the attack strategies is calculated, knowing that an attack could involve either a *BVR_Attack* or a *VR_Attack*. Alternatively, the probability of achieving a *Radar_Lock* on an aggressor aircraft is a measure of the likelihood that the defender aircraft will achieve radar-lock given the aggressor aircraft type and location.

The definition of the probability for a reduction operator is formalised here:

Definition 1 (probability of a reduction operator): The probability of reduction operator, *A*, is defined as the likelihood of that operator's successful application in a particular P-state, i.e. $p(A \mid P\text{-}state)$. This probability is conditional on the information contained in the P-state representing the prevailing description of the world. These conditional probabilities are formulated under the assumption that the environment is suitable to the application of that reduction operator (as outlined in the preconditions). If the environment is not suited to the specified reduction operator, the probability of its success is nil. From this point forward, the probability of an action, *A*, given the state of the world, i.e. $p(A \mid P\text{-}state)$, will be referred to as the probability of action *A*, and represented by $p(A)$. (The conditionality is inferred.)

Definition 1 indicates that the likelihood of successfully applying the reduction operator depends on the type of environment in which it is used. The probability information is given in the form of a limited set of conditional probabilities. The limitation comes from these factors:

[5]Assuming the reduction operator has been accurately constructed, and the course of action outlined in the plot achieves the goals of the operator.

- Those descriptions of the world deemed impossible by the preconditions of an operator are not considered in the calculation of probability. For example, the *Fire_Weapon* reduction operator requires (in its preconditions) that a weapon be armed. Hence, conditional probability $p(Fire\_Weapon(x) \mid \neg Weapon\_Armed(x))$ would not need to be obtained.
- The P-state is described by a set of propositions. However, not all these propositions may be considered relevant to the likely successful application of the reduction operator. Hence, the set of conditionals can be reduced to considering only relevant information. For example, the velocity of the aggressor aircraft does not affect the likelihood of successfully achieving an *Arm_Weapon* reduction operator.

The above constraint on the type of information required to be included in the conditionality statements reduces the complexity of acquiring such information. As demonstrated in figure 4, the conditional probabilities included in the allocated slot of the reduction operator are arranged in such a way as to minimise the search time and representational space. U-Plan will search through the list of propositional statements that describe what must be true of the environment, selecting the first set of propositions that match the current P-state. The idea, therefore, is to include the most detailed information earlier in the listing; the more general (catch all) statements being left until later. For example, in the *Attack* reduction operator case, the probability of the defender successfully performing any attack depends solely on the intent of the aircraft it is engaging. However, in the *Radar_Lock* operator example, the probability of success is lowest when the aggressor has an $Alt_3(0\ 1)$ altitude. This probability increases (depending on the type of aircraft) when the altitude is $Alt_3(2\ 3)$, and is highest when the previously mentioned cases are not indicative of the environment.

The availability of this probability information is domain specific and may be difficult to obtain. In the air combat domain the probabilities are obtained empirically (based on historical data and expert opinion).

The probability of success alone does not provide sufficient information to select a reduction operator, as it does not take into account the goals of the system. It is for this reason that, associated with each reduction operator (listed in the plot of a parent reduction operator[6]), is a measure of fulfilment, representing the degree to which the reduction operator achieves the goal of the parent (see section 4.1).

Postconditions

Postconditions outline what the operator directly intends to achieve by its application. This does not include any side effects that may be exposed by the deductive causal theory (section 3.1). The postconditions are particularly useful when searching for an

[6] The parent reduction operator is one level higher in the abstraction that the current reduction operator, lists the current reduction operator in its plot, and provides the goal to be achieved by the current reduction operator.

operator that achieves a satisfiable precondition. In the Radar_Lock operator example, the postcondition is achieving radar lock on the aggressor aircraft. The Attack and Acquire_Target operators do not directly alter a P-state, and therefore have empty postconditions.

Planfail

The planfail slot outlines what to do if the operator fails during planning. In most cases, the instruction will be to backtrack (as is the Attack operator's planfail directive). However, the occasion may arise when the failure of an operator has broader consequences, such as ruling out options further up the abstraction hierarchy. This information is domain specific and often difficult to characterise. If a Radar_Lock cannot be achieved, then the planfail tells U-Plan no BVR_Attack manoeuvres will be successful as they all require the defender aircraft's radar to lock onto the aggressor (due to the type of long-range missiles used by the defender). If the strategy being performed is a beyond-visual-range attack, U-Plan marks BVR_Attack operator as failed and implements its planfail. If the strategy being performed is not a BVR_Attack, then a default backtrack is applied. Similarly, if both Radar_Lock and Visual_Lock cannot be achieved, no Attack strategy can be successful, and U-Plan will be required to find an acceptable Turn_Away manoeuvre.

3.1 Causal Theory

U-Plan uses a deductive causal theory (Wilkins, 1988; Pednault, 1988) to deduce the context dependent effects of applying a reduction operator to a P-state. Operators explicitly list effects that are likely to change the state of the world. After the application of each reduction operator, a set of triggers are used to determine if the world has been changed in such a way that the deductive rules need to be applied. If so, the deductive causal theory is used to change the P-state to be consistent with all the effects of an action. The effects of applying any reduction operator are recorded in the abstraction hierarchy.

The effects that are deduced are considered to be side effects, where those that are introduced directly by the reduction operator are the direct effects. The use of deduced effects simplifies the description of the operators by removing the need for extensive add and delete lists.

The causal rules consist of four slots. The name of the causal rule is held in the Causal Rule slot. The Trigger contains the propositional statement altered by a reduction operator that triggers the causal rule to be applied. The Precondition contains any additional propositions that must be true of the world for the causal rule to take place. Finally, the Effect lists the propositions that should be true in the P-state. These Effects are documented as an indirect (or domain specific) effect of the action.

Causal Rule:	Deduce_Tracking_Status_vr
Trigger:	Radar_Mode(Locked)
Precondition:	($Range_2$ vr)
Effect:	Tracking_Status(Identified)

Causal Rule:	Deduce_Tracking_Status_bvr
Trigger:	Radar_Mode(Locked)
Precondition:	($Range_2$ bvr)
Effect:	Tracking_Status(Located)

Figure 5: The causal rules used to deduce Tracking_Status.

For example, figure 5 shows an example of the two causal rules used to deduce any change in Tracking_Status proposition (or information about the defender held by the aggressor). Both rules are triggered by achieving a radar lock (i.e. Radar_Mode(Locked)) on the aggressor. The first causal rule alters the Tracking_Status to Identified if the two aircraft are in visual range. The second alters the Tracking_Status to Located if the two aircraft are beyond visual range.

Two problems may arise when using deductive causal rules (Wilkins, 1988). The first is how to respond when two deductive rules clash by attempting to alter the same predicate to different values. This situation usually arises due to an error in the deductive database. Hence the normal response by U-Plan is to report the clash of the two rules and implement the first rule (ignoring the second).

The second problem identified by Wilkins (1988) is potentially more serious. This arises in domains where planning variables are used. It may be necessary to instantiate plan variables to allow the application of the causal rules, thereby possibly missing a planning solution. This is not a common problem in the typical U-Plan domains that are more structured, and in which the use of planning variables are less likely to be necessary. However, in case the problem arises, U-Plan does not allow the instantiation of a plan variable by the deductive causal rules. The causal rule automatically fails when it requires information held as a plan variable. The assumption is that, when the plan variable is instantiated, the causal rule will be applied.

4. Single Plan Generation

Many classical planning systems use a state-based search strategy to solve planning problems. To find a solution one applies operators to a state description until an expression describing the goal state is found. U-Plan does not construct a state-based search tree, but constructs a strategy hierarchy (also used by Bonissone *et al.*, 1994) which is a decision tree like structure, where the nodes in the hierarchy represent a continuous transition of actions from the strategic (at the root node) to the tactical (at the leaf nodes). The nodes closest to the root node are highest in strategic intent, representing not only a decision at a high level of abstraction, but the direction the plan will take. The nodes closest to the leaf nodes have maximum detail, representing task and action sequences.

The strategy hierarchy can be represented as an AND/OR search tree: the root node representing the strategic goal of the system, and the leaf nodes representing the tactical details of how the goal is to be achieved. Each node in the tree is a subgoal node representing the current goal and a description of the P-state being planned for. Certain pairs of nodes in the abstraction hierarchy are connected by arcs, representing the application of a reduction operator that produces the subsequent subgoal node. For example, figure 13 shows part of the strategic portion of the strategy hierarchy for the air combat domain, where the goal is to *Defend_Assets* by engaging an aggressor. The strategy hierarchy reveals the high level strategies available to the planner, and the manoeuvres that achieve these goals. A typical example of a *BVR_Attack* manoeuvre (the *Cutoff_Intercept*), as given in figure 14, demonstrates the tactical detail of implementing such an attack.

The order in which reduction operators are selected during planning is based on a calculation of expected fulfilment (section 4.1). The calculation is dependent on the reduction operator and the P-state in which the operator will be applied. In figure 13 the expected fulfilment appears in brackets above each subgoal node. Each single subgoal node represents part of a plan to achieve the subgoal at the next highest level of abstraction.

The following sections outline how the reduction operators are selected and implemented, and the process that is continually reviewing these decisions.

4.1 Calculating Expected Fulfilment

Expected fulfilment is a quantitative measure used to rank the reduction operators that achieve the goals of the active operator (i.e. the next reduction operator chosen to be expanded). For example, if *Attack* is the active operator (whose goals we wish to achieve), then the expected fulfilment for a *BVR Attack* and *VR Attack* (the operators that achieve *Attack*) are calculated and used as a basis for the selection of the operator that should be attempted first.

Probability theory provides an effective method for choosing actions capable of producing consistently accurate choices. What probability offers is the ability to capture what is in essence an abstraction of human judgement, through the careful manipulation of observation. Whereas probability is used to represent the likelihood of an event, utility is used to represent the desirability of the consequent of an action.

The operation of utility theory is well understood (von Neumann and Morgenstern, 1947; Pearl, 1988). The actions available to a rational agent at any one time are characterised as to their applicability and consequence. The selection of an action amounts to maximising the probability and desirability of the consequent of the action. If for each configuration c for the set of consequents C, we assign a utility measure $U(c)$, representing the degree of desirability, then the overall expected desirability associated with action a is given by:

$$EU(a) = \sum_{c} U(c)P(c|a,e), \qquad [1]$$

where, $P(c \mid a,e)$ is the probability distribution of the consequence configuration c, conditioned upon selecting action a and observing evidence e (contained in the P-state).

In this report fulfilments are used as a variation on utility theory whereby fulfilments are used as a local measure of the degree to which the consequent of the action achieves the intended goal and the desirability of that action. The term 'fulfilment' is used to capture the essence of utility scaled according to the desire to use a particular approach. The term 'utility' is not used in this description, as it is introduced by von Neumann and Morgenstern (1947) to capture the more general concepts of desirability, cost, risk etc. To avoid confusion with this general expression, the term 'fulfilment' is used to focus attention on a measure of the degree to which specified goals are achieved.

Definition 2 (fulfilment of a reduction operator): Fulfilment is defined as the desirability of applying a specific reduction operator in order to achieve the goals of the parent operator. This may include the degree to which the operator achieves the specified goals, or the estimated cost or risk involved in selecting a particular strategy. □

Fulfilment values for a child operator are given in the plot of the parent. When the plot gives an OR option, it reveals that the goal of the operator can be achieved by any one of the given child operators. The fulfilment value associated with each child operator is a quantitative measure of the benefit of achieving the parent's goal in this particular way. At least one of the children should have the same fulfilment value as the parent. If none of the children could offer a fulfilment equal to that of the parent operator, there would be no justification for the original fulfilment of the parent operator. Such an assignment is based on the optimistic allocation of fulfilments, implying that if the best possible tactics outlined by this operator were available, the outcome will achieve

the goals with the given degree of fulfilment. (Section 4.4 describes how U-Plan compensates for assigning optimistic fulfilment values.)

When the plot of an operator is an AND condition, a list of operators are given that, when applied to the P-state, achieve the goal of the parent. In such cases, there are no alternative courses of action made available to the planner. For this reason, the fulfilment of each child operator in the plot is given the same fulfilment value as the parent. The goal of the parent is wholly achieved by the set of children and, if any one of the child operators fail, then the parent also fails.

The expected fulfilment is used as a measure of an action's likelihood to produce the consequent that achieves the agent's goals. If we use the measure *F(c)* to represent the degree of fulfilment of consequent, *c*, then the overall expected fulfilment associated with action *a* is given by:

$$EF(a) = F(c)P(c|a, ps), \qquad [2]$$

where $P(c \mid a,ps)$ is the probability of achieving consequent *c*, conditioned upon selecting action *a* and the current P-state *ps*.

For example, to calculate the expected fulfilment of the *Attack* operator in figure 4, the probability[7] of successfully executing an attack in the given P-state is multiplied by the degree of fulfilment[8] obtained by executing the action.

The expected fulfilment of action *a*, *EF(a)*, is regarded as a gauge of the merit of action *a*. The expected fulfilment is used as a procedure for choosing among alternative (or competing) actions. When given the choice between two actions (e.g. *Attack* and *Turn_Away*) the selection is based on the action that yields the highest expected fulfilment (i.e., *EF(Attack)* or *EF(Turn_Away)*). This result will depend on the description of the P-state when the selection is made. This process can be thought of as establishing a rank order in which one should attempt to apply these operators.

4.2 Foldforward Analysis in the Abstraction Hierarchy

Traditionally, utility theory has been used to analyse decision problems by organising elements of the problem into a decision tree (described in Pearl, 1988). Two techniques exist for generating a plan using a decision tree. The first (called a *foldback analysis*) involves comprehensively expanding the problem to facilitate a search for the optimal course of action, starting from the leaf nodes and working back towards the root. Alternatively, a *foldforward analysis* can be used. Such an analysis requires a detailed

[7]The probability for the Attack operator is obtained from the operator description and represents the likelihood that an Attack applied in the current P-state would succeed.

[8]Similarly, the fulfilment is obtained from the parent of the *Attack* operator, and represents the degree to which an *Attack* achieves the higher level goal of *Defend_Assets*.

description of what information is available and what will be available in the future, a complete list of actions, precise details about their effects on the world, their utility, etc. The implementation of a *foldforward analysis* is usually thought to be too unruly for practical use (Pearl, 1988).

Variations of the foldforward and foldback analyses using EF values were considered for use in the abstraction hierarchy. The foldback analysis requires the calculation of EF values by propagating probability and fulfilment values from the bottom up, in an abstraction hierarchy generated through the exhaustive combination of operators. The foldforward analysis would require an algorithm for calculating EF using the partially evolved abstraction hierarchy. U-Plan does not use a foldback analysis, as it requires all possible combinations of operators to be evaluated in the abstraction hierarchy in order to obtain the EF values.

The intractability of foldforward analysis (Pearl, 1988) comes from the need to calculate the probability and fulfilment of a high level action, without the benefit of committing to a single plan strategy or tactic. The result is that, at each forward step, one must calculate the probability (and fulfilment) that at least one of the possible combinations of operators will successfully achieve the current goal. For example, in this classical foldforward analysis, to obtain the EF of the Attack operator (see fig. 13) one must evaluate the probability and fulfilment values for all the operators that constitute every beyond-visual-range and within-visual-range attack.

U-Plan uses a variation on the foldforward approach by which both actions and information are represented at the level of abstraction equivalent to the decision being made. In the abstraction hierarchy, the calculation of probability and fulfilment for the *Attack* operator is based on the information at the same level of abstraction (not on the set of operators at the following levels of abstraction). The decision to *Attack* is based on the historical data available on the consequence of attacking an aggressor. However, this measure of the expected fulfilment of *Attack* is only an estimate of the performance of the *Attack* action (based on previous history). As the subsequent lower level details of that strategy are chosen (i.e. a specific attack type is selected) the expected fulfilment of the *Attack* operator will be updated to incorporate the additional detail. (This process is discussed in section 4.4.) The update of the expected fulfilment value primarily occurs to check that the measure used to make the earlier decision has not changed substantially. For example, the expected fulfilment of the *Attack* operator will be updated as detail about the attack strategy is added, and the original decision in favour of the *Turn_Away* operator can be re-evaluated (see section 4.4.)

U-Plan is not generating an optimal plan for every possible world. This approach is designed to guide the selection of operators, with the intention of producing a useful plan in a reasonable amount of time. However, plan generation is not carried out for all possible worlds, as U-Plan reuses plans on P-states other than the ones for which they were generated. Under these circumstances, an optimality measure would have to consider not only the EF of the plans generated, but also the number of the plans generated. A study of the optimality of the plans generated by U-Plan is not carried

out, as there are many interpretations of what is optimal, and the selection of any one is highly domain dependent.

In planning problems, one desires to choose a number of actions from a set of possible actions that, when applied to a world state, achieve specified goals. In general, it is too time-consuming to run a complete analysis of all possible combinations of action orderings. Hence, some form of pre-processing (e.g. an abstraction hierarchy) and intelligent search is carried out to determine the order in which operators are applied. As discussed in this section, U-Plan uses an expected fulfilment calculation in an abstraction hierarchy to determine the order in which operators are tested. The problem of selecting an action from a list of possible actions is well covered in the search theory literature, (Nilsson, 1980; Pearl, 1984; Korf 1987; Pearl, 1988).

4.3 Applying an Operator

Once the reduction operators that achieve the goals of the parent operator have been ranked using the expected fulfilment calculation, they can be tested to determine their suitability to the P-state. The successive application of reduction operators (from highest to lowest EF) to the given P-state then takes place until a suitable reduction operator is found. If the necessary preconditions of a reduction operator are true in the active P-state, then the reduction operator is provisionally selected and the satisfiable preconditions are tested. If any of the satisfiable preconditions are not true, U-Plan can attempt to satisfy them using reduction operators of equal or lower abstraction. If the necessary preconditions are not met, or the satisfiable preconditions can not be achieved, the operator is rejected and its planfail procedure is implemented.

Once both sets of reduction operator preconditions can be shown to be true in the active P-state, the operator is accepted and its plot can be applied. The plot represents the effects the reduction operator has on the state of the world, and the subgoals that may be used to achieve this goal. When applying the plot, the next level of the strategy hierarchy is exposed, and again the operators that achieve these goals are ranked using expected fulfilment. The plot of actions at the lowest level of abstraction specify how the P-state is physically changed by their application. For example, the *Attack* operator (fig. 4) can be applied if the aircraft has weapons, sufficient fuel, and the target location is known. The goal of the *Attack* operator can be achieved by either the *BVR_Attack* or *VR_Attack* operators, and the order in which they are tried depends on their calculated EF values.

To aid the explanation of how operators are selected and used, the terms *expansion of a reduction operator, OR node expansion,* and *AND node expansion* are defined. First, the expression *expansion* is used in relation to applying a reduction to a particular P-state, and is defined as follows.

Definition 3 (expansion of a reduction operator): A reduction operator has been expanded when the following conditions have been met:

- information pertaining to the ranking of the reduction operator has been extracted (i.e. the expected fulfilment has been calculated and utilised by U-Plan to determine the order in which actions should be investigated)

- conditions required of the environment by the reduction operator in its necessary and satisfiable preconditions are true of the P-state

- the effect the reduction operator has on the world as outlined in the plot are carried out, or the actions that achieve the goals of the reduction operator have been provisionally added to the planning hierarchy.

□

A reduction operator has been *expanded* when it has been successfully applied to a P-state. Two types of expansions may occur. When the reduction operator has been applied as part of an OR operation (as described in the plot of the parent operator), the result is an *OR node expansion.*

Definition 4 (OR node expansion): An expansion of a reduction operator is an OR node expansion when the plot of the parent operator defines the method of accomplishing its goals by the application of one reduction operator from a set of more than one reduction operators. This is represented syntactically in the plot using the OR notation, although in real terms this is an exclusive OR (XOR) operation, as only one of the operations will be performed.

□

Alternatively a reduction operator, when applied to a P-state, can achieve its goals by the ANDed application of a set of lower level reduction operators (as described in the plot of the parent operator) resulting in an *AND node expansion* (defined as follows).

Definition 5 (AND node expansion): An expanded reduction operator is an AND node expansion when the plot of the parent operator defines the method of accomplishing its goals by the application of a set of reduction operators.

□

To generate a layer in the abstraction hierarchy, U-Plan will expand operators in the order outlined by the EF calculation until the next layer of the strategy hierarchy has been exposed. (For example, in figures 13 and 14 *Attack* and *Turn_Away* are at one layer, while *Set_Altitude, Obtain_Separation, ...* and, *Fire_When_Ready* make up another layer in the abstraction hierarchy). At this point, the earlier selection of specific actions are reviewed (as described in the following section).

4.4 Reviewing Selected Operators

When constructing a strategy hierarchy, it is possible that as a plan's detail is filled out it becomes less likely to succeed. One reason for this is that the initial strategic

decisions are based on information at a more coarse level of abstraction. As the plan is expanded and more tactical decisions about the implementation of specific strategies are made, the expected fulfilment of specific plan branches may decrease. The other reason this occurs is that, when the expected fulfilment is calculated for an operator, the fulfilment component is an optimistic assessment of that action's ability to achieve the goal, (i.e. the fulfilment values of parent operators are based on the best child's fulfilment values). However, as planning continues it is likely, at a lower level of abstraction, that the best actions (i.e. actions with fulfilment values lower than its parent's) may not be applied. This should result in a reduction of the expected fulfilment for that plan branch.

This makes it important to review earlier decisions while planning. After the application of a group of reduction operators, U-Plan compares the expected fulfilment of the current subgoals with those of previous subgoals, and determines if they fall below the previous values plus an offset. Including an offset is an iterative deepening strategy.[9] The offset value will depend on the difference in abstraction level of the subgoals. It is expected that, as the system uses lower level information, the expected fulfilment of the plan will decrease. This offset value helps avoid the problem of the system's jumping around from branch to branch in the strategy hierarchy.

Previous operator selections are reviewed at certain stages of the evolution of a plan to ensure the decision to take a certain planning direction is still favourable. In the air combat domain this means that, as planning continues down to lowest level, the decision to use an *Attack* instead of a *Turn_Away* strategy is being constantly reviewed. To review these selections, one must have a way of *updating* the EF values calculated for these nodes in the abstraction hierarchy, based on the detail added to that nodes planning branch. A set of update rules are used to re-evaluate the fulfilment and probability of each operator expanded, given the most recent planning developments. The update rules used depend on whether the expansion is an AND node expansion or an OR node expansion. A detailed explanation of these update rules and how this quantitative search technique relates to existing techniques is contained in Mansell and Smith (1994d), and Mansell (1994c).

In the case where an operator is expanded and produces an OR branch in the abstraction hierarchy, the update rules used to determine the fulfilment and probability of a *parent* node given a set of possible *children* are given by:

$$F(parent) = \{F(child) \Big|_{children}^{MAX} EF(child)\} \quad \text{and} \qquad [3]$$

[9] A number of iterative deepening strategies exist that can be applied to this problem of selecting a suitable offset between different abstraction levels (Mansell 1995).

$$P(parent) = \{P(child) \Big|_{children}^{MAX} EF(child)\} \quad [4]$$

where *children* is the set of children operators that have been expanded out and may be successfully applied to the current P-state. Simply stated, rules 3 and 4 tell us that the fulfilment and probability values for a parent of an OR node is equal to the fulfilment and probability of the child node with the greatest expected fulfilment that has not been ruled inapplicable to the P-state (i.e. due to failure of preconditions during expansion).

The updating of the parent reduction operator at an AND node involves updating the probability and fulfilment as follows:

$$F(parent) = \{F(child) \Big|_{children}^{MIN} F(child)\} \quad \text{and} \quad [5]$$

$$P(parent) = \prod_{children} P(child) \quad [6]$$

In the AND case, the rules and their justifications are less obvious. The fulfilment of a parent operator is replaced by the child from the set of children with the lowest fulfilment. Normally, when confronted by an AND node, the children are given the same fulfilment as that of the parent. The justification is that, as the parent can only be achieved by the specified sequence of actions, then this reduction is simply a refinement of the parent operator (i.e. these actions should wholly achieve the desired goal). However, the situation may arise when, at a lower level in the abstraction hierarchy, the fulfilment value for one of the child operators may itself be updated (by its subsequent descendants) to a new, lower value. When such a circumstance arises, the child operator is not fulfilled completely by its descendant and, consequently, the parent operator is no longer completely fulfilled by its children. The AND node rules for updating fulfilment take such eventualities into account.

In the subset of the air combat example given in figure 6, a simple scenario for a *Close_In* manoeuvre operator is evaluated. The expected fulfilment for this operator is calculated (the first number in the square brackets above the operator, i.e. 850) based on the fulfilment and probability values (shown in the braces below the operator, i.e. 1000 and 0.85 respectively) obtained from the operator. On expanding the *Close_In* operator, the next level of the plan is uncovered. This shows that the *Set_Bearing, Acquire_Target* and *Fire_Ready* operator are to be applied. The fulfilments and probabilities for these are calculated and shown in braces below the operators. As this is an AND operation, update rules 5 and 6 are used to update the fulfilments and probabilities for the parent, *Close_In*, operator (shown in the second set of braces below the operator, *{1000,0.81}*). These updated values are used to calculate the updated EF value for the *Close_In* operator (i.e. [810]).

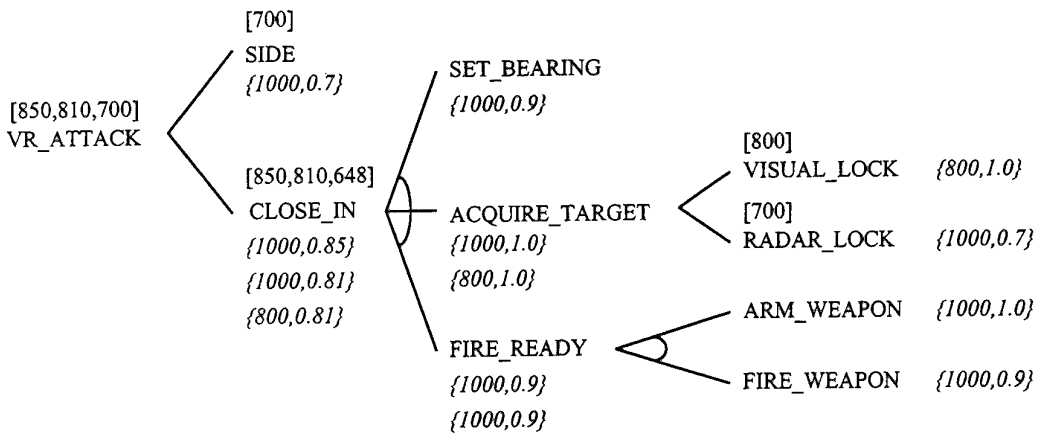

Figure 6: *An example of the abstraction hierarchy for a simple Close_In manoeuvre operator scenario that demonstrates the updating of fulfilments and probabilities. Progressive [EF] and {fulfilment, probability} values are given for each operator.*

Figure 6 also includes an example where rules 3 and 4 are used to update the fulfilment and probabilities for the parent of an OR node. In this case the *Acquire_Target* can be achieved by either a *Visual_Lock* or a *Radar_Lock*. The *Visual_Lock* is chosen as it has the higher EF of the two. However, the *Visual_Lock* has a lower fulfilment than its parent, which is propagated back through the branch. It should be noted at this point that, as a result of propagating these values back up the branch, the *side* operator becomes favourable over the expanded *Close_In* branch.

A detailed explanation and justification for these update rules is contained in Mansell (1994c). This report also highlights that the calculated expected fulfilment of a reduction operator is dependent on the accuracy of its probability and fulfilment values. When choosing between a set of reduction operators, the minimum accuracy with which one must know the EF of an action depends on the ratio of the EF of competing actions. An algorithm is given that allows a knowledge engineer to measure the required accuracy with which probability and fulfilment values must be known, to guarantee that the operator with the greatest *calculated* EF is the operator with the true highest EF. Mansell (1994c) also establishes how this quantitative search technique relates to existing search strategies used by AI planning systems. An overview of this work also appears in Mansell (1994a and 1994b).

4.5 Plan Critics

The rationale behind the use of criticism is central to the ideal of a planning hierarchy (Sacerdoti, 1977; Wilkins, 1988). U-Plan's planning algorithm produces, in parallel, a set of actions that constitutes a nonlinear plan. The critics search for destructive interactions among individual actions as a plan is produced. Since the expansion of each level of the abstraction hierarchy is based on changes made by the operators, the

search involved in any particular expansion is relatively small. To do a global search for interactions after the expansion of each reduction operator is computationally intractable (Wilkins, 1988); thus the critics are employed to periodically check the validity of the plan, and possibly modify the plan in response to any inconsistencies. A more detailed description of U-Plan's use of critics can be found in Mansell (1994c).

4.6 Summary

U-Plan constructs linear plans which describe a sequence of actions that, when applied in a particular P-state, have a given probability of producing the goal state. Planning is carried out for all P-states (with support and plausibility above a set threshold), with the most likely P-states being planned for first (as described in section 2.4). The hierarchical planning process involves selecting a strategy and reducing it to a set of executable operators.

Initially, U-Plan is given a goal to achieve. This goal function is placed as the PLANHEAD node of a plan tree. The set of reduction operators that satisfy the goal node are obtained from the plot of the goal node. These reduction operators are added to the planning hierarchy, and the subgoal they produce are constructed. The expected fulfilment for each of the subgoals is then calculated by multiplying the operator's fulfilment with its probability. The fulfilment is determined from the degree to which the operator achieves the goal of the parent operator; and the probability is gained from the probability function associated with the operator.

The planning algorithm for U-Plan uses the EF values to select which reduction operator in the strategy hierarchy is to be expanded next. The reduction operator with the greatest EF is selected first. If the necessary and satisfiable preconditions of this reduction operator are true in the active P-state, then the reduction operator is selected; else the planfail is applied. (This usually involves backtracking and trying the next best reduction operator.) When an operator is found that has both sets of preconditions satisfied in the active P-state, then the plot of that operator can be applied. Applying the plot will either expose the next level of the strategy hierarchy, or describe the changes that the action it represents make on the world.

When a complete layer of the strategy hierarchy is exposed, all previous operator selections are reviewed (as described above). If the decision is made to continue planning following the existing plan branch, another layer of the strategy hierarchy is added to the plan branch by applying all the operators as described. If the decision is made to switch plan branches, then the next layer of the strategy hierarchy will be based on the alternative operator. For example, if (in fig. 6) the decision was made to pursue a *Side* manoeuvre (instead of the *Close_In* manoeuvre), then the next layer in the abstraction would consist of the operators that constitute the *Side* manoeuvre's plot.

5. Generating Plans for Multiple Worlds

5.1 Plan Reapplication

U-Plan applies plan reapplication in an attempt to determine if a plan generated for one initial P-state can be used on another initial P-state. The desired result is fewer plans than the number of initial P-states. This is accomplished by first generating a plan for the initial P-state with the greatest support (as described in the previous section) and placing it in a plan library.[10] Then, before a plan is produced for the next best P-state, U-Plan will attempt to reuse a plan from the plan library to achieve the goal function in the new P-state. This process of first attempting to apply existing plans from the plan library continues for the rest of the initial P-states (with a support and plausibility above a given threshold). The initial P-states the plan has been successfully applied to are recorded, and this information used to add weight to the plan at plan-selection (see section 5.2).

A plan is reapplicable if all the reduction operators in the plan (that are not redundant) have their preconditions met under the new initial P-state, and when applied result in the goal state being achieved. That is, each reduction operator in the totally ordered nonlinear plan is applied to the new P-state in order, and if all operators succeed and the goal state is reached, the plan has been successfully reapplied.

If a plan, during reapplication, fails due to the unsuccessful application of an operator, that plan is not entirely discarded. U-Plan will attempt to use the part of the plan that was successful and planning continues from the point where the plan failed. The desire is to construct plans with the same or similar strategies by reusing at least part of the plan at the high level of abstraction.

Often the situation arises in which more than one plan partially works for a new initial P-state. A number of options are available as to which part plan to continue planning from; one is to select the plan with the greatest expected fulfilment; another is to select the plan with the greatest support. Both these options are appealing for different reasons (depending on the domain and type of plan desired) and are available for selection at knowledge engineering. U-Plan uses the partial plan with the greatest supporting evidence, as one of the core issues in U-Plan is to produce a super-plan that provides clearly ranked planning alternatives. By reusing the abstract portion of the plan with the greatest support, one is effectively adding the evidence for the P-state to part of the abstract portion of the plan (if plan generation succeeds from this point). When plans are merged together (see section 5.2) this process of plan reapplication should increase the total evidence for that part of the plan common to all plans that use the same plan strategy.

---

[10]Each new plan generated (or modified) is added to the plan library, which only holds plans generated for the current planning problem.

Plan reapplication is employed to ensure that plan strategies generated for the most likely possible worlds are reapplied to less likely possible worlds. The result is a set of plans that can be merged to generate a workable super-plan (i.e, a super-plan with fewer branches and knowledge acquisition operations - discussed in section 5.2).

The problem of applying a plan generated for one possible world to a completely different possible world has long been recognised as being complex (Fikes & Nilsson, 1971). The difficulty of representing internal dependencies in plans stands out as a major issue (Kambhampati, 1989). A number of methods for plan reuse have been developed (Kambhampati, 1989; Huhns & Acosta, 1988; Alterman, 1988; Hammond, 1986). The Plan reapplication[11] module is the most limiting facet of the U-Plan algorithm. The only reason U-Plan constructs totally ordered nonlinear plans is to simplify the process of reapplication of plans. However, plan reapplication is critical to planning, given uncertain and incomplete information, as:

- it allows fewer plans than there are possible worlds,
- it improves the efficiency of planning due to the complexity of generating a plan.

One obvious extension to this work would be to incorporate a hierarchical nonlinear plan reuse module such as PRIAR (Kambhampati, 1989), which would open up more complex domains and possibly increase the efficiency by producing fewer plans for possible worlds.

5.1.1 Heuristics

The working version of U-Plan allows for the application of heuristics to accelerate the search of existing plans. These heuristics involve running a rough test to determine whether the plan is likely to fail for a given P-state. The test itself involves looking for known predicate–operator combinations in the plan that are known to be incongruous. In the air combat example discussed in this report, certain predicates that make up the P-states are incompatible with particular strategies in plans. This particular heuristic test only guarantees to determine which plans will not work for certain P-states. If the heuristic test is passed, a rigorous examination of the plan is carried out.
It is not assumed that all domains suited to planning using U-Plan will utilise these heuristics. If such heuristics are not available (i.e. not entered during knowledge engineering) then U-Plan will attempt to apply plans from first principles (as outlined in the above section).

5.2 Super-Plans

Once a plan exists for all the P-states, with support and plausibility above some threshold, a single super-plan is constructed. This is achieved by merging the set of

[11]The term 'reapplication' is used here so as not to confuse this simple algorithm with the core area of research into plan reuse and modification operations.

plans constructed for the set of initial P-states with the aid of knowledge acquisition operators. The super-plan begins with the initial actions common to both plans (if any); these actions could be applied before any information is acquired. When the sequence of actions in the plans differs, a branch in the super-plan is added and the remainder of each plan constitutes each branch of the super-plan. At each branch in the super-plan a knowledge acquisition operator is added, attaining the information required to select which action in the super-plan to apply next.

For example, figure 7(a) shows a set of three simple plans (plan{ps(a)}, plan{ps(b),ps(d)}, and plan{ps(c)}), composed for four P-states. The merging of these plans generates the super-plan (as shown in fig. 7(b)). In this super-plan operators identical to all plans (i.e. *Set_Heading*) are applied first. Following this, additional information about the aggressor's altitude is sought before the defender can select his attack altitude. Similarly, the type of aggressor aircraft information may be required before committing to a particular strategy (i.e. an attack from range 45 or 5).

The situation may arise in which the required information is not available (or deemed too expensive to warrant its acquisition[12]) and an action must be selected, based on the evidence supporting each of the branches in the super-plan. The mass one attributes to each plan branch is given by:

$$m(p_i) = \sum_{\Psi \subseteq \Phi} m(\Psi) \qquad [7]$$

where p_i is the plan branch, and Φ is the set of all initial P-states that use p_i as their plan. Simply stated, the mass of each plan is equal to the sum of the mass of the initial P-states that use the plan; this includes the mass attributed to the disjunction of any of the initial P-states in the set. The mass associated with each branch of the super-plan is therefore equal to the sum of the masses of the plans included in that branch. This information is used to calculate a support and plausibility (using the equations provided by Dempster-Shafer theory (Shafer, 1976)) for each plan branch.

[12]See section Knowledge Acquisition Operators' for further discussion on this.

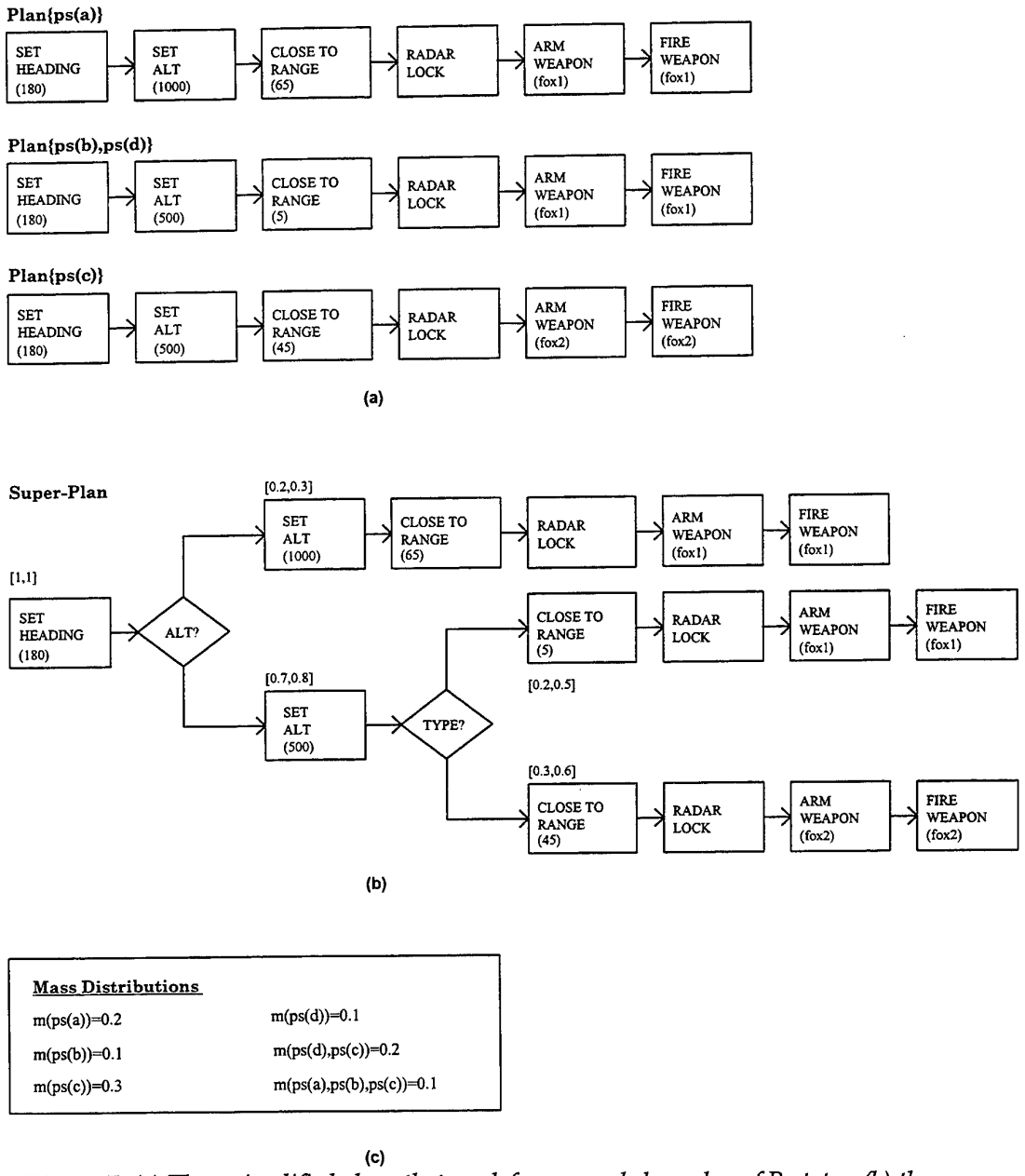

Figure 7: (a) Three simplified plans that work for a recorded number of P-states; (b) the super-plan generated from these simple plans; (c) the mass distribution for the P-state frame.

For example, in figure 7(c) a mass distribution is given for the frame of discernment that encompasses the four P-states planned for in figure 7(a). An evidential interval is calculated for each point in the super-plan where branching occurs (represented by the numbers in parentheses above the first operator in each branch). The first branch occurs when deciding whether to attack at altitude 1000 or 500, depending on the

altitude of the aggressor. If this information were not available (or not cost effective) the decision is made based on the evidential interval calculated of either branch, which in this case sees a clear win for altitude 500. However, in the second branch of the super-plan a far closer contest, although still clearly preferred, is to attack from range 45 instead of 5.

5.3 Knowledge Acquisition Operators

When deciding whether to acquire information, it is important to evaluate the cost as well as the benefit. This cost may be in the form of the time it takes, the resources it uses, the direct effect it has on the environment, or the information it also gives to an opponent. A trade-off exists between when to acquire this additional information, based on the cost of doing so, and the relative advantage it provides when selecting a branch. For example, should we acquire the altitude information in figure 7(b) if one branch is clearly more likely than the other?

The knowledge acquisition operators (KA operators) are not treated as normal reduction operators. For starters, they can only be called by U-Plan during the composition of a super-plan. An example of the *Acquire_Altitude* KA operator is given in figure 8. The KA and reduction operators are similar (so as to take advantage of certain parts of the mainstream U-Plan coding), but define distinct properties. Each KA operator contains information about how it is to be applied. Included in the KA operator's framework is the *name*, the *abstraction level* it operates in, the *necessary preconditions* that must be true of the P-state, and the *satisfiable preconditions* it will attempt to make true (as in the mainstream reduction operators). The *plot* of the reduction operator provides step-by-step instructions on how to perform the action represented by the operator, and how the operator changes the state of knowledge about the world. The *postconditions* summarise what the operator intends to accomplish. Finally, the KA operator includes a slot that estimates the cost of acquiring the information. At present, the cost can be either a constant or variable. One example of when the cost of acquiring information could be thought of as constant is when it depends on the time required to make the acquisition[13] (e.g. visual identification of the aggressor aircraft by another agent).

The *Acquire_Altitude* KA operator is an example of variable cost. In this case, the variable is a key word which corresponds to a numerical value in a constantly maintained look-up table. In general, KA operators directly or indirectly change a dynamic attribute of the P-state that requires quantifying. In the *Acquire_Altitude* KA operator (fig. 8), obtaining the altitude of the aggressor requires the radar to be set to *track* mode, which results in the defender possibly losing the element of *Surprise*. The

[13]However, U-Plan does not deal specifically with temporal constraints such as the time it takes to execute specific actions.

Name:	Acquire_Altitude
Level:	3
N-Precond:	Radar
S-Precond:	
Plot:	(AND (Radar(track)) (Alt3 (!x)))
Postcond:	Alt3
Cost:	Surprise

Figure 8: An example of the Acquire_Altitude knowledge acquisition operator.

value *Surprise* may be high (depending on the manoeuvre being performed) or it may be zero if the aggressor knows the defender's location.[14]

The planned acquisition of additional knowledge in the super-plan is not guaranteed to take place at any time. The necessary and satisfiable preconditions outline under what conditions the KA operator can take place. However, the possibility also exists that the additional information will not be obtained due to exogenous forces acting on the environment. Under such circumstances the support and plausibility, calculated for each branch of the super-plan, can be used as a quantitative measure of the weight of evidence supporting that particular planning alternative. If the given KA operator fails (or is inappropriate under the prevailing conditions) the plan branch with the best evidential interval may be chosen to be pursued (as occurs when no KA operator exists, discussed in section 5.2).

This aspect of evaluating the cost of acquiring additional knowledge and trading this off with the value of the information is a very complex issue which is not addressed in any detail by this work. U-Plan incorporates a cost facility that is intended to be used by a human operator who, using his own experience and judgement, will decide the worth of knowledge acquisition.

[14]The explicit representation or modelling of another agent's knowledge is not specifically dealt with here. Instead, an assessment of the aggressor's state is solely based on the defender's state description.

6. Dynamic Environment

The assumption that the world remains static during planning has been a point of criticism among classical planners, and an argument for the use (at least partially) of reactive planning techniques. U-Plan is designed to operate in a dynamic environment, that is, one where the body of information available to describe the state of the world may be constantly changing. Of particular interest to the U-Plan system is when changes to the balance of information affect the P-states, and how *significant* those changes may be on existing plan (i.e. whether they necessitate replanning).

U-Plan uses a separate module to monitor changes in the dynamic environment and will interrupt the planning process if the world changes significantly. To define a significant change in a proposition held in a P-state, and the true state of the environment, we must first look at how the initial information may change.

The environment in which planning takes place is recognised as being dynamic. So how does one maintain an accurate description of the environment for the purpose of plan generation? If a dynamic view of the world is held, the planner runs the risk of having some of the propositions necessary to specific operators negated before the plan can be executed. This has been a long recognised problem in the planning literature, resulting in a number of planning paradigms (anytime algorithms (Dean & Boddy, 1988), dynamic planning (Cohen *et al.*, 1989; Hayes-Roth *et al.*, 1989; Georgeff & Ingrand, 1987; Raulefs *et al.*, 1987) and reactive planning (Schoppers, 1987; Georgeff & Lansky, 1987).

In the dynamic and imprecisely described environment, the view of the world held in the P-state can become incompatible with the true description of the world in two ways:

1. The propositional statements used to capture an initial piece of evidence may change at a later date, invalidating the statement in the P-state. This is an easily detected change in the environment that could either invalidate a plan, or have no effect on the plan. U-Plan does not tackle the difficult problem of determining when an altered proposition invalidates a plan. However, a simplistic algorithm is employed that will repeal a plan if a proposition required by a precondition of an action is invalidated. When such a change in the environment is recorded, replanning is the course adopted by U-Plan. This algorithm does not reason about any interaction between operators that may have compensated for that invalidation, or distinguish between necessary and satisfiable preconditions. Examination of these and other responses to changes in the world is a good future direction for research.

2. The degree of evidence supporting a particular proposition in the initial P-state may change in time. When this occurs, the mass distribution (and consequently the

support and plausibility) for the propositions may result. The effect in this situation is a change in the mass distribution for the generated plan(s).

It is the second condition that is of most interest to U-Plan. That is, when is a change in the environment significant enough to alter the order in which two plans are ranked (using their evidential interval)? It turns out that the rank order of two plans does not alter until the rank order of the evidence supporting the propositional statements changes (Mansell, 1994c); for example, if two propositional statements are given the following mass distribution:

$$m(Alt_3((0\ 1))) = 0.4$$

$$m(Alt_3((2\ 3))) = 0.2$$

$$m(Alt_3((0\ 1),(2\ 3))) = 0.4$$

Let us assume a change in the environment sees an alteration in the mass distributed among the Alt_3 predicates (i.e. a change in the degree of uncertainty in the environment). Then the rank order of the plans produced for the super-plan will not change unless the mass attributed to $Alt_3(0\ 1)$ falls below the mass attributed to $Alt_3(2\ 3)$.

7. Implementing U-Plan

U-Plan has been implemented and extensively trialled in a simulated air combat domain. This section outlines some of the results achieved in this domain, including (1) the number of plan branches (i.e. individual plans produced) given different numbers of P-states; (2) the time taken to produce a super-plan; and (3) the sensitivity of U-Plan to the initial evidence provided (discussed briefly in Mansell, 1993b).

In order to assess the operation of U-Plan a control planning system, C-Plan, has been constructed. C-Plan is a hierarchical planner that uses the same decision theoretic operator selection process as U-Plan, but does not attempt to reapply plans, merge plans, or acquire knowledge.

7.1 Plans Generated

When planning given a set of possible worlds, C-Plan will construct a plan for every initial P-state. The number of plans generated by U-Plan depends on the domain. In the air combat domain, U-Plan produces substantially fewer plans than C-Plan (see fig. 9). This is largely due to U-Plan's ability to reapply plans in particular types of domains. Also included in figure 9 is the number of *unique plans* generated by C-Plan for the purpose of distinguishing between the reduction in plan numbers due to redundant information and the reapplication of plans. The difference between the C-

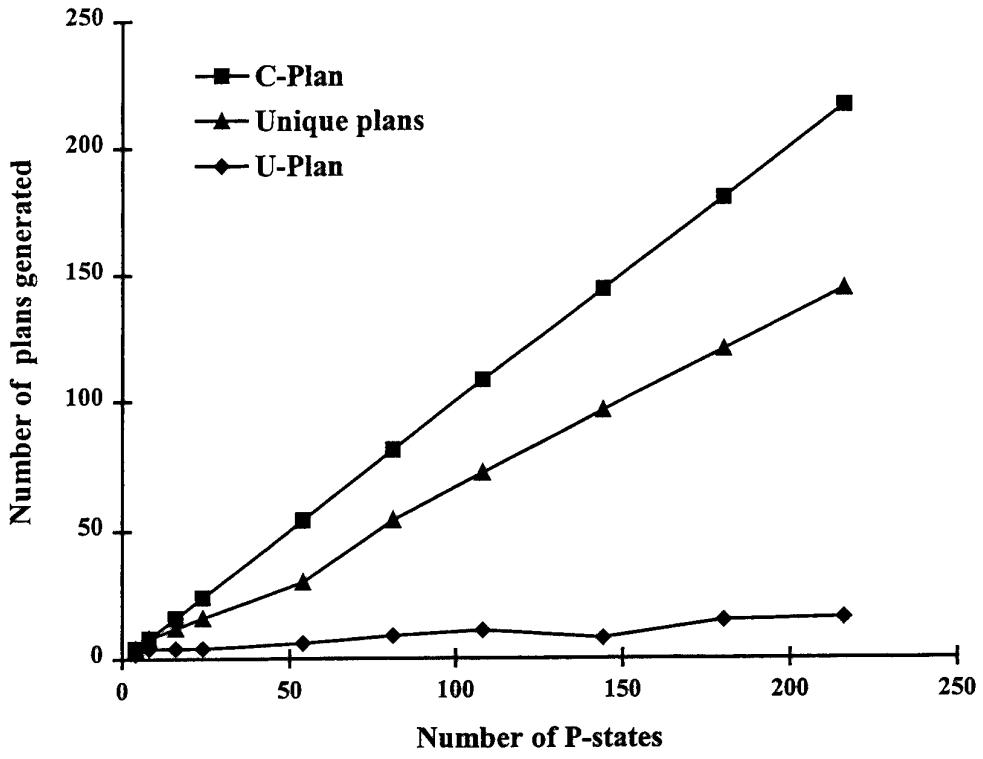

Figure 9: Comparison of the number of plans generated by U-Plan, C-Plan and the number of unique plans generated by C-Plan in the air combat domain.

Plan line and the *unique plan* line represents identical plans that have been merged to determine how many different plans were generated. This implies that the information which differentiates the P-states and results in a new plan to be generated was superfluous (at this time), as the same plans are generated regardless. However, this information in other P-states may generate different planning alternatives (i.e. the information is redundant in a limited number of scenarios). The difference between the *unique plan* line and *U-Plan* line represents the reduction in plan numbers due to the reapplication of existing plans to other P-states. These P-states when planned for individually, produced different plans from those attributed to them by U-Plan.

The reduction in the number of plans here is aided by the type of the domain which, by its nature, sustains a number of possible planning solutions. In a worst case domain, i.e. where a different plan must be generated for every possible world, U-Plan produces one plan for each initial P-state.

7.2 Timing

The process of reapplying plans can be costly. The more plans U-Plan generates, the larger is the set of possible plans that may be reapplied. The effect of this overhead can be alleviated by the implementation of some simple heuristics. In the air combat domain heuristics can be used to quickly evaluate the suitability of a plan to the current P-state. Figure 10 plots the amount of CPU time spent running C-Plan, U-Plan and a version of U-Plan that uses heuristics during plan reapplication, H-U-Plan. This demonstrates that, when heuristics are available, U-Plan constructs fewer plans and intends to acquire the knowledge to differentiate between them with no time penalty over planning for every possible state.

As demonstrated in Figure 10, without the use of heuristics U-Plan is more time-consuming than the C-Plan system. The reason for this is in the amount of time U-Plan spends searching existing plans for a solution to different P-states. To combat this problem, a set of heuristics are used to reduce the amount of time spent searching for alternative plans.

The heuristics used in the alternate version of U-Plan involve applying a rough test to determine whether the plan is likely to fail for a given P-state. The test itself involves looking for known incongruous proposition–operator combinations in the plan. In the air combat example used here, certain propositions that make up the P-states are incompatible with particular strategies in plans. This particular heuristic test only guarantees to determine which plans will not work for certain P-states. If the heuristic test is passed, a rigorous examination of the plan must be carried out.

U-Plan spent on average almost 80% of its time (fig. 11) searching for existing solution plans for between 8 and 216 P-states (represented by the *U-Plan search* line). The time spent by H-U-Plan searching (represented by the *H-U-Plan search* line) the existing plans for a solution is substantially lower. The difference between these two lines represents the total amount of time saved using the heuristics. When heuristics are not available, the reapplication of plans becomes the overriding factor in the amount of time spent planning.

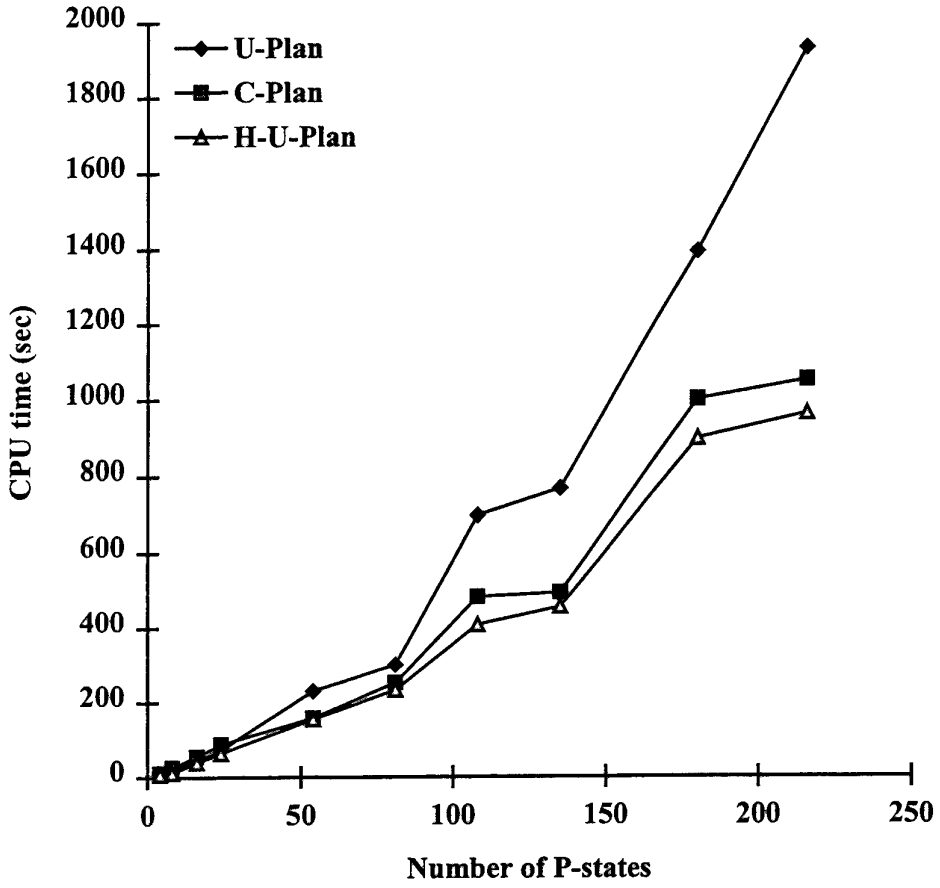

Figure 10: Comparison of the amount of CPU time used by U-Plan, C-Plan and H-U-Plan (or Heuristic-U-Plan) for various numbers of initial P-states on a Symbolics 3645

In the air combat example, U-Plan produces plans that, mostly, reuse a small number of strategies. This is made possible by the reapplication of plans, originally produced for one P-state, and utilised by a number of other P-states. The cost of reducing the number of unique plans generated is paid in the time taken searching for a suitable existing plan. This computational result was recorded, even though reapplying a single plan is faster (approximately 0.4 CPU seconds) than generating a new plan (approximately 2.2 CPU seconds).

7.3 Sensitivity of Initial Evidence

Initial information about the world is collected by U-Plan and used to generate a set of P-states. This information has a mass assigned to it representing the degree to which it

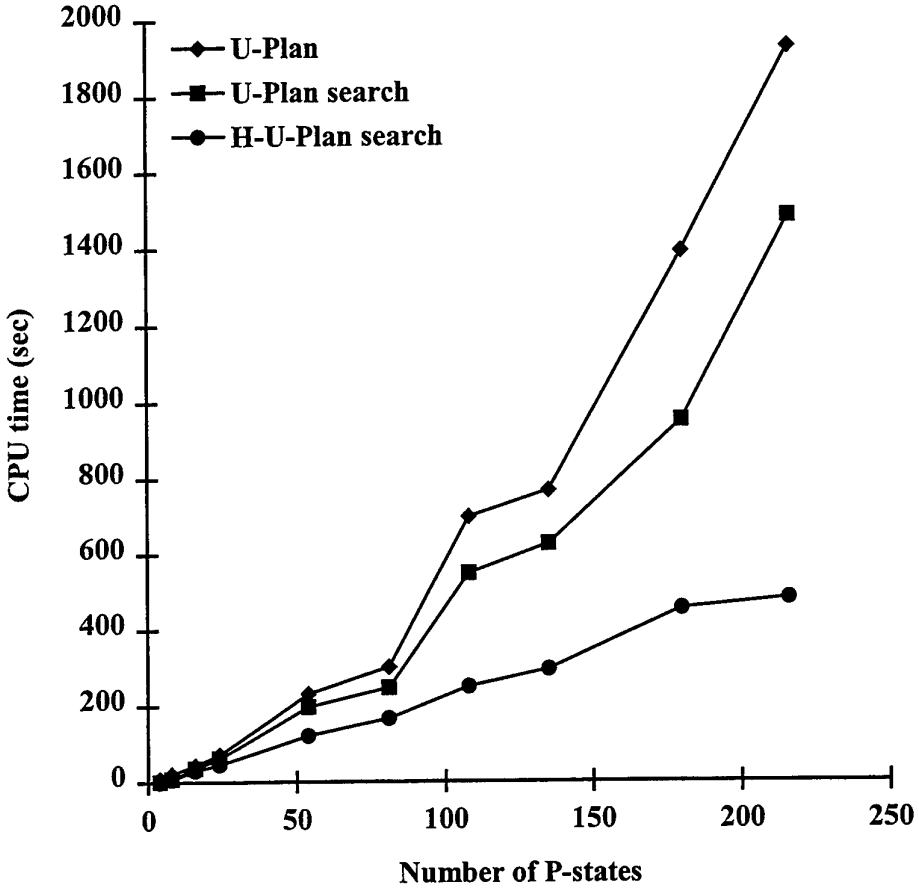

Figure 11: Shows the total time taken by U-Plan, the time spent reapplying existing plans, and how this search time can be cut using H-U-Plan's heuristics.

is believed the information accurately characterises the environment. These masses are used to calculate the support and plausibility of each P-state, which are in turn used to determine the order in which P-states have plans generated for them. Of particular interest to the user of U-Plan is the degree of change required to the initial evidence's mass distribution to produce a super-plan that has different alternative branches, or a changed rank order of those branches, to form an original super-plan.

The analysis of the initial information's sensitivity has been carried out by running a large number of trials on three sets of possible world scenarios. In all three examples, the point at which the relative order of supported states changes produces an alteration in the execution order of the super-plan. The type of change in the super-plan depends on the proposition that triggered the alteration. If a tactical change in the relative ordering occurs, a tactical change to the super-plan is observed. A tactical

change occurs when the proposition at the lowest level changes to another proposition, but both propositions share the same representation at the next level of abstraction. For example, the change from $Alt_3(2\ 3)$ to $Alt_3(4\ 5)$ is a tactical change, as both propositions are represented at the next level by the same proposition, $Alt_2(low)$. Conversely, the change from $Type_3(f1)$ to $Type_3(b1)$ is a strategic change, as they are represented by different propositions at the next level.

For example, if the most likely altitude of the opponent's aircraft were to change from $Alt_3(4\ 5)$ to $Alt_3(11\ 20)$, the order in which the super-plan would recommend execution given no further information would change from an attack at $Alt_3(4\ 5)$ to an attack at $Alt_3(2\ 3)$. Similarly, if the change is in the ordering of a strategic proposition, a new set of strategies is likely to be adopted by the super-plan. An example of this may be when the greatest support goes from aircraft type *fighter* to *bomber*. Such a change in evidence is likely to result in the super-plan containing a completely different set of plan strategies, say, from a *Cutoff_Intercept* to a *Hook* manoeuvre.

This behaviour is illustrated in figure 12. This plot shows how changing the relative order of the mass functions of propositions alters the type of super-plan generated for a given environment. In this environment, 4 of the attributes are uncertain and each has 3 possible values. (Evaluation of a 16 P-state and a 256 P-state was also undertaken with similar results; Mansell, 1994c). In the 81 P-state case the opponent's velocity, aircraft type, altitude and heading attributes are all uncertain and have non-zero mass associated to 3 different possible values.

The sample environments under consideration were examined for their reaction to altering the level of support from one to four of the propositions. The result of such changes depends on the proposition(s) altered. However, in all cases a change in the super-plan was recorded only when the relative order of the values of specific attributes changed. In each of these environments, the super-plans produced are broken up into four types:

- type 1—are the original super-plans produced before mass numbers are varied
- type 2—are the same original super-plans with a change in the tactical order
- type 3—are the same original super-plans with a change in the strategic order
- type 4—are strategic super-plans different from the original super-plans.

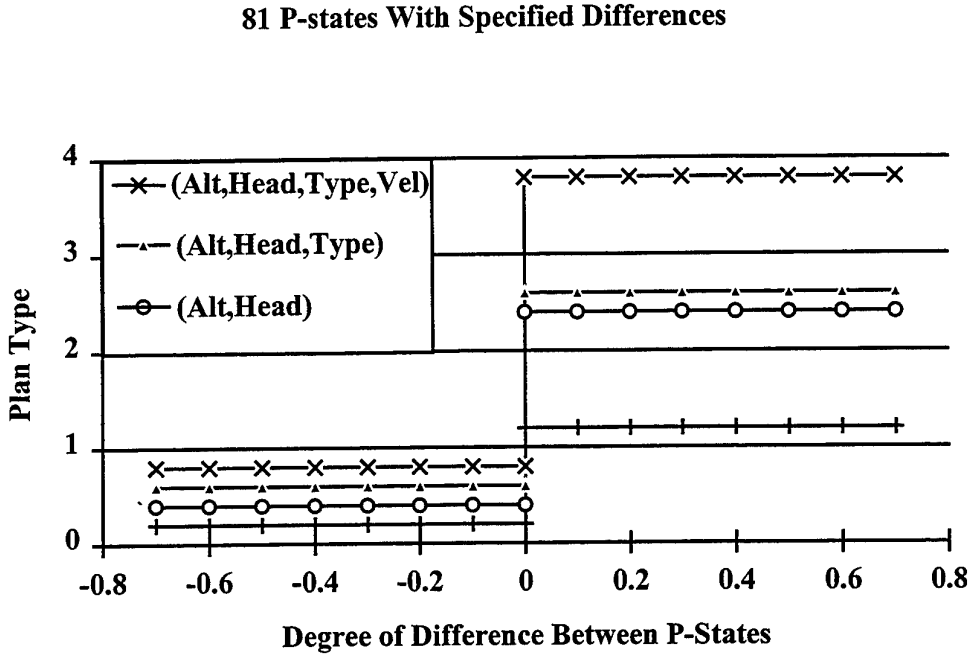

Figure 12: Demonstrates the point at which a change in the mass distribution of the initial evidence alters the super-plan generated in a 81 P-state environment. The progression from negative to positive represents a change in rank order of initial evidence propositions, the x-origin being the cross-over point.

In the 81 P-state example (fig. 12), varying the tactical attribute (*Alt*) changed the tactical order of the super-plan, from a *Cutoff_Intercept* at (11 20) to a *Cutoff_Intercept* at (21 30). In the case where two attributes (*Alt* and *Head*) are altered, the super-plan undergoes a strategic reorder from a *Cutoff_Intercept* manoeuvre having the highest support to a *Hook* manoeuvre having the highest support. The change of three attributes (*Alt, Head* and *Type*) had the same result as the two proposition example; a change in the strategic super-plan order. Finally, changing four attributes (*Alt, Head, Type* and *Vel*) resulted in the generation of an entirely new Super-plan, where neither the *Cutoff_Intercept* nor the *Hook* manoeuvres have the greatest support.

One observation to be made is that, in general, when the rank ordering of a tactically critical attribute is altered, a tactical change to the ensuing super-plan occurs. This can be considered a minimal change to the super-plan. Similarly, a rank order change to a strategically critical attribute generally resulted in a strategically different super-plan. When both strategically and tactically critical attributes are altered, the maximum effect (i.e. a strategic change) to the super-plan can be expected. A strategic change is a more significant alteration to the super-plan, as it highlights that replanning must take place to generate the same super-plan given the evidence changes. Conversely, when

the evidence changes result in an alteration of super-plan branch ordering, redistribution of the initial evidence can be used to update the initial super-plan.
An attribute is not limited to being exclusively tactically or strategically critical. Attributes may exhibit strategic qualities under certain circumstances, and tactical qualities under others. An example of this is the opponent's velocity. When a Chase_Manoeuvre has been selected, the velocity only influences the selection of a closing speed. However, depending on the angle of attack, the velocity will also influence the type of manoeuvre up for selection.

It is useful, when monitoring a dynamic world, to understand when a change in the support for values of an attribute will influence an alternative super-plan. Such information allows U-Plan to make an informed decision as to when the evidence describing the environment has altered significantly (see section 6 for a clarification of this term), with respect to the stored representation of the world. At this point, the two decisions available to the planner are: (1) to scrap the plan and start from scratch; (2) re-evaluate the plans generated so far. A discussion of U-Plan's response to this problem is found in section 6.

7.4 Summary of Results

A number of empirical results have been presented in this section. The performance based data has been compared with a control planner called C-Plan, which creates a plan for every possible world. The results obtained from the air combat domain showed that:

- U-Plan is capable of producing a super-plan in less time than it would take a traditional planner to produce one plan for every possible world. The most important factor in reducing the run time speed is the amount of time spent attempting to reapply existing plans to new P-states.
- The U-Plan algorithm generates substantially fewer plans than one for every possible world. U-Plan has the added advantage, over C-Plan, of presenting these plans in the form of a super-plan which takes advantage of common action sequences and incorporates the acquisition of additional information to choose between plan alternatives (when that information is attainable).
- The branch orderings within a super-plan generated by U-Plan are sensitive to the ranking of the initial evidence gathered. The order in which a super-plan would be executed (when additional knowledge is unavailable) is not altered by a shift in the distribution of mass among propositions until the order in which the propositions is altered changes.

8. Related Work

The problem of planning under uncertainty is currently being addressed by a number of people. Researchers at SRI International (Wilkins, Myers, Lowrance & Wesley, 1994)

have developed Cypress, an AI planning system developed to operate in dynamic and unpredictable environments. Cypress is based on the merging of a number of mature, and powerful planning and execution technologies, namely the SIPE-2 (Wilkins, 1988) plan generator, the PRS-CL (Georgeff & Lansky, 1987) reactive execution system, and the Gister-CL (Lowrance *et al.*, 1991) system for reasoning under uncertainty.[15] These systems are able to interact through the services of the ACT language.[16]

The approach taken by Wilkins *et al.* is to construct a plan for a user defined goal using the well established planning system SIPE-2. The plan generated by SIPE-2 is then translated to Acts and executed by the executor. The user can instruct the uncertain reasoner, GISTER, to reason about uncertain information. In particular, GISTER is used to select between alternative objects and resources to be used in the plan, and to choose between alternative operators that could be applied to achieve the defined goal. Plan generation takes place while PRS is continually monitoring the environment and instructing the executor (via Acts) on how to react to events as they arise.

This system differs from that of U-Plan, as Cypress effectively produces a plan for the possible world with the greatest evidential support, then monitors the execution of that plan. If changes in Cypress's representation of the world (by unpredictable events or updated evidence) adversely affect the plan, SIPE-2 produces a new plan and execution continues till the goal has been achieved. Cypress is blessed with the advantage of linking mature planning and reacting technologies, resulting in a potentially powerful system. U-Plan does not examine planning from the executor's side of the equation (like Cypress), as its output is intended to be used by a human operator as a decision aid, and hence, is not likely to be privy to the feedback provided in Cypress.

Lowrance and Wilkins (Lowrance & Wilkins, 1990; Lowrance *et al.*, 1991) also present a formalism for constructing plans given both incomplete and uncertain information about the initial state of the world using operators with uncertain outcomes. This algorithm first generates the set of possible worlds using uncertain and incomplete information (as does U-Plan). However, this is where the similarity ends. The approach taken by Lowrance *et al.* is to construct a plan for each possible world using the well established planning system SIPE-2 (Wilkins, 1988). The plans generated by SIPE-2 are then fed into the uncertain reasoning system GISTER (Lowrance, 1991), where a quantitative measure (in the form of an evidential interval (Lowrance, 1991)) is calculated that rates the likelihood of that plan being successfully executed. The result is a pool of plans, each containing a measure of possible success given the evidence available to describe the environment. This approach differs from that of U-Plan, as U-Plan does not intend to produce a plan for every possible world, and Lowrance *et al.* do not include the intention to acquire additional information.

---

[15] SIPE-2, PRS-CL, and Gister-CL are trademarked systems produced at SRI International.

[16] ACT is a domain-independent language for representing knowledge about actions used by both planners and executors.

A number of nonmonotonic reasoning approaches have been applied to planning given approximate information about the environment. Elkan (1990) has developed an approximate planning system by first defining a new formalism for describing the planning problem. The candidate plans are quickly found by allowing them to be based on unproven assumptions about the world. The formalism describing the planning problem maintains details of which antecedents of the rules remain unproven (the default rules), allowing only plausible candidate plans to be constructed. Planning then redefines the candidate plans incrementally by attempting to justify the assumptions on which they depend.

Approximate planning systems are useful when precise and qualitative information about the world is all that is available. They may also be useful when a planning decision must be made before all assumptions in the plan can be proven. As discussed in section 7, U-Plan also operates effectively in a qualitative environment, where the imprecision equates to being given the approximate orderings of belief in the information (although these orderings must be expressed quantitatively).

Researchers at the University of Washington have developed a planning language that operates in an environment which does not assume the availability of complete information. UWL[17] (Etzioni, Hanks & Weld, 1992) is an extension of the STRIPS language (Fikes *et al.*, 1971). The basic concept behind UWL is to annotate the preconditions and postconditions (similar to the STRIPS add and delete lists) of operators to specifically represent: (1) change in the state of the world; (2) change in the state of knowledge about the world; (3) provision for injunctions against changing specific propositions that describe the world; and (4) use of information that will not become available until run-time. The operators are defined as either causing a change to the state of the world, or observing attributes of the world (i.e. acquiring knowledge about the world). This allows Etzioni *et al.* to create a provably correct plan in the form of a totally ordered set of operators. This plan could be represented as a tree in which conditional branches are incorporated when insufficient information is available to determine if a specific operator is appropriate.

This work demonstrates that, by simply extending the preconditions and postconditions of standard STRIPS operators, a planning system capable of knowledge acquisition goals can be produced. Such a system is capable of generating a provably correct plan given accurate but incomplete information. UWL uses a substantially different form of operator representation from that of U-Plan; this is partially because U-Plan is a hierarchical planner. The final plan produced by UWL includes conditional branches (as does the U-Plan super-plan) that order the final execution of operators based on additional information acquired. The major difference between UWL and U-Plan is that UWL uses incomplete but accurate information about the environment.

Following the work on UWL, an algorithm for probabilistic planning called BURIDAN (Kushmerick, Hanks & Weld, 1995)) has been developed at the University of

[17]UWL stands for the University of Washington Language.

Washington. BURIDAN is a probabilistic approach to planning given uncertain initial information and actions with non-deterministic effects. This algorithm models the initial world state using a probability distribution over possible worlds, and models actions using a conditional probability distribution over changes in the world. BURIDAN searches through a space of plans, terminating when it finds a sequence of actions that have a probability of achieving the goal greater than a user-supplied threshold. This approach uses a different action representation and planning algorithms from U-Plan and does not provide information producing actions. However, both techniques operate using an uncertain initial state of the world.

C-BURIDAN (Draper, Hanks, and Weld, 1994) is an extension to BURIDAN's probabilistic planning algorithm, combined with a framework for contingent action response based on the SNLP algorithm (Poet & Smith, 1992). C-BURIDAN is a step closer to U-Plan in that knowledge producing operators have been incorporated into the planning algorithm. However, U-Plan uses knowledge acquisition to distinguish between planning alternatives, while C-BURIDAN uses knowledge acquisition to determine whether actions contingent on the information provided can take place. This allows C-BURIDAN to provide plan branching and rejoining. (Rejoining is not a facility provided by U-Plan.)

Both planning algorithms use probabilistic information to generate a set of possible worlds. (U-Plan also uses incomplete information.) The planning algorithms, through entirely different procedures (e.g. C-BURIDAN is not a hierarchical planner), produce a plan composed of causal and knowledge producing actions for the set of possible worlds. U-Plan is designed to provide workable plans in complex environments that require the course of action to be based on achieving a strategic goal. On the other hand, C-BURIDAN would appear to be best suited to lower level planning based on achieving tactical goals in domains where only a handful of possible worlds exist (particularly as plan rejoining becomes costly as the number of plan branches increase).

SUDO-Planner (Wellman, 1990) specifically deals with domains where uncertainty arises, because an agent cannot flawlessly predict the state of the environment after the application of operators. SUDO-Planner uses qualitative probabilistic networks to represent and reason about actions. This system also uses tradeoff formulation (Wellman, 1990) to separate significant decisions from trivial choices. SUDO-Planner constructs plans by applying actions that change the belief attributed to a proposition or event, and consequently allowing for partial goal satisfaction. This is an aspect of uncertainty not dealt with by U-Plan.

Bonissone and Dutta present RUM (Bonissone, Dutta & Wood, 1994), an approach for planning where an uncertain and incomplete description of the environment is likely, and where the effects of actions in the environment are not assumed to be completely predictable. Planning takes place in an abstraction hierarchy that is capable of reacting to changes in a dynamic environment. Planning in RUM can be considered as a search for the best possible strategy to adopt. Plans are generated based on the rule-based

selection of strategic operators for which a measure of *desirability* is calculated. This desirability is based on a measure of the rules value given an uncertain and incomplete description of the environment.

Of all the planners described in this section RUM is most comparable to U-Plan, as it deals with uncertain and incomplete domain information. They both plan in an abstraction hierarchy, searching for the best possible plan strategy to apply. However, where U-Plan uses a more conventional AI approach to planning, RUM is more closely related to an advanced expert system. U-Plan (through the super-plan) provides a course of action that takes an agent from an initial state to a goal state, acquiring additional information, as specified, along the way.

8.1 Domains of Application

Planners have been applied to a number of application domains (as well as the simple exploratory domains such as the blocks world). Table 1 characterises existing planning under uncertainty technologies in a 2-dimensional space of required response time and the uncertainty measures used. The entries that contain "N/A" imply that, although a system may come under this heading, it is not likely to be a planning system. Entries that are left blank imply the author is not aware of an appropriate planning system. This table is intended as a guide to selecting the appropriate technology given the domain, as well as a summary of these systems. What follows is a summary of these planners' domain(s) of application.

- SUDO-Planner has been applied to what has been termed the clinical decision making where treatment must be selected based on the diagnoses of inconclusive symptoms.

- RUM has been applied to a mergers and acquisitions domain resulting in MARS (a mergers and acquisitions reasoning system) (Bonissone *et al.*, 1994). MARS generates decisions on the take-over strategies and tactics using uncertain and incomplete information about the target company's situation and the takeover climate in a dynamic environment.

- CYPRESS has been applied to a military force planning domain, where the object is to allocate the correct military units for specific missions. This information is based on such imprecise information as the expected strength of enemy and friendly forces. A units strength is based on the assessment of the number of troops, morale, readiness, training and experience, mobility, etc.

- Elkan has applied his planning algorithm to more typical elementary AI problems such as the Yale shooting problem and the Christians and the lions problem.

- BURIDAN has been applied to a typical robot motion planning domain, where the object is to plan the actions of a robot to accomplish specified goals in its environment. The actions taken in this domain usually involve interacting

Table 1: Planning technologies and their domains.

Response Time	Uncertainty Measures			
	Bayes	**D-S**	**Fuzzy**	**Qualitative**
Real Time	Control theory **Robot control**		N/A	N/A
Emergence Response	BURIDAN & C-BURIDAN **Robot motion planning**	U-Plan **Air combat and chemical spills**	N/A	N/A
Exhaustive		Cypress **Military force planning**	RUM **Mergers and acquisitions**	SUDO-Planner **Clinical decision making** Elkan

(grasping, moving, painting, etc.) coloured blocks, given a probability distribution over an initial set of possible states.

- C-BURIDAN extends the above robot motion planning environment to allow *observe* (or knowledge acquisition) *operators*, and actions that are contingent on the information they provide. For example, the *Inspect* operator is used to observe whether an object is blemished and should be rejected.

- U-Plan has been applied to a two aircraft air combat domain, where the object is to plan the actions of a defending aircraft using only available information.

9. Conclusion

This report outlines the ideas behind U-Plan, a system for planning given an uncertain environment and incomplete information. The system represents the incomplete and uncertain description of the environment using a set of possible worlds. Each of these possible worlds contains a representation of the world at a number of abstraction levels. This enables the planning system to make high level (strategic) decisions, based on a high level representation of the world. As more abstract world representations generally encompass a number of low level possible worlds, the high level planning can develop a strategy for a number of possible worlds. This allows the system to commit more readily to a plan strategy, when given a number of possible states of the world.

The planning system's ability to fit part or all of an existing plan to a number of possible worlds has the potential to produce fewer plans than one for each possible world. The computational cost of planning in this manner depends on the domain of application, and the relative number of plans produced. Any computational saving over a decision theoretic hierarchical planner constructing plans for each possible world relies on two properties: (1) U-Plan constructing fewer plans than one per P-

state; and (2) the application of heuristics to quickly and accurately determine whether attempting to reapply a plan to another P-state is worthwhile. As U-Plan's speed depends on producing a manageable number of possible plans, the system is targeted towards domains where a number of possible plans exist for the possible worlds. Hence, planning is based on constructing a superior plan that achieves the system's goals given the available information. Conversely, a domain that required a unique plan for every possible world would be more computationally costly to plan for under U-Plan.

The inclusion of the system's ability to plan to acquire information when beneficial to do so enhances U-Plan's operation. What results is a plan tree that exploits common action sequences, and provides the mechanism to select which branch to pursue in the plan tree. This yields a more favourable result than producing a new plan for every possible world.

U-Plan is intended for use within domains where decisions must be made before full knowledge is available; in particular, emergency type domains where actions should proceed while information is being collected, or complex environments where complete information is unobtainable. U-plan is currently being applied to a fire hazard action response domain, in which a super-plan is constructed to combat industrial fires.

A number of possible extensions to U-Plan have been identified as areas for future research. Initially, U-Plan could be altered to generate partially ordered plans, and a module added that reapplies partially ordered nonlinear plans to P-states. A number of methods for plan reapplication have been developed (Kambhampati, 1989; Huhns & Acosta, 1988; Alterman, 1988; Hammond, 1986). The facilities to do temporal reasoning are an important aspect of real world planning; hence adding such facilities would be a logical extension to U-Plan. The role of a temporal reasoning module would be to include in plans the time constraints posed by realistic actions. Other areas of interest include: allowing for operators that have an uncertain result; investigation into how to identify the best possible world to begin planning; and applying this system to other domains.

Acknowledgments

The author gratefully recognises Grahame Smith of the Cooperative Research Centre for Intelligent Decision Systems and Elizabeth Sonenberg of The University of Melbourne for their many insightful comments on this work. I am also thankful to the staff at the Australian Artificial Intelligence Institute (AAII), as this research has benefited from their excellent research environment and fine facilities.

The majority of research documented in this report was completed while the author was a DSTO postgraduate research fellow with The University of Melbourne.

References

R. Alterman (1988), Adaptive Planning. *Cognitive Science*, 12.

P. P. Bonissone, S. Dutta and N. C. Wood (1994), Merging Strategic and Tactical Planning in Dynamic, Uncertain Environments. In *IEEE Transactions on Systems Man and Cybernetics*, 24(2): 841–862.

D. Chapman (1987), Planning for Conjunctive Goals. *Artificial Intelligence*, 32: 333–377.

P. R. Cohen, M. L. Greenberg, D. M. Hart and A. E. Howe (1989), Trial by Fire: Understanding the Design Requirements for Agents in Complex Environments. In *AI Magazine*, 10(3): 34–48.

T. Dean and M. Boddy (1988), Reasoning about Partially Ordered Events. *Artificial Intelligence*, 36(3): 375–399.

D. Draper, S. Hanks, and D. Weld (1994), Probabilistic Planning With Information Gathering and Contingent Execution. In *The Second International Conference on Artificial Intelligence Planning Systems*, Chicago, 31–36.

C. Elkan (1990), Incremental Approximate Planning: Abductive Default Reasoning. In *Working Notes of the AAAI Spring Symposium on Planning in Uncertain, Unpredictable, or Changing Environments*, 34–38.

O. Etzioni, S. Hanks and D. Weld (1992), An Approach to Planning with Incomplete Information. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, Massachusetts, 115–125.

R. Fikes and N. Nilsson (1971), Strips: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2: 189–208.

R. Fikes, P. Hart and N. Nilsson (1972), Learning and Executing Generalised Robot Plans. *Artificial Intelligence*, 3: 251–288.

M. Georgeff and F. Ingrand (1989). Decision-making in an Embedded Reasoning System. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. Detroit, Michigan, 972–978.

M. Georgeff and A. Lansky (1987), Reactive Reasoning and Planning. In *Proceedings of the 6th AAAI*, 677–682.

K. J. Hammond (1986), CHIEF: A Model of Case-based Planning. In *Proceedings of the 5th AAAI*, 267–271.

S. Hanks and D. McDermott (1994), Modeling a Dynamic and Uncertain World I: Symbolic and Probabilistic Reasoning about Change. *Artificial Intelligence*, 66(1), 1–56.

F. Hayes-Roth, R. Washington, R. Hewett, M. Hewett, and A. Seiver (1989), Intelligent Monitoring and Control. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 243–221.

M. N. Huhns and R. D. Acosta (1988), ARGO: A System for Design by Analogy. *IEEE Expert*, 53–68.

S. Kambhampati (1989), *Flexible Reuse And Modification in Hierarchical Planning: A Validation Structure Based Approach*, Technical Report CS-TR-2334, University of Maryland, USA.

R. Korf (1987), Planning as Search: A Quantitative Approach. *Artificial Intelligence*, 33(1): 65–88.

N. Kushmerick, S. Hanks, and D. Weld (1995). An Algorithm for Probabilistic Planning. *Artificial Intelligence*, 76(1–2): 239–286.

J. Lowrance and D. Wilkins (1990), Plan Evaluation under Uncertainty. In *Proceedings of the Workshop on Innovative Approaches to Planning Scheduling and Control*, California.

J. Lowrance, T. Strat, L. Wesley, T. Garvey, E. Ruspini and D. Wilkins (1991), *The Theory, Implementation and Practice of Evidential Reasoning*. Technical report, Artificial Intelligence Centre, SRI International, California.

T. M. Mansell (1993a), A Method for Planning Given Uncertain and Incomplete Information. *The Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, Washington DC, 350–358.

T. M. Mansell (1993b), Air Combat Planning Using U-Plan for Post Mission Analysis. In *Proceedings of the First Australian and New Zealand Conference on Intelligent Information Systems*, Perth, Australia, 644–648.

T. M. Mansell (1994a), Quantitative Operator Selection for Planning Under Uncertainty. In *The Second International Conference on Artificial Intelligence Planning Systems*, Chicago, 311–316.

T. M. Mansell and G. Smith (1994b), Operator Selection while Planning under Uncertainty. *The Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Seatlle, Washington.

T. M. Mansell (1994c), *An Approach to Planning Given Uncertain and Incomplete Information*. PhD Thesis, University of Melbourne, Australia.

T. M. Mansell and G. Smith (1994d), A Quantitative Search Strategy in Hierarchical Planning (in preparation).

T. M. Mansell, E. A. Sonenberg, and G. Smith (1995), AI Planning for Hazard Action Response. *The Proceedings of the Eighth International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems,* Melbourne, Australia, 705-710.

J. von Neumann and O. Morgenstern (1947). *Theory of Games and Economic Behavior.* 2nd ed. Princeton University Press, Princeton, New Jersey, USA.

N. J. Nilsson (1980), *Principles of Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., California.

OASG (1977), *Naval Operations Analysis.* Prepared by the Operations Analysis Study Group, Naval Institute Press, Annapolis, Maryland, USA.

J. Pearl (1984), *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Publishing Company, California.

J. Pearl (1988), *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publishers, Inc., California.

E. P. D. Pednault (1988), Synthesizing Plans that Contain Actions with Context-Dependent Effects. *Computational Intelligence,* 4:4 356–372.

M. Poet and D. Smith (1992), Conditional Nonlinear Planning. In *Proceedings of the First International Conference on AI Planning Systems,* Maryland, 189–197.

P. Raulefs, B. D'Ambrosio, M. R. Fehling and S. Forrest (1987), Real-Time Process Management for Materials Composition in Chemical Manufacturing. In *Proceedings of the IEEE Conference on Applications of Artificial Intelligence,* 120–125.

E. Sacerdoti (1974), Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence,* 5:115–135.

E. Sacerdoti (1977), *A Structure for Plans and Behavior.* American Elsevier.

G. Shafer (1976), *Mathematical Theory of Evidence.* Princeton University Press, Princeton, New Jersey, USA.

D. Spivakovsky (1995), *A Short Course in Military Operational Research.* Course notes, held at AMRL-Fishermen's Bend, 16-26 October.

S. Vere (1983), Planning in Time: Windows and Duration for Activities and Goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 5:246–267.

M. Wellman (1990), *Formulation of Tradeoffs in Planning under Uncertainty.* Morgan Kaufmann Publishers, Inc., California.

D. Wilkins (1988), *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann, Los Altos, California.

D. Wilkins, K. L. Myers, J. D. Lowrance and L. P. Wesley (1994), Planning and Reacting in Uncertain and Dynamic Environments, to appear in *Journal of Experimental and Theoretical AI*, Vol. 7.

Appendix 1: The Air Combat Domain

The air combat domain is dynamic and often requires agents to act given uncertain and incomplete information. To generate a plan for aircraft operating in such a domain requires the consideration of a large number of plan strategies. The air combat domain presented in this report considers two agents: the *defender* aircraft (whose objective is to defend himself and a designated airspace) and the *aggressor* (who is invading the airspace controlled by the defender). While the various actions available to the agents in this domain are diverse, certain patterns in the structure of the domain are identified. For example, the planning process for the *defender* involves *strategy selection, strategy implementation, evaluation of attack strategy* and *target monitoring*. To operate in such an environment requires a sophisticated planning and reasoning system.

It is intended that U-Plan generate a suitable super-plan for the defender aircraft given only the information available to the aircraft at plan time. This super-plan should plan to acquire necessary information when appropriate, have a high likelihood of success, be applicable to as many possible worlds as is feasible, and work for the worlds that are most likely to be true. The super-plan produced by U-Plan is intended for use in post mission analysis of the defender aircraft, not as a real-time planning aid.[18]

The strategies available to U-Plan in the air combat domain involve defending airspace or an asset by either attacking and destroying the invading aircraft or turning it away. The *Attack* procedure is centred around the selection of either a Beyond-Visual-Range Attack (*BVR_Attack*) or a Visual-Range Attack (*VR_Attack*). The different ways such strategies can be carried out are numerous, and each one involves a unique course of action. Figure 13 illustrates the types of strategies available to the defender aircraft. The figure shows part of the strategy hierarchy for the air combat domain that represents a continuous reduction of the goal, from the most abstract down to the manoeuvres that achieve these strategies. When producing a plan, U-Plan selects which strategy of these to apply, and implements them in the order demonstrated in the strategy hierarchy. For example, given the goal of defending a specific asset, U-Plan may choose to *Attack* the invading aircraft using a *BVR_Attack* in the form of a *Cutoff_Intercept* manoeuvre.

The implementation of this domain centres on the application of a *BVR_Attack* on the aggressor aircraft. The *VR_Attack* is included as an option primarily to cover all possible eventualities, but is not at the core of this domain.[19] The *VR_Attack* uses a few manoeuvres designed to incorporate a broader range of strategies that may be applied given certain characteristic situations. The same goes for manoeuvres associated with the *Turn_Away* strategy.

---

[18]U-Plan is not running in real-time and so is best applied in post mission analysis.

[19]The visual range attack domain is vast, complex and extremely dynamic, and is therefore not covered in great detail or accuracy in this air combat domain.

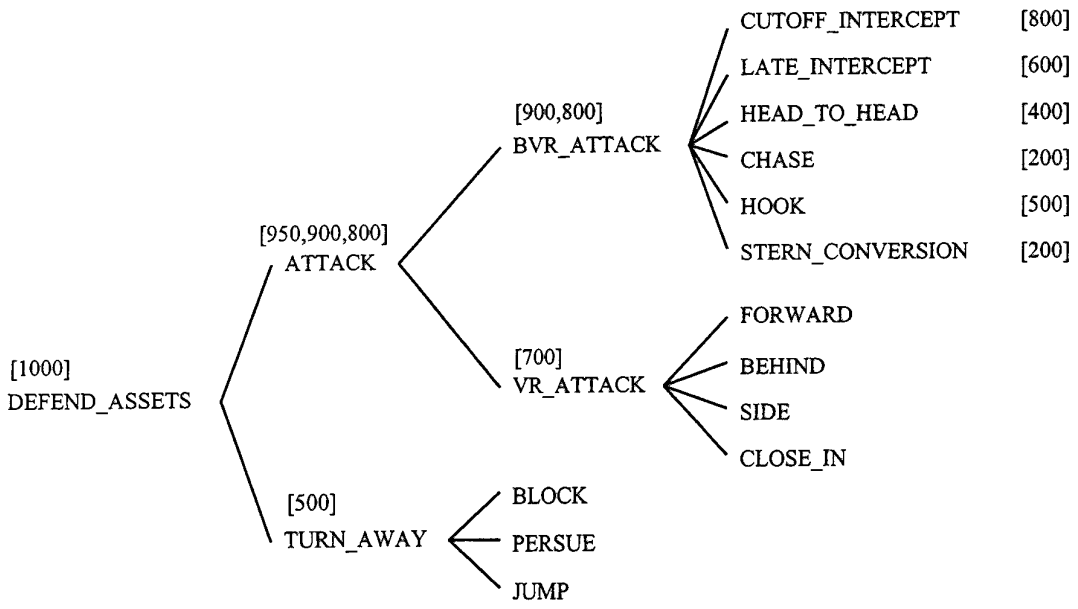

Figure 13: The strategic portion of the air combat abstraction hierarchy.

The manoeuvres named in figure 13 (represented at the leaf nodes of fig. 13) are each described at a lower level by a sequence of tactical actions. For example, the *Cutoff_Intercept* manoeuvre (fig. 13 and fig. 14) involves the application of a number of actions with varying degrees of abstraction. This includes *Set_Altitude* (a lowest level action that changes the state of the world) and *Obtain_Separation* (a tactical action that is achieved by applying the actions of *Set_Bearing* and *Set_Heading*). The *Cutoff_Intercept* manoeuvre is typical of the types of actions required to achieve any of the other manoeuvres.

The manoeuvres represented in this implementation of an air combat domain are loosely based on typical strategies developed by air forces around the world. The way in which they are implemented in this report has seen some standard manoeuvres modified, simplified or invented to suit the investigation of U-Plan. They are not meant to represent the prescribed actions of any particular organisation or be strictly accurate or correct.

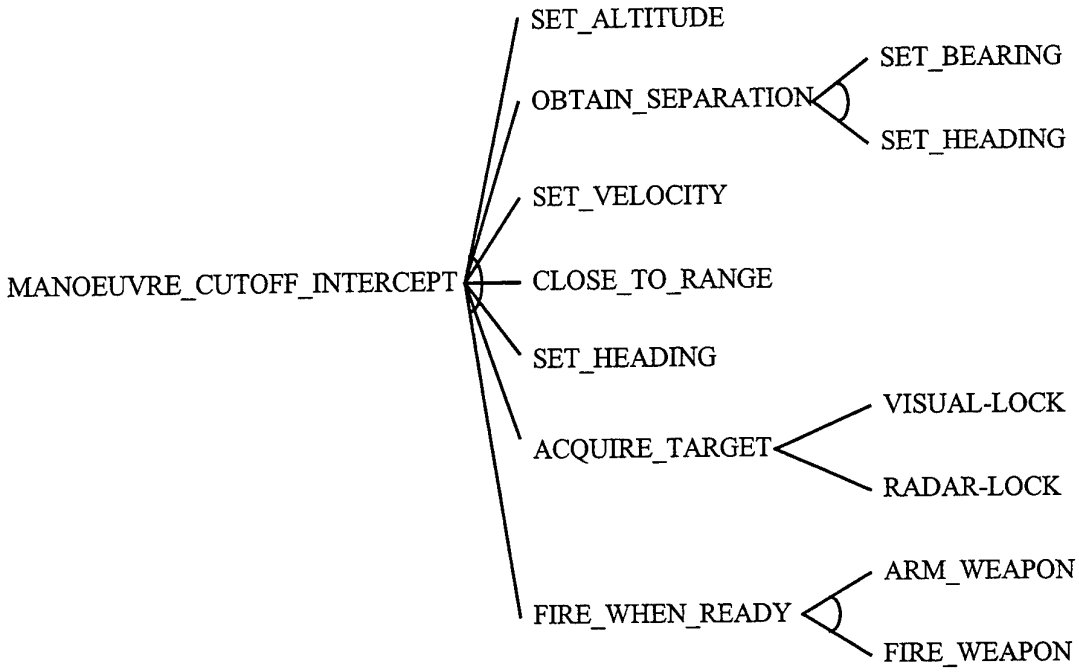

***Figure 14**: This diagram represents the breakdown of the actions required to achieve a Cutoff_Intercept manoeuvre.*

Appendix 2: Submarine Manoeuvre Planning Domain

The process of planning submarine manoeuvres is dynamic and often requires agents to act given uncertain and incomplete information. To generate a plan for a submarine operating in such a domain requires the consideration of a large number of plan strategies. The submarine manoeuvre planning domain presented in this appendix considers two agents: the diesel-electric *submarine* (whose objective is to defend himself while engaging a convoy) and the *Convoy* (who is traversing a region patrolled by the submarine). While the various actions available to the agents in this domain are diverse, certain patterns in the structure of the domain are identified. For example, the planning process for the *submarine* involves *strategy selection, strategy implementation, evaluation of attack strategy* and *target monitoring*. To operate in such an environment requires a sophisticated planning and reasoning system.

It is intended that U-Plan generate a suitable super-plan for the submarine given only the information available to the submarine at plan time. This super-plan should plan to acquire necessary information when appropriate, have a high likelihood of success, be applicable to as many possible worlds as is feasible, and work for the worlds that are most likely to be true. The super-plan produced by U-Plan is intended to provide real-time manoeuvre advice to the submarine's command.[20]

The strategies available to U-Plan in the submarine manoeuvre planning domain involve engaging a convoy within intercept range by either attacking and destroying the high-value-target or waiting to see if the option to attack improves. The *Engage* procedure is centred around the selection of either an *Intercept, Wait* or *Retreat* strategy. The different ways such strategies can be carried out are numerous, and each one involves a unique course of action (see figure 15 for a description of the type of intercepts represented in this scenario). Figure 16 shows part of the strategy hierarchy for the submarine manoeuvre planning domain that represents a continuous reduction of the goal, from the most abstract down to the manoeuvres that achieve these strategies. When producing a plan, U-Plan selects which strategy of these to apply, and implements them in the order demonstrated in the strategy hierarchy. For example, given the goal of destroying the high value asset (HVA), U-Plan may choose to *Intercept* the convoy using a *Torpedo_Attack* in the form of an *Ambush* manoeuvre.

The implementation of this domain centres on the application of an *Intercept* on the convoy. The *Wait* strategy is a special case of the intercept strategy, where the submarine can not (or will not) risk manoeuvring into an attack position. The *Retreat* is included as an option primarily to cover all possible eventualities, but is not at the core of this domain. The *Wait* strategy uses a few manoeuvres designed to incorporate a

[20]U-Plan does not currently run in real-time, but with appropriate development, could produce a plan of action in a timely enough manner for submarine operations (i.e. less than approximately 30 seconds).

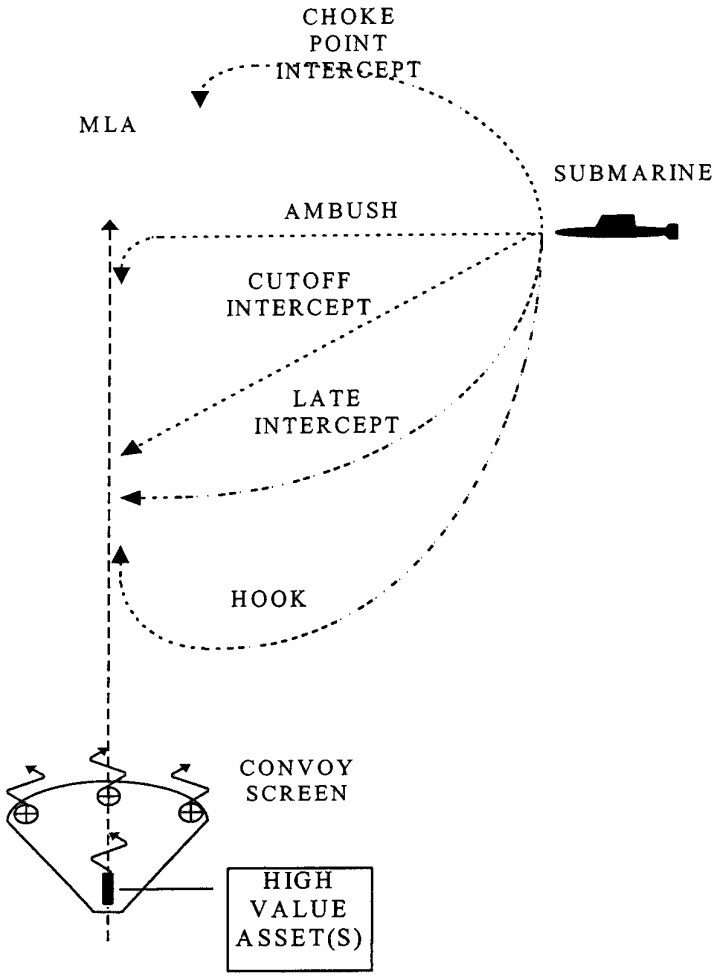

Figure 15: This figure pictorially describes the strategies that may be used by a submarine to intercept a convoy.

broader range of strategies that may be applied given certain characteristic situations. The same goes for manoeuvres associated with the *Retreat* strategy.

The manoeuvres named in figure 16 (represented at the leaf nodes of fig. 16) are each described at a lower level by a sequence of tactical actions. For example, the *Cutoff_Intercept* manoeuvre (fig. 16 and fig. 17) involves the application of a number of actions with varying degrees of abstraction. This includes *Set_Depth* (a lowest level action that changes the state of the world) and *Obtain_Separation* (a tactical action that is achieved by applying the actions of *Set_Bearing* and *Set_Heading*). The *Cutoff_Intercept* manoeuvre is typical of the types of actions included in the other manoeuvres.

The manoeuvres represented in this implementation of a submarine manoeuvre planning domain are loosely based on strategies described in operational research texts

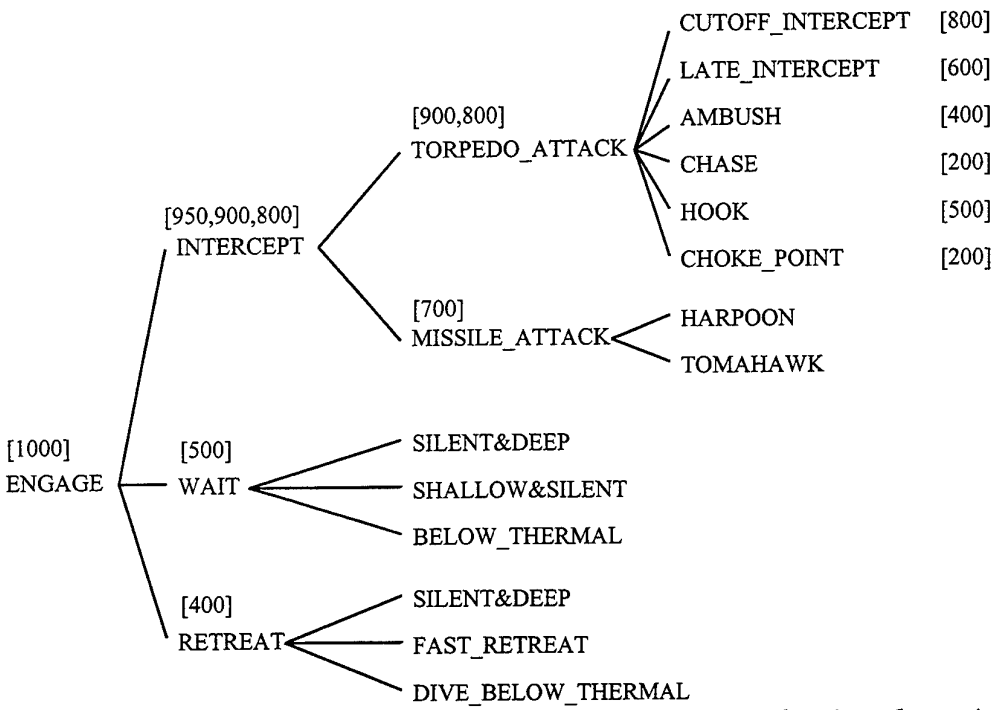

Figure 16: The strategic portion of the submarine manoeuvre planning abstraction hierarchy.

(OASG, 1977, and Spivakovsky, 1995). The way in which they are implemented in this report has seen some standard manoeuvres modified, simplified or invented to suit the investigation of U-Plan. They are for illustrative purposes only and do not represent the prescribed actions of any particular organisation nor are they strictly accurate or correct.

State representation

U-Plan assumes that an incomplete model of the world is all that is available, and uses a set of initial possible states (P-states) to describe what might be true of the world (discussed in detail in section 2). A P-state is a complete description of one possible world using propositional statements. The type of information required to represent the state of the world in the submarine manoeuvre planning domain is:

- convoy's mean line of advance (MLA) heading,
- convoy's MLA velocity,
- number of high value assets,
- number of screening ships,
- type of screening ships,

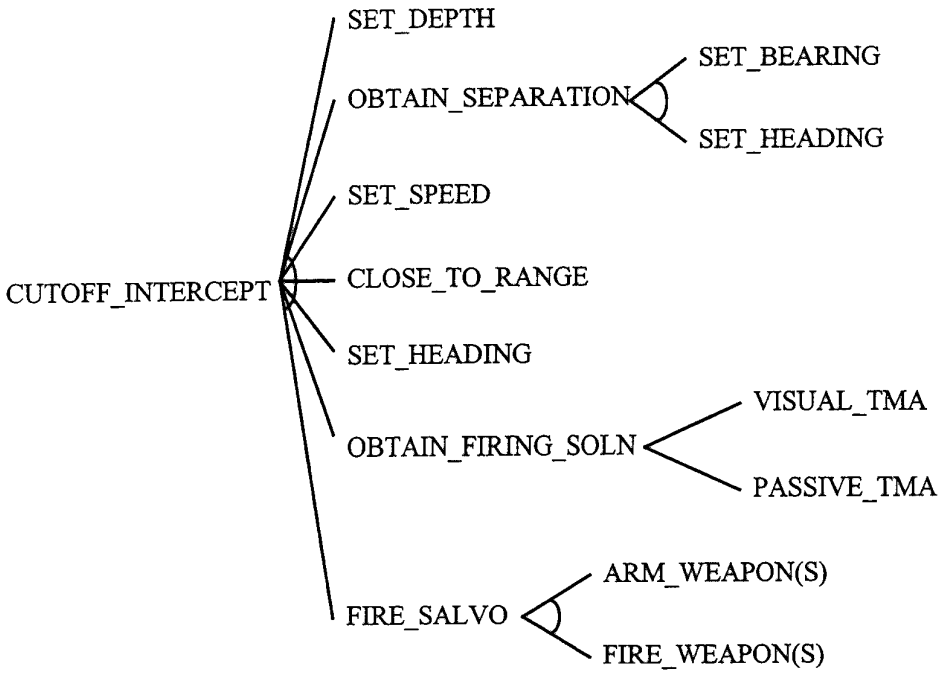

Figure 17: This diagram represents the breakdown of the actions required to achieve a Cutoff_Intercept manoeuvre.

- additional ASW assets (eg, LAMPS helicopters, maritime patrol aircraft (MPA), attack submarines, etc.),
- own ship kinematics (velocity, heading, self noise, etc.),
- weapons characteristics,
- environmental conditions (background noise level, etc.),

This information is used to generate a sufficient description of the environment in which planning will take place. U-Plan requires this information to determine when an action or strategy is appropriate, and the probability of successfully completing that action. Some of this information is assumed to be known with certainty (i.e., own ship velocity, heading, self noise levels, background noise levels, location of thermals, etc), while some of this information may be imprecisely known. Preliminary work suggests that the imprecise information may be hierarchically arranged into two levels abstraction, level 1 representing information at a strategic level, and level 2 providing a tactical description of the environment. Figure 18 demonstrates the amount of detail contained in the description of each level of abstraction. The predicates are represented by an attribute (ie, Range, Heading, ASW_Threat, etc), the abstraction level (ie, 1 or 2), and a value (of the form number, range, or name).

LEVEL 1 ATOMIC FODs

Θ_{No_HVA1} = {small, medium, high, unknown}

Θ_{No_ASW1} = {small, medium, high, unknown}

Θ_{Head1} = {known, unknown}

Θ_{MLA_Vel1} = {slow, medium, fast, unknown}

Θ_{Range1} = {within_intercept, beyond_intercept, unknown}

$\Theta_{ASW\_Threat1}$ = {low, medium, high, unknown}

Θ_{ASW_Type1} = {Helicopter, Land_based_Air, submarine, unknown}

LEVEL 2 ATOMIC FODs

Θ_{No_HVA2} = {0, 1, 2, 3, 4+, unknown}

Θ_{No_ASW2} = {0, 1, 2, 3, 4, 5, 6, 7, 8+, unknown}

Θ_{Head2} = {0, 45, 90, 135, 180, 225, 270, 315, unknown}

Θ_{Vel2} = {1-5, 5-7, 8-10, 11-15, 16-20, 20+, unknown}

Θ_{Int_Range2} = {1-5, 5-7, 8-10, 11-15, 16-20, 20+, unknown}

Θ_{Type2} = {Kirov, Kara, Kresta, Kashin, Sovremenny, Udaloy, Krivak, unknown}

Θ_{Air_ASW2} = {LAMPS, MPA, unknown}

Θ_{Sea_ASW1} = {SSN, COOP, unknown}

Figure 18: Demonstrates the type of imprecise information used by U-Plan in the submarine manoeuvre planning domain.

For example, the level 1 description of the ASW threat ($\Theta_{ASW\_Threat1}$) can be described as low, medium, high, or unknown. This description will be used to make the high level decisions like whether to attack and what general strategy to use. When it comes to planning the specific manoeuvres to be used, the more detailed, level 2, description of the world is used. The level 2 description provides information about the exact type of ASW threats out there, including ASW screen ship type (ie, Kirov, Kara, Kresta, Kashin, Sovremenny, Udaloy, Krivak, unknown), air ASW threats (ie, LAMPS, MPA, unknown), and other ASW threats (ie, SSN, COOP, unknown).

Reduction Operators

Planning operators represent actions that the system may perform in the given domain. The role of an action is to change the state of the world; the aim of an operator is to represent how applying that action will change the system's view of the state of the world. U-Plan's reduction operators give alternative methods for achieving

the goal at a lower level of abstraction, or at the tactical level it describes the direct effects of an action on the P-state (section 3).

The submarine manoeuvre planning domain currently requires the knowledge engineering of the set of reduction operators. Mansell (1994c) presents a methodology for the knowledge engineering of an entire U-Plan application domain, which should take place with active feedback from a domain expert. What follows is a summary of the issues that will be considered by the knowledge engineer in filling the reduction operator's slots:

- Name: The name should clearly and unambiguously describe the action it represents.
- Abstraction: This will depend on the level of detail contained in the description of the world (P-state) that is required to determine probability of success and preconditions (ie, level 1 or 2).
- Necessary preconditions: What must be true of the world before the operator is applicable.
- Satisfiable preconditions: What we must make true in the world before the operator can be selected.
- Plot: The set of reduction operators that achieve the goals of this reduction operator.
- Probability: The probability of successfully executing this reduction operator in the current environment. These probabilities will be conditioned on the number and type of ASW threats in the convoy's screen. They will also depend on the probability of ASW forces detecting the submarine. The contents of this slot may be obtained from simulations or operations research studies.
- Postconditions: This describes what changes the operator should make to world.
- Planfail: What to do if the reduction operator fails during planning.

It is typically very difficult to accurately describe the interactions that constitute the probability and preconditions slots.

Knowledge Acquisition

The submarine manoeuvre planning domain will typically require the production of a course of action given an uncertain and incomplete description of the world. In order for U-Plan to generate an unambiguous super-plan, the system must be able to reason about when and where to acquire additional (mission critical) information. In this domain, the process of acquiring additional information will be a tradeoff between maintaining stealth, battery requirements, time, and safety.

For example, target motion analysis (TMA) can be performed by using passive sonar and an appropriate manoeuvre (for triangulation), or from the periscope. Passive sonar TMA is less accurate (than periscope TMA) and may take substantially more time, which is a drain on the battery (due to manoeuvring). While periscope TMA may also provide precise classification information, it is at the cost of greatly increasing the possibility of detection (as the periscope must penetrate the surface). In addition, the convoy may be moving away from the submarine, hence there may not be time to perform passive sonar TMA.

Other mission critical information that may be acquired during plan execution is target classification, number of screen ships, number of HVAs, and thermal layer calculation.

Reasoning with the Physical Environment

U-Plan uses demon processes to reason about the effect of actions in the physical environment. These processes require reasonably detailed physical models to accurately describe (and therefore predict) the motion of surface and sub-surface vessels. These processes have not been implemented for the submarine manoeuvre planning scenario, as the work described in this appendix is in its infancy.

Discussion and Recommendations

The submarine manoeuvre planning scenario described in this appendix is well suited to U-Plan's approach to planning. However, further development is required before a concept demonstration system can be produced. In particular, the following development must take place.

- U-Plan must be ported from Symbolics LISP to CLIM. It is anticipated that this would take approximately 150 hours to complete by an experienced CLIM programmer. This would include a basic graphical user interface.

- Knowledge engineering of the submarine manoeuvre planning domain must be completed. The complexity of this process depends on the requested fidelity of the operators. A concept demonstration system using generic operator probabilities of success could be produced in less than approximately 100 hours.

DISTRIBUTION LIST

U-Plan: An Approach to Planning Given Uncertain and Incomplete Information

T. M.Mansell

AUSTRALIA

DEFENCE ORGANISATION

S&T Program

Chief Defence Scientist
FAS Science Policy
AS Science Corporate Management
} shared copy

Director General Science Policy Development
Counsellor Defence Science, London (Doc Data Sheet)
Counsellor Defence Science, Washington (Doc Data Sheet)
Scientific Adviser to MRDC Thailand (Doc Data Sheet)
Director General Scientific Advisers and Trials/Scientific Adviser Policy and Command (shared copy)
Navy Scientific Adviser
Scientific Adviser - Army (Doc Data Sheet and distribution list only)
Air Force Scientific Adviser
Director Trials

Director Aeronautical and Maritime Research Laboratory

Chief of Maritime Operations Division
Dr Ross Barrett (MOD)
Dr Alan Steele (MOD)
Dr David Liebing (MOD)
Mr Graeme Manzie (MOD)
Dr Todd Mansell (MOD) (2 copies)
Dr Stephen Hood (ITD)
Mr Conn Copas (ITD)
Dr Jennie Clothier (ITD)

DSTO Library

Library Fishermens Bend
Library Maribyrnong
Library Salisbury (2 copies)
Australian Archives
Library, MOD, Pyrmont
Library, MOD, HMAS Stirling

Capability Development Division

Director General Maritime Development
Director General Land Development (Doc Data Sheet only)
Director General C3I Development (Doc Data Sheet only)

Navy

SO (Science), Director of Naval Warfare, Maritime Headquarters Annex, Garden Island, NSW 2000 (Doc Data Sheet only)

Army

ABCA Office, G-1-34, Russell Offices, Canberra (4 copies)

Intelligence Program

Defence Intelligence Organisation
Library, Defence Signals Directorate (Doc Data Sheet only)

Corporate Support Program (libraries)

OIC TRS, Defence Regional Library, Canberra
Officer in Charge, Document Exchange Centre (DEC), 1 copy
*US Defence Technical Information Centre, 2 copies
*UK Defence Research Information Center, 2 copies
*Canada Defence Scientific Information Service, 1 copy
*NZ Defence Information Centre, 1 copy
National Library of Australia, 1 copy

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy
Library
Head of Aerospace and Mechanical Engineering
Deakin University, Serials Section (M list), Deakin University Library
Senior Librarian, Hargrave Library, Monash University
Librarian, Flinders University

OTHER ORGANISATIONS

NASA (Canberra)
AGPS

OUTSIDE AUSTRALIA

ABSTRACTING AND INFORMATION ORGANISATIONS

INSPEC: Acquisitions Section Institution of Electrical Engineers
Engineering Societies Library, US
Documents Librarian, The Center for Research Libraries, US

INFORMATION EXCHANGE AGREEMENT PARTNERS

Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US

SPARES (10 copies)

Total number of copies: 61

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

U-Plan: An Approach to Planning Given Uncertain and Incomplete Information

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

T. M. Mansell

5. CORPORATE AUTHOR

Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Vic 3001

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-RR-0103	AR-010-205	Research Report	April 1997

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
510/207/0586	DST 95/101	DSTO	67	44

13. DOWNGRADING/DELIMITING INSTRUCTIONS

None

14. RELEASE AUTHORITY

Chief, Maritime Operations Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT

No limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

uncertainty, decision making, operational effectiveness, mission planning

19. ABSTRACT

This paper describes research into planning in an uncertain environment. In particular, it describes U-Plan, a planning system that constructs quantitatively ranked plans given an incomplete description of the state of the world. Information acquisition operators are then applied to choose between plan alternatives. U-Plan has been trialled in a simulated one-on-one air combat domain. Results of the evaluation in this report included number of plans produced, sensitivity and timing data.