Expressive Planning and Explicit Knowledge

Robert P. Goldman and **Mark S. Boddy**
{ goldman | boddy }@src.honeywell.com
Honeywell Technology Center, MN65-2200
3660 Technology Drive
Minneapolis, MN 55418

Appeared in the *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, Brian Drabble, ed., AAAI Press, 1996, pp. 110–117.

Abstract

We are concerned with the implications and interactions of three common expressive extensions to classical planning: conditional plans, context-dependent actions, and nondeterministic action outcomes. All of these extensions have appeared in recent work, sometimes in conjunction, but the semantics of the combination has not been fully explored. As we have argued in previous work, providing a coherent semantics for conditional planning with context-dependent actions requires that the planner's information state be modelled separately from the world state. In this paper, we present a new planning language, WCPL, encompassing these extensions. The semantics of WCPL includes an explicit treatment of the planner's information state as knowledge, as opposed to some form of context labelling. In addition to clarifying and unifying a disparate set of results from earlier work, we extend that work: WCPL handles both conditional and fail-safe plans for an action representation including both context-dependent and nondeterministic actions.

Introduction

The past few years have seen an increasing focus on the application of planners to "real world" domains. This focus has brought into sharp relief the fact that planning models and algorithms were largely unable to handle certain domain features that are quite common in practice, but had been simplified out of the vast majority of theoretical models. A variety of possible extensions to the classical model have been proposed or analyzed, much of it in recently published work. The issues addressed include context-dependent effects of actions (Pednault 1988; McDermott 1991; Penberthy & Weld 1992), duration and other metric constraints (Tate, Drabble, & Dalton 1994; Vere 1983), uncertain action effects and limited knowledge about the world (Kushmerick, Hanks, & Weld 1993; Etzioni *et al.* 1992; Pryor & Collins 1995), building conditional plans (Peot & Smith 1992; Goldman & Boddy 1994a; Warren 1976), and exogenous events (Blythe 1994), among others.

These extensions in expressive power interact in interesting and sometimes unexpected ways. In previous work, we have shown that constructing conditional plans including context-dependent actions makes untenable the assumption that the planner has perfect world knowledge (Goldman & Boddy 1994b). The intuition behind this result is that conditional plan branches must be conditioned *on* something, that being additional information available to the planner only at runtime. There are two ways to introduce the possibility of gaining this additional information. First, the domain may include nondeterministic actions (actions whose outcome is essentially a random variable), where the outcome that occurred can be determined after the fact. Alternatively, the planner may have only incomplete knowledge of the world state in any given situation, and actions may as a side effect add to that knowledge. Some previous work on conditional planning (e.g., (Peot & Smith 1992; Pryor & Collins 1995)) has finessed or ignored the fact that "observing" the current value of some predicate is different from performing an action whose outcome is only stochastically determined. [1]

In this paper, we explore the implications and interactions of three common expressive extensions to classical planning: conditional plans, context-dependent actions, and nondeterministic action outcomes. All of these extensions have appeared in recent work, sometimes in conjunction, but the semantics of the combination has not really been addressed, save in the probabilistic planner C-BURIDAN (Draper, Hanks, & Weld 1993). We present a new planning language, WCPL, providing a unified treatment and a coherent semantics for conditional planning with context-dependent and nondeterministic actions, including a principled treatment of the distinction between ground truth and the planner's information state.

Rather than represent the planner's information

[1] In particular, repeated observations of the same proposition should have the same result.

state in terms of a set of labelled contexts, as for example in (Peot & Smith 1992; Draper, Hanks, & Weld 1993), we provide an explicit semantics for the planner's knowledge as a modal operator over possible states of the world. Modelling the planner's information state in this way enables us to handle both information-gathering in conditional plans, and conformant or "fail-safe" plans, in which it can be shown by reasoning over possible outcomes, despite incomplete information, that the plan will achieve the specified goal(s). In fact, we can integrate these approaches in a way that the labelling approaches cannot: reasoning by cases can modify the planner's information state at runtime, which can then lead to conditional branching in the plan where there may have been no explicitly labelled "context" or "outcome."

This flexibility allows us to handle both versions of Moore's "bomb in the toilet" problem (McDermott 1987). The "bomb in the toilet" actually covers two substantially different problems. In the first version, the planner is confronted with two packages, one of which contains a bomb. There is no way to distinguish the explosive package from its harmless counterpart. There is also a toilet, immersion in which will render the bomb ineffectual. In this case, there is a simple and intuitive plan for ensuring that there will not be an explosion: immerse both packages. In the second version of the bomb in the toilet, only one package will fit in the toilet, but the planner has available a bomb-detecting action that will identify which package needs to be immersed to prevent an explosion. In this case a conditional plan is needed: find out where the bomb is, then immerse the appropriate package. Handling both of these problems requires a planning system to be able both to reason by cases, and to construct conditional plans based on the acquisition of new information.

In the rest of this paper, we present the syntax and semantics of WCPL's action representation ("Actions"), add a modality corresponding to the planner's knowledge of the current world state and define information-gathering actions ("Knowledge"), and add conditional branches to the planning language ("Conditional plans"). We then provide a detailed solution to the two versions of the bomb in the toilet problem. Finally, we discuss some related work, draw conclusions and describe some directions for further research.

For more convenient reference, we have moved all of the definitions and propositions to the end of the paper.

Actions

Previous work in this area, both ours and others', has used notations derived from STRIPS and ADL (Pednault 1988). In this paper, we find it more convenient to employ propositional dynamic logic (PDL). PDL provides us with previously existing constructs for nondeterminism and conditionals. Our discussion of PDL follows Rosenschein's (1981). To maintain the relation between this work and previous work as clearly as possible, we relate the PDL notation to STRIPS or ADL-style equivalents where possible.

Syntax

In propositional dynamic logic, we speak of programs made of actions and propositional wffs that describe world states. We use the term "programs," rather than plans, both to conform to PDL usage and to avoid confusion when we provide the PDL corresponding to a plan language like STRIPS. Given a set of atomic propositions, Φ, and a set of atomic actions, $\aleph$, we define the wffs, $\mathcal{P}$, and programs, $\mathcal{A}$, over Φ and $\aleph$ as follows:

1. if $\alpha \in \aleph$, then $\alpha \in \mathcal{A}$
2. if $\phi \in \Phi$, then $\phi \in \mathcal{P}$
3. if $P, Q \in \mathcal{P}$, then $\neg P, P \vee Q \in \mathcal{P}$
4. if $P \in \mathcal{P}$, and $A \in \mathcal{A}$, then $\langle A \rangle P \in \mathcal{P}$
5. $\epsilon \in \aleph$
6. if $P \in \mathcal{P}$ and P contains no sub-formula of the form $\langle A \rangle P$, then $P? \in \mathcal{A}$.
7. If $A, B \in \mathcal{A}$, then $A; B \in \mathcal{A}$
8. If $A, B \in \mathcal{A}$, then $A \cup B \in \mathcal{A}$

ϵ is the null action. We provide the conventional extensions of notation, e.g. $\wedge$ for conjunction, as syntactic sugar. We also provide a dual for the $\langle\rangle$ modality, $[]$ (see discussion of semantics, below). A literal is an atomic proposition or its negation—if $\phi \in \Phi$, then ϕ and $\neg\phi$ are literals.

PDL also employs Kleene star for (indefinite) repetition of actions. However, we do not use Kleene star in WCPL, so we have left it out of the discussion here.

Semantics

The semantics of PDL is defined in terms of structures, S, written (W, π, m). W is a non-empty set of world states. π is an interpretation of Φ, mapping each proposition to those states in W in which that proposition is satisfied. π is extended to compound wffs in the standard way. m is an interpretation of the singleton programs (those consisting of a single action), mapping each action α to a set of pairs of world states denoting permitted state transitions for α. The interpretation of programs is defined recursively, based on the relations in m (see Definition 1). '?' can be read as 'test,' $P?$ limiting the possible subsequent world states to the set of worlds where P holds. ';' denotes sequential composition of programs and $\circ$ denotes composition of relations.

$\langle A \rangle P$ is satisfied in those worlds w for which there exists another world w' and a program A such that P is satisfied in w' and $(w, w') \in m(A)$. Intuitively, P may be true after A. There is a dual of $\langle\rangle$, corresponding to necessity $[]$, where $[A]\, P$ may be read as P necessarily holds after A.

Plans in PDL

In this section, we discuss the PDL representation for plans, starting with the classical definition of an action, then expanding to actions with conditional and context-dependent outcomes. A classical STRIPS operator can be written as a set of preconditions, γ and a set of postconditions, δ, where both γ and δ are conjunctions of literals. In the classical definition, preconditions define not where the action will be successful, but where it can be applied. If the initial state is such that one or more steps in a given plan will have unsatisfied preconditions, the effects of that plan are undefined.

For mathematical convenience, we want m to be a complete relation. However, we also want to preserve the distinction between well-formed and ill-formed plans. In order to do this, we introduce a distinguished atomic proposition **planfail** $\notin \Phi$.

We require m to have the property captured in Definition 2. m is restricted to state transitions (s, t) in which either the preconditions (γ) is satisfied in s and δ is satisfied in t, or in which γ is not satisfied in s and **planfail** is satisfied in t. If γ is satisfied in s, the truth value for both **planfail** and any proposition not mentioned in δ must be the same in s and t. This definition preserves the STRIPS assumption and records in the sequence of worlds whether or not the plan's effects are well-defined as described above. It also follows from this definition that failure persists for any action, once it becomes true.

Planning is the process, given a description of an initial state, and a goal, of finding a plan that will achieve the goal when executed in the initial state. In the WCPL framework, the classical planning problem is posed as follows: given a goal G and an initial state description, D, find a program, A, such that:

$$D \rightarrow [A]\,G \qquad (1)$$

$$D \rightarrow [A]\,\neg\mathbf{planfail} \qquad (2)$$

The first condition enforces that the plan achieve the goal, the second that actions only be attempted when their preconditions hold. This framework is slightly more general than conventional classical planning in that it permits planning with only partial knowledge of the initial world state; the planner needs to know only the *relevant* aspects of the initial state. In practice, this is a common extension for classical planning systems.

The next step is to extend the definition above to context dependent actions. We start with the same representation as in Buridan (Kushmerick, Hanks, & Weld 1993), encoding context-dependent effects in a set of postconditions, each associated with one of a set of mutually exclusive and exhaustive *triggers*.

In our revised action description, we encode triggers and the associated postconditions by replacing the single conjunctive postcondition δ with a set of trigger/postcondition pairs, $(\tau(i), \delta(i))$ where each $\tau(i)$ and $\delta(i)$ is a conjunction of literals. We require exhaustivity and mutual exclusion over the triggers $\tau(i)$:

$$\forall i, j \; i \neq j \Rightarrow \neg\tau(i) \wedge \tau(j)$$
$$\bigvee_i \tau(i)$$

The semantics of a context-dependent action is then as given in Definition 3 We can reason about the effects of these actions using axiom schemas. For each action α and proposition ϕ, let T_+ be the set of $\tau(i)$ st $\phi \in \delta(i)$ and let T_- be the set of $\tau(i)$ st $\neg\phi \in \delta(i)$, then we have axioms:

$$[A]\,\phi \equiv \bigvee T_+ \vee \left(\left(\neg \bigvee T_-\right) \wedge \phi\right)$$

In other words, ϕ necessarily holds following action A if and only if either some trigger condition holds whose effects include ϕ, or ϕ is true immediately preceding A and no trigger condition is satisfied whose effects negate ϕ. In this framework, T_+ is the *causation preconditions* for ϕ, and T_- is the causation preconditions for $\neg\phi$.

At this point our framework has the expressive power of Pednault's ADL, restricted to a propositional world description. For planning systems that enforce complete knowledge of the initial state, this is no more expressive than using STRIPS rules; triggers can be treated as schema encoding multiple distinct sets of preconditions for what should be distinct operators. For a system allowing incomplete knowledge, the situation is different. Triggers which cannot be distinguished on the basis of what is known about the current state of the world introduce disjunction in the effects of actions. Limited quantification such as appears in UCPOP (Penberthy & Weld 1992) can be viewed as encoding the same kind of disjunction in a different notation.

This action definition is still not strong enough to accommodate information-gathering actions (Goldman & Boddy 1994b), which requires a representation for knowledge.

We would also like to be able to reason about *nondeterministic* actions: actions whose outcomes are *in principle* not predictable by our agent, such as coin tosses. A common and intuitive way of treating these *nondeterministic* actions has been as context-dependent actions whose triggers are unknown or unknowable.

For actions that can only be done once, or where any subsequent repetition has an outcome dependent on the first outcome, this is reasonable. Observations, for example, have this property. The proposition to be observed is unknown (but has some defined truth value) before the observation. After the observation, we know what that value is, and, in the absence of any actions possibly changing that value, all subsequent observations must return the same result. For actions that represent individual, independent events such as coin flips, matters are not so simple. Each separate instance of that action would have to have its own unique, unknown "proposition," making the set of propositions defining world states a function of the current (partial) plan. Instead, we augment our action definition, to allow each trigger to have not one outcome, but a set of mutually-exclusive and exhaustive outcomes.

We redefine actions in Definitions 4 and 5. Combining nondeterminism and context-dependence is notationally cumbersome. Therefore, we provide the definition in two steps. First, we define nondeterministic outcomes in terms of a (nondeterministic) choice between primitive, deterministic actions (Definition 4). Then, we apply this construction to the use of triggers to provide context-dependence and provide the overall semantics (Definition 5). Intuitively, then, nondeterministic, context-dependent actions are equivalent to a conditional statement, branching on the state of the triggers, within which is nested a nondeterministic choice.

Planning with nondetermistic actions in the absence of any capability for explicit information-gathering requires the assumption that the outcomes of those actions are always observed by the planning agent when they happen. This is the assumption made by Peot and Smith's CNLP and by our own PLINTH. In more realistic domains, we must take into account that the outcomes of context-dependent and nondeterministic actions may not be known. This is the topic of the next section.

Knowledge

In this section, we define the machinery necessary to reason about information-gathering and the agent's information state. We do this by defining an accessibility relation and a conventional Kripke semantics (see Definition 6). The agent's "knowledge" is defined as the set of propositions that hold in all knowledge-accessible worlds: those accessible through κ. The details are given in Definition 7. Intuitively, the agent's beliefs are founded on a correct understanding of the outcomes of actions.

Let us consider how we are to represent the action of observing some proposition, $\phi \in \Phi$ ($obs(\phi)$). There may be some preconditions (e.g., the radio must be on and the agent must be near it in order to observe the weather). There will be two triggers for the action, ϕ and $\neg\phi$. The postconditions for these triggers are $\mathbf{Know}\phi$ and $\mathbf{Know}\neg\phi$, respectively.

We escape the trap, common to systems confusing truth and knowledge, of needing to conclude that ϕ was true-and-known-to-be-true in the past, because ϕ is a trigger and $\mathbf{Know}\phi$ the effect. The definition of the accessibility relation κ does not permit us to conclude anything about the truth value of $\mathbf{Know}\phi$ before the observation was made.

Observations may also occur as a side-effect of other actions, where the outcome of the action may be observed by the executing agent. In that case, when composing the library of operators, one should indicate those literals that become known, as well as becoming true. For example, on a well-regulated train system, one knows when one has reached the destination.

An executing agent may also come to know a literal simply because that literal holds in all possible outcomes of an action. An example of this kind of reasoning is conformant motion. If one moves towards an obstacle a distance sufficient to cover all possible initial positions, one can infer one's position at the end of the motion, even without any knowledge of the starting point.

Conditional plans

Previous conditional planners, including CNLP, Plinth, and C-Buridan, all make the branching actions in their conditional plans implicit: in a given context, do a_1. In a different context, do a_2. The problem of making those contexts correspond to appropriate branch points and branching conditions is left to the planning algorithm.

In PDL, we can introduce branching into our programs explicitly using a combination of '?' and '$\cup$.' That is, branching is defined in terms of testing and non-deterministic choice. To take the example from Peot and Smith's paper, we might have a conditional program of the form:

$$\begin{array}{ll} (\text{know road clear?; drive to ski resort}) & \cup \\ (\text{know road not clear?; fly to ski resort}) & \cup \\ ([\text{not know road clear} \wedge \text{not know road not clear}]\text{; fail}) & \end{array}$$

These conditional tests do *not* correspond to any conventional notion of a "plan step." They are an explicit part of a WCPL plan for which there is no corresponding structure in any STRIPS-based planning language we are aware of.

Like Rosenschein (1981), we restrict our programs so that introducing choice points preserves termination. This is done by requiring the tests to be mutually exclusive and exhaustive. We differ from Rosenschein in the restriction we make on the propositions in the tests. We require that the conditionals be of the form:

$$\begin{array}{l} \mathbf{Know}(\phi)?; A \cup \\ \mathbf{Know}(\neg\phi)?; B \cup \\ ((\neg\mathbf{Know}(\phi) \wedge \neg\mathbf{Know}(\neg\phi))?; C) \end{array}$$

where A, B, C are programs whose conditionals are restricted as above. This formal restriction corresponds to the fact that our conditionals all pertain to the agent's knowledge of its environment at plan execution time. We can make more complex conditionals by nesting conditionals of the above form.

We now have the necessary tools to solve the "bomb in the toilet" problem in both of its forms. We present our solutions in the following section.

Examples

In this section, we present WCPL plans for the two versions of the bomb problem.

In order to correctly represent dependencies in the domain in the presence of uncertainty, it may be necessary to introduce dummy actions. [2] These dummy actions have alternative outcomes that exhaustively enumerate possible initial states with respect to some set of propositions. For the bomb problems, we introduce the following dummy action:

Bomb location: *(BL)*	
preconditions:	$\emptyset$
postconditions:	
trigger	**outcomes**
$\top$	{in(bomb, pkg1), ¬in(bomb, pkg2)}
	{¬in(bomb, pkg1), in(bomb, pkg2)}

which captures the constraint that the bomb can't be in two packages at once.

We also need two actions that involve putting packages in the toilet. Here is an action schema that describes these actions:

Flush(?pkg)	
preconditions:	$\emptyset$
postconditions:	
trigger	**outcome**
in(bomb, ?pkg)	{in(?pkg, toilet), ¬armed(bomb)}
¬in(bomb, ?pkg)	{in(?pkg, toilet)}

This is sufficient to tackle the first version of Moore's problem, in which we are unable to observe the contents of the packages (and have no need to, for that matter). The initial situation is: {armed(bomb), ¬in(pkg1, toilet), ¬in(pkg2, toilet)}. The goal is {¬armed(bomb)}. One solution to the problem is the plan:

$$BL; \mathit{flush(pkg1)}; \mathit{flush(pkg2)}$$

It is easy to see that the above plan is well-formed, since none of the actions have preconditions. The plan will also necessarily achieve its goal. In order to show that this is the case, we need Proposition 1. We may then prove the correctness of this plan, using the inference rule: $[A;B]\,P \equiv [A]\,[B]\,P$, as follows:

$$\begin{aligned}
[BL; \mathit{flush(pkg1)}; \mathit{flush(pkg2)}]\, \neg\mathrm{armed(bomb)}?\\
[BL; \mathit{flush(pkg1)}]\,[\mathit{flush(pkg2)}]\, \neg\mathrm{armed(bomb)}\\
[BL; \mathit{flush(pkg1)}]\, \neg\mathrm{armed(bomb)} \vee \mathrm{in(pkg2,\ toilet)}\\
[BL]\,[\mathit{flush(pkg1)}]\, \neg\mathrm{armed(bomb)} \vee \mathrm{in(pkg2,\ toilet)}\\
[BL]\, \neg\mathrm{armed(bomb)} \vee \mathrm{in(pkg2,\ bomb)} \vee \mathrm{in(pkg1,\ bomb)}\\
[BL]\, \mathrm{in(pkg2,\ bomb)} \vee \mathrm{in(pkg1,\ bomb)}
\end{aligned}$$

For the second version of the problem, we restrict ourselves to a single-package toilet, refining our flush operator as follows:

Flush(?pkg)	
preconditions:	{¬full(toilet)}
postconditions:	
trigger	**outcome**
in(bomb, ?pkg)	{in(?pkg, toilet), full(toilet), ¬armed(bomb)}
¬in(bomb, ?pkg)	{full(toilet), in(?pkg, toilet)}

We require, in addition, an inspection axiom schema:

Inspect(?pkg) *(I(?pkg))*	
preconditions:	$\emptyset$
postconditions:	
trigger	**outcome**
in(bomb, ?pkg)	{**Know**(in(bomb, ?pkg))}
¬in(bomb, ?pkg)	{**Know**(¬in(bomb, ?pkg))}

The initial conditions for the second problem are:

$$\{\ \mathrm{armed(bomb)}, \neg\mathrm{in(pkg1,\ toilet)}, \neg\mathrm{in(pkg2,\ toilet)}, \neg\mathrm{full(toilet)}\ \}$$

and the goal, as before, is {¬armed(bomb)}. One solution to this problem is the program:

$$\begin{aligned}
&BL; (\mathit{I(pkg1)});\\
&(\mathbf{Know}(\mathrm{in(bomb,\ pkg1)})?; \mathit{flush(pkg1)}) \cup\\
&(\mathbf{Know}(\neg\mathrm{in(bomb,\ pkg1)})?; \mathit{flush(pkg2)}) \cup\\
&(\ (\neg\mathbf{Know}(\mathrm{in(bomb,\ pkg1)}) \wedge\\
&\ \neg\mathbf{Know}(\neg\mathrm{in(bomb,\ pkg1)}))? \quad ; \epsilon\ )
\end{aligned}$$

The proof of correctness proceeds very much as for the first problem, with the minor complications that we must employ the fact that if the bomb is not in package 1, it must be in package 2, and we prove that the branch of the plan in which the agent is ignorant will never be reached.

Other related work

There is a large body of related work in a number of different areas. Many of those connections have been drawn elsewhere in the paper, but there are three broad areas that should at least be touched upon.

Conditional planning

Though most of the work on conditional planning is relatively recent, the first conditional planning system was described almost twenty years ago (Warren 1976). WARPLAN-C was a linear planner using STRIPS rules extended to express uncertain outcomes, designed to support automatic programming. The system was purely forward-planning. CNLP (Peot & Smith 1992) is a nonlinear conditional planner. As mentioned previously, their system finesses the distinction between world states and the planner's knowledge.

Finally, the Cassandra planning system (Pryor & Collins 1995) addresses many of the same issues discussed here, with some differences in the results. First, knowledge and truth are not clearly separated. Second, their encoding of nondeterministic actions as context-dependent actions with special "unobservable" secondary preconditions means that a distinction must be made *a priori* as to which plan steps have uncertain effects, rather than inferring that information from the

[2] These are generalizations of the dummy start action used in conventional planners.

agent's information state. Finally, Cassandra is not capable of the reasoning by cases needed to provide a solution to the first version of the bomb in the toilet problem.

Logics of knowledge

Scherl and Levesque (1993) address the problem of integrating knowledge into an action representation by defining regression operators for a knowledge predicate in the situation calculus. There is also a sizable literature on refining logics of knowledge to handle various paradoxes and modelling problems (e.g., knowledge preconditions for actions) (Morgenstern 1987). So far, we avoid these issues by sticking to a propositional representation.

Probabilistic planning

The planning systems Buridan (Kushmerick, Hanks, & Weld 1993) and C-Buridan (Draper, Hanks, & Weld 1993) have an underlying semantics very similar to WCPL. In both cases, uncertain information is characterized in terms of quantification over sets of possible world states, and the effects of actions as maps between sets of world states. However, we do not attach probabilities to uncertain information. We avoid a full probabilistic model in the interest of simplicity and possibly heuristic adequacy (this remains to be demonstrated).

Buridan includes nondeterministic events and uncertain knowledge of the initial state, but there is no provision for adding information. All reasoning about the probabilities associated with propositions in any given situation can be done at the time the plan is constructed. C-Buridan adds to this model the possibility of adding information through observations and making decisions based on that information.[3] C-Buridan labels are not explicitly assigned a semantics corresponding to knowledge, which appears to prevent the integration of conformant and conditional planning possible in WCPL.

Conclusions and future work

In this paper, we present an approach to the problem of constructing conditional plans in a framework involving partial information and context-dependent and nondeterministic actions. One of the central results of this investigation is that these extensions are not independent. One cannot apply the solutions to the individual problems in some "composition" and expect the resulting language to make sense, much less to solve the overall problem.

We present a new planning language, WCPL, providing a unified treatment and a coherent semantics for conditional planning with context-dependent and nondeterministic actions, including a principled treatment of the distinction between ground truth and the planner's information state. We provide an explicit semantics for the planner's knowledge as a modal operator over possible states of the world. Modelling the planner's information state in this way enables us to integrate information-gathering in conditional plans, and conformant or "fail-safe" plans. This allows us to handle both versions of Moore's "bomb in the toilet" problem.

There is plenty of room to extend the work presented here. Two issues we have not yet dealt with in a satisfactory way are exogenous events and the kind of quantification handled in UWL (Etzioni *et al.* 1992) by the use of run-time variables. Given a finite domain, existential quantification of this sort can be represented as disjunction, so this does not appear to be a semantic problem, but there is certainly a heuristic problem to be addressed.

We are applying WCPL to analyze planning algorithms. In collaboration with Pryor, we have developed a semantics for Cassandra plans and proof of soundness of the Cassandra algorithm, based on WCPL.[4] We are also working to develop an algorithm for planning in WCPL.

Acknowledgements We would like to thank G.N. Kartha and the two anonymous reviewers for assistance in correcting errors in the manuscript.

References

Blythe, J. 1994. Planning with external events. In de Mantaras, R. L., and Poole, D., eds., *Uncertainty in Artificial Intelligence,Proceedings of the Tenth Conference*, 94–101. Morgan Kaufmann.

Draper, D.; Hanks, S.; and Weld, D. 1993. Probabilistic planning with information gathering and contingent execution. Technical Report 93-12-04, Department of Computer Science and Engineering, University of Washington, Seattle, WA.

Etzioni, O.; Hanks, S.; Weld, D. S.; Draper, D.; Lesh, N.; and Williamson, M. 1992. An approach to planning with incomplete information. In Nebel, B.; Rich, C.; and Swartout, W., eds., *Principles of Knowledge Representation and Reasoning:Proceedings of the Third International Conference*, 115–125. Los Altos, CA: Morgan Kaufmann Publishers, Inc.

Goldman, R. P., and Boddy, M. S. 1994a. Conditional linear planning. In Hammond, K. J., ed., *Artificial Intelligence Planning Systems: Proceedings of the Second International Conference.* Los Altos, CA: Morgan Kaufmann Publishers, Inc.

Goldman, R. P., and Boddy, M. S. 1994b. Representing uncertainty in simple planners. In Doyle, J.; Sandewall, E.; and Torasso, P., eds., *Principles of*

[3] Essentially, moving from an unobservable Markov process to the problem of controlling a partially-observable Markov process.

[4] A report on this work will soon be available.

Definition 1 (Semantics of programs)

$$\begin{aligned}
m(\epsilon) &= \{(s,s) \mid s \in W\} \\
m(P?) &= \{(s,s) \mid s \in \pi(P)\} \\
m(A;B) &= m(A) \circ m(B) \\
m(A \cup B) &= m(A) \cup m(B)
\end{aligned}$$

Definition 2 (PDL equivalents for STRIPS actions) *A STRIPS operator o with preconditions γ and postconditions δ is defined as a PDL action A with the following semantics:*

$$\begin{aligned}
m(A) = \{ \quad & (s,t) \subseteq W^2 \mid \\
& (t \models \mathbf{planfail} \equiv s \models \mathbf{planfail} \vee s \notin \pi(\gamma)) \wedge \\
& (\forall \phi \in \Phi,\ \ t \models \phi \equiv (s \models \phi \wedge \neg\phi \notin \delta) \vee (\phi \in \delta))\}
\end{aligned}$$

Definition 3 (Context-dependent actions)

$$\begin{aligned}
m(A) = \{ \quad & (s,t) \in W^2 \mid \\
& (\mathbf{planfail} \in t \equiv \mathbf{planfail} \in s \vee s \notin \pi(\gamma)) \wedge \\
& (\forall \phi \in \Phi,\ t \models \phi \equiv ([s \in \pi(\tau(i))] \rightarrow (s \models \phi \wedge \neg\phi \notin \delta(i)) \vee (\phi \in \delta(i))))\}
\end{aligned}$$

Definition 4 (Nondeterministic context-dependent actions) *Actions are specified in terms of a set of preconditions, γ, and postconditions δ. As before, δ is a set of outcomes of the form $(\tau(i), \delta(i))$, where the $\tau(i)$'s are mutually-exclusive and exhaustive triggers. However, now the $\delta(i)$'s will be of the follwing form:*

$$\delta(i) \stackrel{\triangle}{\equiv} \{\delta(i,1), \delta(i,2) \ldots \delta(i,k)\}$$

for some k, where the $\delta(i,j)$'s are conjunctions of literals.

Definition 5 (Outcomes of nondeterministic actions) *First we define for each trigger a corresponding pseudo-action, $\alpha(i)$ and define*

$$m(A) = \{(s,t) \mid \forall i\, [s \in \pi(\tau(i)) \wedge (s,t) \in m(\alpha(i))]\}$$

and

$$m(\alpha(i)) = \{(s,t) \mid \exists \delta(i,j) \in \delta(i) \wedge (s,t) \in \delta(i,j)\}$$

finally,

$$m(\delta(i,j)) = \{(s,t) \mid \forall(\phi \in \Phi),\ t \models \phi \equiv (\neg\phi \notin \delta(i,j) \wedge s \models \phi) \vee (\phi \in \delta(i,j))\}$$

Definition 6 (Accessibility relation) *We augment the semantics of the logic with a second accessibility relation k and an interpretation of wffs κ over the worlds in W (the interpretation is defined as π, mutatis mutandis). We define κ as follows:*

$$\begin{aligned}
k(\alpha) &= \{(s,t) \mid s,t \in W \wedge [\exists((s',t') \in m(\alpha))\ \ st\ s = s' \wedge t = t']\} \\
k(\epsilon) &= \{(s,s) \mid s \in W\} \\
k(P?) &= \{(s,s) \mid s \in \kappa(P)\} \\
k(A;B) &= k(A) \circ k(B) \\
k(A \cup B) &= k(A) \cup k(B)
\end{aligned}$$

Definition 7 (Knowledge) *We define the knowledge modality in terms of quantification over belief-accessible worlds.*

$$\begin{aligned}
[A]\, \mathbf{Know} P &\stackrel{\triangle}{\equiv} \{s \in W \mid \forall t \in W, (s,t) \in k(A) \rightarrow t \in \kappa(P)\} \\
\langle A \rangle\, \mathbf{Know} P &\stackrel{\triangle}{\equiv} \{\exists s \in W \mid \forall t \in W, (s,t) \in k(A) \rightarrow t \in \kappa(P)\}
\end{aligned}$$

Proposition 1 (Axiom schema for context-dependent outcomes) *For each $\alpha \in \aleph$ we have*

$$[\alpha] \bigvee_{i,j} \delta(i,j)$$

Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94). San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Kushmerick, N.; Hanks, S.; and Weld, D. 1993. An algorithm for probabilistic planning. Technical Report 93-06-03, Department of Computer Science and Engineering, University of Washington, Seattle, WA. to appear in *Artificial Intelligence*.

McDermott, D. V. 1987. A critique of pure reason. *Computational Intelligence* 3:151–160.

McDermott, D. 1991. Regression planning. *International Journal of Intelligent Systems* 6(4):357–416.

Morgenstern, L. 1987. Knowledge preconditions for actions and plans. In McDermott, J., ed., *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, 867–874. Los Altos, CA: Morgan Kaufmann Publishers, Inc.

Pednault, E. 1988. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence* 4(4):356–372.

Penberthy, J. S., and Weld, D. S. 1992. UCPOP: a sound, complete, partial order planner for ADL. In Nebel, B.; Rich, C.; and Swartout, W., eds., *Principles of Knowledge Representation and Reasoning:Proceedings of the Third International Conference*, 103–114. Los Altos, CA: Morgan Kaufmann Publishers, Inc.

Peot, M. A., and Smith, D. E. 1992. Conditional nonlinear planning. In Hendler, J., ed., *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, 189–197. Los Altos, CA: Morgan Kaufmann Publishers, Inc.

Pryor, L., and Collins, G. 1995. Planning for contingencies: A decision-based approach. Unpublished manuscript.

Rosenschein, S. J. 1981. Plan synthesis: A logical perspective. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, 331–337. Los Altos, CA: Morgan Kaufmann Publishers, Inc.

Scherl, R. B., and Levesque, H. J. 1993. The frame problem and knowledge-producing actions. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 689–695. Menlo Park, CA: AAAI Press/MIT Press.

Tate, A.; Drabble, B.; and Dalton, J. 1994. Reasoning with constraints within O-Plan2. 99–109.

Vere, S. A. 1983. Planning in time: Windows and durations for activities and goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5(3):246–267.

Warren, D. H. 1976. Generating conditional plans and programs. In *Proceedings of the AISB Summer Conference*, 344–354.