Representing Uncertainty in Simple Planners

Robert P. Goldman
Honeywell Technology Center
MN 65-2200
3660 Technology Drive
Minneapolis, MN 55418
goldman@src.honeywell.com

Mark S. Boddy
Honeywell Technology Center
MN 65-2200
3660 Technology Drive
Minneapolis, MN 55418
boddy@src.honeywell.com

Abstract

In this paper, we present an analysis of planning with uncertain information regarding both the state of the world and the effects of actions using a Strips- or (propositional) Adl-style representation [4, 17]. We provide formal definitions of plans under incomplete information and conditional plans, and describe Plinth, a conditional linear planner based on these definitions. We also clarify the definition of the term "conditional action," which has been variously used to denote actions with context-dependent effects and actions with uncertain outcomes. We show that the latter can, in theory, be viewed as a special case of the former but that to do so requires one to sacrifice the simple, single-model representation for one which can distinguish between a proposition and beliefs about that proposition.

1 INTRODUCTION

In this paper, we present an analysis of planning with uncertain information regarding both the state of the world and the effects of actions. Our focus on planning leads us to limit the expressive and inferential power of the formal systems we consider in the interests of efficiency. We do not develop with a full theory of knowledge and action of the sort which has concerned, e.g. Moore [13], Konolige [7], Haas [6], or Morgenstern [14]. In particular, we do not address what Morgenstern calls *knowledge preconditions* for actions and plans: determining when one knows enough to perform an action or successfully execute a plan, respectively. We are concerned only with taming uncertainty about the results of actions or uncertainty about the state of the world in the process of planning with a Strips- or (propositional) Adl-style representation [4, 17]. For the sake of computational efficiency, we restrict ourselves to a single model of the world, representing the planner's state of knowledge, rather than a more complex formalization including both epistemic and ground formulas. Our goal in this investigation is an increased understanding of *conditional plans*: plans in which the course of events is dictated by the actual outcome of actions whose effects cannot be predicted *a priori* [19, 5].

We clarify the definition of the term "conditional action." This term has been variously used to denote actions with context-dependent effects and actions with uncertain outcomes. We show that the latter can, in theory, be viewed as a special case of the former. However, we also show that to do so requires one to sacrifice the simple, single-model representation for one which can distinguish between a proposition and beliefs about that proposition. Using this definition, we provide formal definitions of plans under incomplete information and conditional plans.

Section 2 provides some preliminary definitions used in the rest of the paper. Section 3 presents an extension of Strips-rule planning to handle cases where the planner has only a partial model of the initial situation, and in which (a restricted kind of) information may be gained and lost through actions. We extend the Strips add and delete lists to include three truth values of true, false and unknown, losing in the process the use of negation-as-failure over the propositions that hold in a given situation. The primary limitation of this framework is that it assumes that the planner has sufficient knowledge to completely predict the outcomes of its actions — although one effect of these actions may be to forfeit information. There is no provision for actions with unpredictable outcomes. In Section 5 we remedy this omission, building on the conditional planning work of Peot and Smith.

Following this, we contrast conditional actions with context-dependent actions (Section 7). We show some problems with using context-dependent actions for planning under conditions of partial information. Finally, we compare our work with other work in the area, and present a summary and conclusions.

2 PRELIMINARY DEFINITIONS

We start with a set of *propositions*, $\{P_i\} = \mathcal{P}$. We define a *model*, $\mathcal{M}$ of $\mathcal{P}$, as a triple, $\langle T, F, U\rangle$ which is a partition of the propositions of $\mathcal{P}$ into statements which are true, false and unknown, respectively. A *partial model* is a triple $\langle T, F, U\rangle$ where T, F and U are disjoint subsets of $\mathcal{P}$. A model $\mathcal{M} = \langle T_{\mathcal{M}}, F_{\mathcal{M}}, U_{\mathcal{M}}\rangle$ is an *extension* of a partial model $\pi = \langle T_\pi, F_\pi, U_\pi\rangle$ if $T_\pi \subseteq T_{\mathcal{M}}$, $F_\pi \subset F_{\mathcal{M}}$ and $U_\pi \subseteq U_{\mathcal{M}}$. In the interests of brevity, we speak of a model $\mathcal{M}$ which extends a partial model π as satisfying or entailing π: $\mathcal{M} \vdash \pi$.

We define a set of operators, $\mathcal{O}$, similar to STRIPS operators. Operators are defined as ordered pairs $O = \langle P_O, E_O\rangle$, where P_O (the *preconditions* of O) and E_O (the *postconditions* of O) are both partial models. An operator O defines a partial function from models to models. The function is partial because f_O may only be applied to a model $\mathcal{M}$ which is an extension of P_O: the action represented by O may only be taken in a state where O's preconditions are satisfied.

One could extend the operators to be total functions from models to models by defining $f_O(\mathcal{M}) = \langle \emptyset, \emptyset, \mathcal{P}\rangle$ for any $\mathcal{M}$ that does not satisfy the preconditions of O. As there is little reason to plan to achieve complete ignorance, it makes more practical sense to insist that actions only be performed when their preconditions are satisfied.

We refer to models which entail P_O as satisfying the preconditions of O. For any operator O with effects $E_O = \langle T_E, F_E, U_E\rangle$ and model $\mathcal{M}$ satisfying the preconditions of O, the corresponding function, $f_O(\mathcal{M})$ is defined as follows:

$$f_o(\mathcal{M}) = \left\langle \begin{array}{l} (T_{\mathcal{M}} - (F_E \cup U_E)) \cup T_E, \\ (F_{\mathcal{M}} - (T_E \cup U_E)) \cup F_E, \\ (U_{\mathcal{M}} - (T_E \cup F_E)) \cup U_E \end{array} \right\rangle$$

This definition generalizes the STRIPS assumption to a three-valued logic. For a (possibly partial) model $\mathcal{M}$, let $\mathrm{true}(\mathcal{M}) = T_{\mathcal{M}}$, $\mathrm{false}(\mathcal{M}) = F_{\mathcal{M}}$ and $\mathrm{unk}(\mathcal{M}) = U_{\mathcal{M}}$. Additionally, for an operator $O = \langle P_o, \langle T_o, F_o, U_o\rangle\rangle$, let $\mathrm{precond}(O) = P_o$ (a partial model) and $\mathrm{affected}(O) = T_o \cup F_o \cup U_o$ (a set of propositions).

Note that the U sets represent lack of knowledge on the part of the agent *while the plan is being executed*, rather than the state of knowledge of the planner while it is constructing the plan. For example, a regression planner like McDermott's PEDESTAL [10] will not, in general, project the full state of the world after one of the steps of its plan is done. The planner cannot afford to compute this complete knowledge; it only commits to the truth value of propositions which are needed to ensure that its plan will be successful. However, PEDESTAL's completed plan *could* be used to project a series of complete truth assignments to the propositions describing the world. Once the plan is complete, all uncertainty has been banished. This is not the case in our framework.

3 THE SIMPLE PLANNING PROBLEM

A planning problem is a pair $\langle G, S\rangle$ where G is the *goal* of the planning problem and S is the *initial state*. The initial state S is a complete (i.e., not partial) model of $\mathcal{P}$, while the goal G is a partial model. The solution of a planning problem is a plan whose result satisfies G. In the following paragraphs we describe the result of a plan and how such a plan may be derived.

Following Lifschitz [9], we define a sequence of actions, or plan, as $\alpha = (\alpha_1, \ldots, \alpha_N)$. Assuming that this sequence of actions starts in initial situation $\mathcal{M}_0$, we say the plan is *accepted* by $\mathcal{M}_0$ iff there exists a sequence of models $\mathcal{M}_1 \ldots \mathcal{M}_N$ such that

$$\forall i, 1 \le i \le N, \quad \begin{array}{l} f_{\alpha_i}(\mathcal{M}_{i-1}) = \mathcal{M}_i \text{ and} \\ \mathcal{M}_{i-1} \vdash \mathrm{precond}(\alpha_i) \end{array}$$

We refer to $\mathcal{M}_N$ as the *result* of the plan α.

4 REGRESSION

Regression [24] can be described informally as reasoning backward from the desired effects of an action to what had to be true when the action was executed. One rationale for regression was the need to prove that plan steps used to achieve one goal would not clobber another goal of the same plan. Waldinger suggested a further use for regression: to determine where a step should be added to a linear plan. If a precondition of a new step is threatened by an existing step, put the affected step before the offender, and then *protect* that precondition (i.e., use regression to ensure that any further additions to the plan leave it unchanged). This is the way regression is employed in PEDESTAL [10].

In STRIPS-rule planners, regression only requires verifying that the proposition does not unify with the results of a given operator. Since those results are by definition the sole effect, known or unknown, of applying that operator, the proposition is unaffected by the action. Adding context-dependent effects as in ADL [17] complicates matters. Instead of just verifying that an operator will not affect a given proposition, regression involves deriving the conditions under which the operator will not affect that proposition.

Pednault[17] gives the conditions under which a proposition, p, will hold after the execution of an action a (for the propositional case) as follows:

$$I(a, p) = c_a \vee (p \wedge \neg d_a)$$

where c_A is the condition under which p is on a's add list and d_a is the condition under which p is deleted by

a. More precisely p will hold after act a is performed iff $I(a,p)$ holds before a is performed:

$$f_a(\mathcal{M}) \vdash p \equiv \mathcal{M} \vdash I(a,p)$$

Regression for the operators described here is somewhat more complex; our use of three truth values precludes use of the excluded middle, which can otherwise be used to good effect in simplifying the computation of causation and preservation preconditions. For the language described above, the regression operators are as follows (for p meaning p is true, $\overline{p}$ meaning p is false and $\tilde{p}$ meaning p is unknown):

$$\begin{aligned}
I(a,p) &\equiv (p \wedge p \notin \text{affected}(a)) \vee p \in \text{true}(a)\\
I(a,\overline{p}) &\equiv (\overline{p} \wedge p \notin \text{affected}(a)) \vee p \in \text{false}(a)\\
I(a,\tilde{p}) &\equiv (\tilde{p} \wedge p \notin \text{affected}(a)) \vee p \in \text{unk}(a)
\end{aligned}$$

The regression operator $I(a,p)$ may be interpreted as "p will be true following a," and similarly for the other truth values.

5 ACTIONS WITH UNCERTAIN OUTCOMES

In their paper on conditional non-linear planning [19], Peot and Smith extend the STRIPS model of actions to include actions whose outcomes are uncertain. They use these conditional actions to model observation actions. In this section, we extend our operator representation to include conditional actions and provide regression operators for them.

The operator semantics we have defined thus far is insufficient to model the use of observation to gain information. Information may be "gained" in a way analogous to the use of compliant motion for robots: operators may be selected in such a way as to reduce the set of unknown propositions whatever the initial state (e.g., ram into the wall as a way of reducing uncertainty in your position).[1]

The problem is that there is no way to describe an action with uncertain effects. Different effects must be the result of different operators or the same operator with context-dependent effects in different states. As long as the initial state S is a complete model, observations cannot be modelled in this way, because the resulting state is completely determined by the state in which the observation occurs.

Peot and Smith's operators are pairs $O = \langle P_O, O_O\rangle$. As before, P_O, the set of preconditions, is a partial model. O_O is a set of mutually exclusive and exhaustive possible outcomes when an action of type O is performed (i.e., exactly one of the outcomes will be the result of the action). Each outcome is a pair of the form $\langle \alpha_i, E_{\alpha_i}\rangle$. The α_i's are unique identifiers for outcomes. Let us define $\mathit{Olabels}(O) = \bigcup_i \alpha_i$ (for the sake of tidiness, we require the *Olabels* sets for different operators to be disjoint). If O is not a conditional operator, then $\mathit{Olabels}(O) = \{\perp\}$. $E_{\alpha_i} = \langle T_{O,\alpha_i}, F_{O,\alpha_i}, U_{O,\alpha_i}\rangle$ is a partial model describing the effects of O in the event of outcome α_i.

We define a new set of partial functions $f'_O : \mathit{Olabels}(O) \times \mathcal{M} \mapsto \mathcal{M}$ to describe the effects of the new operators. If an action of type O is performed in state $\mathcal{M} = \langle T_\mathcal{M}, F_\mathcal{M}, U_\mathcal{M}\rangle$ (which satisfies P_O) and outcome α_i occurs, the effect will be

$$\mathcal{M}' = \left\langle \begin{array}{l} (T_\mathcal{M} - (F_{O,\alpha_i} \cup U_{O,\alpha_i})) \cup T_{O,\alpha_i},\\ (F_\mathcal{M} - (T_{O,\alpha_i} \cup U_{O,\alpha_i})) \cup F_{O,\alpha_i},\\ (U_\mathcal{M} - (T_{O,\alpha_i} \cup F_{O,\alpha_i})) \cup U_{O,\alpha_i} \end{array} \right\rangle$$

That is, $f'_O(\alpha_i, \mathcal{M}) = \mathcal{M}'$. For actions that are not conditional, we define $f'_O(\perp, \mathcal{M}) = f_O(\mathcal{M})$. The (partial) functions $f'_O : \mathit{Olabel}(\mathrm{O}) \times \mathcal{M} \mapsto \mathcal{M}$ may be mapped straightforwardly to a (still partial) function $f : O \times \mathit{Olabels} \times \mathcal{M} \mapsto \mathcal{M}$, where $\mathit{Olabels} = \bigcup_{O\in\mathcal{O}} \mathit{Olabels}(O)$.

We extend our definition of action sequences, given in section 3 to sequences of action, outcome pairs. Similarly, we may extend our earlier definitions of regression operators to:

$$\begin{aligned}
I(a,\alpha,p) &\equiv (p \wedge p \notin \text{affected}(a,\alpha)) \vee p \in \text{true}(a,\alpha)\\
I(a,\alpha,\overline{p}) &\equiv (\overline{p} \wedge p \notin \text{affected}(a,\alpha)) \vee p \in \text{false}(a,\alpha)\\
I(a,\alpha,\tilde{p}) &\equiv (\tilde{p} \wedge p \notin \text{affected}(a,\alpha)) \vee p \in \text{unk}(a,\alpha)
\end{aligned}$$

The conditional actions permit us to describe observation operators. For example, we might want to have a planner which is able to find out what the weather is and plan travel accordingly:

listen to weather report

preconditions: at(home), unknown(storm)

postconditions:

Olabels	effects
α_1	storm
α_2	not(storm)

Note that it is the need to properly formalize observation operators which forces the third truth value on us. We must insist that unknown(storm) be a precondition of this operator in order that the planner not construct a pathological plan in which it keeps observing the weather over and over again until it gets an observation it likes.

Conditional operators may be used for uncertain actions other than observations. In domains in which actions are fallible, conditional operators allow us to build planners which can plan for contingencies in which actions fail to achieve desired effects. For example, in an image-processing domain, one may have a number of possible operations that can remove noise from an image. These operations are fallible; they do

[1] This is the way that BURIDAN handles uncertainty [8].

not always succeed in cleaning up the target image. If we describe these operations as conditional actions, we can build plans in which either the operation succeeds, or we must take other, additional actions to clean up the image.

This representation of conditional actions makes it cumbersome to represent and reason about effects that are common to all outcomes of a given operator. One might revise the representation to make this easier, but we do not believe it would repay the effort. In general, conditional actions will be introduced into a plan in the interest of varying outcomes. To take Peot and Smith's example, one might introduce an observation action to determine the state of a road between two points. One would be unlikely to do so simply to expend time (Peot [2] calls for conditional plans to be "without augury").

We view conditional action sequences as action sequences which branch forward in time. A *conditional sequence* is a tree, each of whose nodes corresponds to an action (an instance of an operator). Each edge in the tree will be labeled with an outcome label of the action at the tail of the edge.

More formally, a conditional sequence, $\mathcal{C} \subset \{(n, l, n') | n, n' \in \text{uid}(\mathcal{C}), l \in \mathit{Olabels}(\text{op}(n))\}$. uid($\mathcal{C}$) is a set of unique identifiers for acts in the conditional sequence $\mathcal{C}$. op is a function mapping uids of acts to the operator of which the act is an instance.

In order to be nonredundant, a conditional sequence must have only one triple (n, l, n') for every pair (n, l).

Unconditional sequences may be drawn from a conditional sequence. Each such unconditional sequence is a rooted path through the tree $\mathcal{C}$. That is, an unconditional sequence, $\mathcal{P}$ is written as $\mathcal{P} = (n_1, l_1) \ldots (n_N, l_N)$ where there must exist an edge $(n_i, l_i, n_{i+1}) \in \mathcal{C}$ and where $n_1 = \text{root}(\mathcal{C})$. In order to be well-formed, an unconditional sequence starting in an initial state $\mathcal{M}_0$ must meet the following conditions:

1. $f(\text{op}(n_i), \mathcal{M}_{i-1}, l_i) = \mathcal{M}_i$ for all $1 \leq i \leq N$.
2. $\mathcal{M}_{i-1} \vdash \text{precond}(\text{op}(n_i))$ for all $1 \leq i \leq N$.

We define $\text{result}(\mathcal{P}) = \mathcal{M}_N$. An unconditional sequence is *complete* if it is a path $\mathcal{P} = (n_1, l_1) \ldots (n_N, l_N)$ for $n_N \in \text{leaves}(\mathcal{C})$.

In order for a conditional sequence, $\mathcal{C}$ to be well-formed, it must be non-redundant and every unconditional path contained in $\mathcal{C}$ must be well-formed. In order to be complete, a conditional sequence must be well-formed and have an out-edge for every pair (n, l) for n an interior (non-leaf) node and l an element of $\mathit{Olabels}(\text{op}(n))$ where n corresponds to an instance of operator o. A conditional sequence $\mathcal{C}$ is a *complete conditional plan* for the planning problem $(\mathcal{M}_0, \mathcal{G}, \mathcal{O})$ if $\mathcal{C}$ is complete and for all complete paths $\mathcal{P} = (n_1, l_1) \ldots (n_N, l_N)$ in $\mathcal{C}$, $\text{result}(\mathcal{P}) \vdash \mathcal{G}$.

[2] Personal communication.

6 CNLP AND PLINTH

In this section we briefly discuss two recently-developed conditional planners. The first, CNLP, is a non-linear conditional planner for which conditional actions were developed; our original intent in this research was to clearly understand CNLP. In the course of our investigations, we came to suspect that the case for non-linear planning was less strong for conditional planning than conventional, "classical" planners. The second planner we discuss here, PLINTH, is a *linear* conditional planner which we have developed to investigate the efficiency tradeoffs between linear and nonlinear approaches to conditional planning.

Peot and Smith's CNLP [19] is a conditional, nonlinear planner based on Rosenblitt and MacAllester's SNLP. Like SNLP, CNLP constructs its plans by a series of interleaved goal satisfaction and threat removal operations. CNLP differs in adding conditional actions like those described here.

CNLP augments the SNLP algorithm to accomodate conditional actions. The labels of conditional outcomes are propagated along causal and conditioning links in the plan graph to record to which branch of the plan each action belongs. These conditioning links are added by a new threat-removal operation, "conditioning apart," which is used to assign steps to different contexts. Effectively one resolves a threat to a given step by adding a constraint that the threatener and the threatened step not both be part of the same branch of the plan. As of yet no results have been published concerning the soundness and completeness of CNLP.

Given the current prevalence and popularity of nonlinear planning, our decision to construct a linear conditional planner may require some explanation. In conventional, "classical" planning applications, nonlinear planning is usually an improvement over linear planning because fewer commitments yields a smaller search space, at a relatively minimal added cost to explore each element of that search space [12]. However, it is not clear that this tradeoff operates in the same way for conditional planners. When plans have multiple branches, the savings from considering fewer orderings is likely to be much less and may not repay the cost in the added complexity of individual plan expansion actions. In particular, the domain in which we have applied PLINTH is one in which subgoal interactions are minor, and thus in which a linear planner can be effectively employed. Conditional linear planning is simpler in conception as well as in implementation. In particular, our conditional linear planner can be shown to be sound and complete; we do not yet know of a sound and complete conditional non-linear planner. Finally, the operation which is needed to properly construct branching non-linear plans — resolving clobberers through conditioning apart — is a very difficult operation to direct.

PLINTH's conditional linear planning algorithm is nondeterministic and regressive.[3] Plans are built by manipulating three important data structures: a partial plan, a set of protections and a set of as-yet-unrealized goals. The planner operates by selecting an unrealized goal and nondeterministically choosing an operator to resolve that goal while respecting existing protections. New goals may be introduced when steps are introduced, either to satisfy preconditions or to plan for contingencies introduced by conditional actions. Essentially this algorithm is the same as that of a conventional linear planner. The crucial difference is in the effect of adding a conditional action to the plan.

When adding a conditional action, A, there will be some outcome, O, such that $A - O$ will establish the goal literal (otherwise A would not have been chosen for insertion). This outcome will establish what one can think of as the "main line" of the plan. However, there will also be some set of alternative outcomes, $\{O_i\}$. In order to derive a plan which is guaranteed to achieve the goal, one must find a set of actions which can be added to the plan such that the goals are achieved after $A - O_i$. for all i. This is done by adding new goal nodes to the plan, which are made successors of the other outcomes. The planner will now plan to realize these other goals as well as the "main-line" goal. Note that actions to handle alternative outcomes may be added either before or after the relevant conditional action. Loosely speaking, we can add to our conditional plans either remedial actions or precautionary actions.

PLINTH is described in greater detail elsewhere [5]. We have demonstrated that the algorithm is sound and complete. The algorithm has been implemented in Quintus Prolog. The implementation uses a depth-first iterative-deepening search strategy so it preserves the theoretical properties of soundness and completeness (up to hardware limitations). PLINTH is being applied to planning image processing operations for NASA's Earth Observing System, in collaboration with Nick Short, Jr. and Jacqueline LeMoigne-Stewart of NASA Goddard.

7 UNCERTAINTY AND SECONDARY PRECONDITIONS

In his work on ADL [15, 17], Pednault introduces operators which have context-dependent effects. This might seem to be an attractive method to represent problems of planning under incomplete information. For example, one attempt at representing an observation operator is the following:[4]

	Check the road from ?x to ?y.
	observe(clear(?x,?y))
preconditions:	at(?x), unknown(clear(?x,?y))
effects:	clear(?x,?y) ← clear(?x,?y)
	not clear(?x,?y) ← not clear(?x,?y)

For those not familiar with ADL: ADL operators are akin to STRIPS operators, but their preconditions are partitioned. STRIPS preconditions are conditions under which the operator can be performed. If the preconditions do not hold when the operator is to be performed, the plan is ill-formed. ADL departs from STRIPS in allowing one to attach additional conditions to the individual effects of operators. For example, one could formalize the action of crossing a bridge by truck as having the preconditions of being at the bridge and in a truck. There might be two possible effects: being on the other side of the bridge, which has as additional precondition that the bridge is sound; and being in the ravine below, which has as additional precondition that the bridge is not sound. We indicate additional preconditions using the "backward chaining arrow," ←. Note that the original ADL syntax uses add lists and delete lists; we use effects instead in the interests of a consistent notation within the body of this paper.

One problem with this representation is apparent in the observation operator given above. Each of the effects has itself as additional precondition! The reason for this paradoxical situation is that planning under uncertainty in this way violates an common planning assumption: that we can treat the planner's model of the world and the state of the world as interchangeable. To handle observations properly, we need to be able to represent and reason about both the state of the world (the road is clear, so when we look, we will find it so) and the planner's state of knowledge (the observation operator makes sense because the planner doesn't know whether the road is clear).

A related problem is that this formalization leads us to an unrealistic model for planning. The obvious way to use an operator like the above is to insert it into one's plan and then continue planning in two contexts: one where the road is clear and one where it is not. But note an undesirable feature of these two contexts: in each of them the road not only is clear, but has always been clear, and the planner should know this. As in the famous problem of Schrodinger's cat, performing the observation seems to cause the entire history of the world to change. Closer to home, this paradox is akin to McDermott's "little Nell" problem [11], in which planning to prevent an action seemed to make planning to prevent it unnecessary. Without explicit

[3] Our development of the algorithm was inspired by McDermott's linear planner PEDESTAL [10], hence the name.

[4] We have tried to follow Pednault's notation fairly closely.

representation of belief, ground truth and the relations between them, it is impossible to model the acquisition of information with only deterministic operators.

8 CASSANDRA

Cassandra is a conditional nonlinear planner that uses secondary preconditions for planning under uncertainty [20]. In order to encode uncertainty, certain actions are given secondary preconditions which are unknowable; Cassandra must plan to gain information about these unobservable pseudo-propositions. Reading about Cassandra bears out our conclusions above about the drawbacks of secondary preconditions for encoding uncertainty.

Cassandra is built on top of the nonlinear planner UCPOP [18]. UCPOP is sound and complete, and uses Pednault's ADL for its action representation. In Cassandra, rather than positing conditional actions like those described here, uncertain outcomes are captured by giving actions secondary preconditions which are "unknowable." These unknowable preconditions have multiple outcome labels, like our outcome labels, and like our outcome labels are mutually exclusive and exhaustive. Instead of branching at conditional actions, Cassandra plans branch at *decisions*. Decisions contain condition-action rules which specify the conditions under which the planner should conclude that a given outcome has occurred. When the plan is executed, the execution monitor should perform only those actions labeled consistently with the outcomes of its decisions. These decision rules provide a mechanism for relaxing an assumption common to both CNLP and PLINTH: that the outcomes of conditional actions are always known.

While we find Cassandra's model of uncertainty attractive, it appears to require more expressive power than its ADL action representation provides. It is consequently difficult to say exactly what Cassandra's plans mean. In particular, Cassandra's decisions and their knowledge preconditions cannot be expressed in ADL. Cassandra's decisions have preconditions of the form "knowif(*proposition*)," but such preconditions are beyond its expressive capacity; Cassandra, like the other planners described here, appears to make no distinction between truth in the world and the planner's beliefs about the world. Accordingly, there can be no satisfactory ADL representation for actions which collect information, as we have argued in the previous section.

9 INFORMATION-GATHERING ACTIONS

Recent work on planning under uncertainty done at the University of Washington has brought to the fore a number of issues concerning information-gathering actions [2]. In particular, the UW group has shown that planners must be able to distinguish goals of information-gathering from other goals of achievement. They provide the following persuasive example:

> Suppose that the planner is told that the hidden treasure it is seeking is located behind "the blue door." Painting a door blue does not satisfy the goal of finding "the blue door" — it merely obscures the identity of the appropriate door. [2, p. 116]

In order to capture the distinction between information-gathering goals and achievement goals, the UWL planning language provides goal annotations: **satisfy**, **hands-off** and **find-out**. Satisfy goals are to be achieved as normal planner goals. Hands-off goals, on the other hand, are restricted to information-gathering. If the planner has a goal annotated with **(hands-off** P**)**, it must achieve its other goals without affecting the truth value of P. Finally, **(find-out** P**)** goals are a hybrid — the planner should prefer to simply observe the truth or falsehood of P, but if the planner must change the value of P for some other reason, that is acceptable. The UW group has developed a conditional planner, SENSp, for UWL, in which the process of matching pre- and post-conditions is altered in order to handle these annotations.[5]

As far as we can tell, the **hands-off** and **find-out** goals are similar encodings of radically different phenomena. The **hands-off** goals are apparently only a special class of preservation goals. They appear different because conventional planning languages are not expressive enough to say "maintain the truth value of P, *whatever it may be now.*" and to permit observation of P without modification. Within our framework, we capture this restriction by ruling out the use of operators which would change the truth value of P over part or all of a given plan. We do so by ruling out all operators O such that

$$P \in \text{affected}(O) \wedge \tilde{P} \notin \text{precond}(O)$$

That is, any operator whose use affects the truth value of P with the exception of those operators that simply inform us whether or not P holds. That is the condition captured by the second conjunct of the condition above: observation operators are those that set (reveal) the value of P and that require that the value of P be unknown beforehand. Once the truth value of P has been determined (either given in the initial conditions or established by observation), we may use the conventional planning technique of protecting P, rather than the criterion above.

[5] In personal communication, Etzioni reports that two further planners based on the UWL language have been developed since this paper was drafted, both nonlinear and interleaving planning and execution.

The **find-out** goals, on the other hand, do not determine what constitutes a plan that satisfies the specified goals. Rather, they specify a preference over different, but equally valid, plans. This insight suggests that the somewhat cumbersome criterion for satisfaction of **find-out** goals [2, p. 119] might be removed, with the associated preference being expressed instead in the cost function over operators. In most cases, the cost of observations should be lower than the cost of achievement. There are three advantages to factoring this concern into the cost function: first, we avoid further complication of the planning problem; second, the cost information may more readily be used in planning search than the **find-out** criterion and third, the cost mechanism allows us to capture a wider range of tradeoffs.

Distinguishing these two annotations as different phenomena within our framework allows us to considerably simplify their treatment. hands-off goals can be enforced using a slight variation on protection assumptions or preservation preconditions, while find-out annotations are reflected in the planner's search control mechanism.

Note Recent exchanges reveal that in new versions of UWL the **find-out** annotation has been revised from "satisfy without altering if possible" to "satisfy without altering."[6] Our criticisms above apply only to the currently-available paper on UWL [2].

10 OTHER RELATED WORK

Early work on modeling knowledge in AI systems by Moore [13], Haas [6] and Konolige [7] provides a different view on modeling knowledge for planning systems. This early work was primarily concerned with modeling knowledge, rather than the development of planning algorithms. More recent work by Morgenstern [14] and Scherl and Levesque [22] brings such work much closer to the point of constructing working planners. However, these representations are still far more complex than those used by most working planners. In particular, they require the use of complex logical machinery (string manipulations or modal logics) in order to capture the distinction between beliefs and the state of the world. We have attempted to maintain the simplicity of existing approaches and, in particular, maintain the single model approach.

11 POSSIBLE EXTENSIONS

Our extension to the use of a third truth value is intended to model lack of knowledge about a proposition. Another use for three-valued logics has been to allow truth-functional treatment of statements which are meaningless [23], particularly the problem of predicating properties of inexistent objects. We may encounter a prosaic version of this problem in planning under uncertainty. For example, consider the problem of an oil-wildcatter[7] who has the option of taking a core sample before drilling. Imagine that we have three propositions describing mutually exclusive and exhaustive outcomes of such a test: (result os), (result cs), (result ns). What is the truth value of these statements in the initial situation? The truth value of this proposition is not well-defined because the proposition predicates a property of an inexistent object.

We suspect that related issues will arise in Etzioni's work on Unix Softbots [2, 3], which act within the Unix operating system. For example, what is the status of a predication about a file which has yet to be created? We note that Etzioni, et. al. have so far avoided constructing any operators which either create or destroy files. One way of addressing this problem would be to add a fourth "truth value" for propositions that are ill-formed in this way.

For some applications, allowing ADL-style operators with secondary preconditions may make planning more efficient. Such operators allow the planner to defer some commitments to precise methods for achieving goals, in the interests of allowing later reuse of operators for additional goals. We would like to retain this advantage, but doing so will require revision of Pednault's regression operators [16], since many of the identities he uses are not valid in three-valued logic.

12 SUMMARY AND CONCLUSIONS

We have provided a formal analysis of STRIPS-style planning under conditions of incomplete information and where the outcomes of actions are not known with certainty. We have also provided a precise definition of conditional plans. This work provides a unifying theoretical framework and vocabulary for a number of disparate conditional planners such as CNLP, SENSp and PLINTH. In the process of defining this framework, we have clarified the relationship between conditional actions and actions with context-dependent effects, and shown that the latter are not sufficient for modelling information-gathering actions (i.e., observations). We have shown that our analysis simplifies the treatment of information-gathering acts and goals.

Acknowledgements The authors would like to thank Alan D. Christiansen, Oren Etzioni and our two anonymous reviewers for helpful comments.

[6] Oren Etzioni, personal communication.

[7] A now-standard problem due to Raiffa [21].

References

[1] Allen, James, Hendler, James, and Tate, Austin, (Eds.), *Readings in Planning*, (Morgan Kaufmann Publishers, Inc., 1990).

[2] Etzioni, Oren, Hands, Steve, Weld, Daniel S., Draper, Denise, Lesh, Neal, and Williamson, Mike, An Approach to Planning with Incomplete Information, Nebel, Bernhard, Rich, Charles, and Swartout, William, (Eds.), *Principles of Knowledge Representation and Reasoning:Proceedings of the Third International Conference*, 1992, 115–125, Morgan Kaufmann Publishers, Inc.

[3] Etzioni, Oren and Segal, Richard, Softbots as Testbeds for Machine Learning, *Proceedings of the 1992 AAAI Spring Symposium on knowledge assimilation*, 1992.

[4] Fikes, Richard E. and Nilsson, Nils J., STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, **2** (1971) 189–208.

[5] Goldman, Robert P. and Boddy, Mark S., Conditional Linear Planning, *Artificial Intelligence Planning Systems: Proceedings of the Second International Conference*, 1994, Morgan Kaufmann Publishers, Inc., forthcoming.

[6] Haas, Andrew R., A Syntactic Theory of Belief and Action, *Artificial Intelligence*, **28** (1986) 245–292.

[7] Konolige, Kurt, A first-order formalisation of knowledge and action for a multi-agent planning system, Hayes, J.E. and Michie, D., (Eds.), *Machine Intelligence 10*, chapter 2, 41–72, (Halstead, New York, 1982).

[8] Kushmerick, Nicholas, Hanks, Steve, and Weld, Daniel, *An Algorithm for Probabilistic Planning*, Technical Report 93-06-03, Department of Computer Science and Engineering, University of Washington, June 1993.

[9] Lifschitz, Vladimir, On the Semantics of Strips, In Allen et al. [1], 523–530, Reprinted from *Reasoning about Actions and Plans*.

[10] McDermott, Drew, Regression Planning, *International Journal of Intelligent Systems*, **6**(4) (1991) 357–416.

[11] McDermott, Drew V., Planning and acting, *Cognitive Science*, **2** (1978) 71–109.

[12] Minton, Steven, Bresina, John L., and Drummond, Mark, Commitment Strategies in Planning: A Comparative Analysis, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., 1991.

[13] Moore, Robert, A Formal Thoery of Knowledge and Action, Hobbs, J., (Ed.), *Formal Theories of the Commonsense World*, (Ablex, Hillsdale, N.J., 1984).

[14] Morgenstern, Leora, Knowledge Preconditions for Actions and Plans, McDermott, John, (Ed.), *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, 1987, 867–874, Morgan Kaufmann Publishers, Inc.

[15] Pednault, Edwin P.D., Extending Conventional Planning Techniques to Handle Actions with Context-dependent effects, *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988, 55–59, Morgan Kaufmann Publishers, Inc.

[16] Pednault, E.P.D., Synthesizing Plans that contain actions with context-dependent effects, *Computational Intelligence*, **4**(4) (1988) 356–372.

[17] Pednault, E.P.D., ADL: Exploring the middle ground between STRIPS and the situation calculus, *First International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers, Inc., 1989.

[18] Penberthy, J. Scott and Weld, Daniel S., UCPOP: A Sound, Complete, Partial Order Planner for ADL, Nebel, Bernhard, Rich, Charles, and Swartout, William, (Eds.), *Principles of Knowledge Representation and Reasoning:Proceedings of the Third International Conference*, 1992, 103–114, Morgan Kaufmann Publishers, Inc.

[19] Peot, Mark A. and Smith, David E., Conditional Nonlinear Planning, Hendler, James, (Ed.), *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, 1992, 189–197, Morgan Kaufmann Publishers, Inc.

[20] Pryor, Louise and Collins, Gregg, *Cassandra: Planning for Contingencies*, Technical Report 41, The Institute for the Learning Sciences, Northwestern University, June 1993.

[21] Raiffa, Howard, *Decision Analysis: Introductory Lectures on Choices under Uncertainty*, Behavioral Science: Quantitative Methods, (Random House, New York, 1968).

[22] Scherl, Richard B. and Levesque, Hector J., The Frame Problem and Knowledge-producing Actions, *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 1993, 689–695, AAAI Press/MIT Press.

[23] Turner, Raymond, *Logics for Artificial Intelligence*, (Ellis Horwood, Ltd., 1984).

[24] Waldinger, Richard, Achieving Several goals Simultaneously, Elcock, E. and Michie, D., (Eds.), *Machine Intelligence*, volume 8, 94–136, (Ellis Horwood, Edinburgh, Scotland, 1977).