A Representation for Efficient Planning in Dynamic Domains with External Events

Jim Blythe
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jblythe@cs.cmu.edu

Abstract

I present a language for specifying planning problems in which there are external events, that is, events beyond the direct control of the planner which may take place while a plan is being executed. The language is based on Miller and Shanahan's representation of narratives in the situation calculus. I describe an implemented planner and prove that it produces correct plans for a subset of this language. I discuss how some aspects of the language affect the efficiency of the planner.

Introduction

An important direction for AI planning systems is to handle dynamic domains — domains where changes take place that are not the direct result of the actions of the agent, and where the world describes a state trajectory in the absence of any actions from the planning agent. In many real-world planning problems, for example, change to the world takes place due to other agents, or due to forces of nature. The planning agent may have limited control over these external events, and limited knowledge about both events that have occurred and events that may possibly occur in the future.

The aim of this paper is twofold: to give an account of an action model for dynamic domains that is used in an implemented planner, and to show how issues of representational power and computational efficiency jointly influence the model. A model for dynamic domains for planners should have at least the following three features. First, the planner should be able to take advantage of events beyond its direct control, incorporating them into a plan as appropriate. Second, the model should allow the planner to reason about a set of different possible futures, each being consistent with the agent's knowledge of events. Third, the planner should be able to efficiently determine the *relevant* events that may effect candidate partial plans, and obtain information that may allow a modified plan to be made safe from these events.

In this paper I focus on a model for representing external change that is faithful to the Weaver planner (Blythe 1994; 1996). I discuss the extent to which it satisfies the criteria above. The model is based on narratives in the situation calculus as described by Miller and Shanahan (Miller & Shanahan 1994). This language was chosen because it provides an extension to the situation calculus that can describe the occurrence of events at specific times and deal with concurrent events.

I divide event types into two sets: those under the control of the planning agent (*actions*) and those beyond the agent's control (*external events*). This division introduces a notion of controllability into the action language which is crucial for reasoning about the plans of agents. I use a scoped circumscription policy as described in (Miller & Shanahan 1994) to reason about plans in the presence of external events. The planner does not explicitly use circumscription to reason about plans, but it creates plans that are sound in the sense that goal satisfaction can be proven from the domain model, initial state and actions using circumscription.

In the next section I present a review of Miller and Shanahan's model for narratives in the situation calculus. In the following section I describe modifications that model the language for external events used by Weaver. The semantics for the events were chosen both for their representational power and to allow for efficient planning algorithms, which is achieved by making aspects of structure explicit and by allowing convenient reasoning for backward chaining planners. After this I describe the planner and also sketch proofs that Weaver is sound with respect to this model. In the final section I describe techniques for reasoning about plans within the model that help satisfy the third criterion of efficiently finding the relevant external events. The final section contains a brief discussion.

A review of narratives in the situation calculus

Miller and Shanahan provide sorts for situations, action types and fluents. The term $Result(a, s)$ represents the situation resulting from performing action a in situation s. The formula $Holds(f, s)$ represents that fluent f holds in situation s. For example, consider a laundry domain which contains two action types, *Wash* and *Hang-out*, and is described by three fluents, *Clean*, *Dry* and *Hanging*. Actions of type *Wash* make *Clean* true and *Dry* false, while actions of type *Hang-out* make *Hanging* true. This is represented by the following formulae:

$Holds(Clean, Result(Wash, s))$
$\neg Holds(Dry, Result(Wash, s))$
$Holds(Hanging, Result(Hang\text{-}out, s))$

where s is universally quantified over situations. In this

case neither action has any preconditions. When there are action preconditions or context-dependent effects, they will be represented as implications. Following Baker (Baker 1991), Miller and Shanahan include the frame axiom

$$[Holds(f, Result(a, s)) \leftarrow Holds(f, s)] \leftarrow \neg Ab(a, f, s)$$

and four existence of situation axioms:

$$Holds(And(g1, g2), s) \leftrightarrow [Holds(g1, s) \wedge Holds(g2, s)]$$
$$Holds(Not(f), s) \leftrightarrow \neg Holds(f, s)$$
$$Holds(g, Sit(g)) \leftarrow \neg Absit(g)$$
$$Sit(g1) = Sit(g2) \rightarrow g1 = g2$$

Facts about resulting states may be proved using a circumscription policy that minimises *Absit* at a higher priority than *Ab* while allowing the initial state $S0$ and the *Result* function to vary. For example this can be used to prove *Holds(Clean, Result(Hang-out, Result(Wash, S0)))* in the laundry domain.

Miller and Shanahan extend the representation to describe *narratives* which deal with the occurrence of actual events. They add to the language a new sort for times, interpreted by the non-negative reals, a new predicate $Happens(\alpha, \tau, \tau')$, representing that an action of type α happens in the time interval between τ and τ', and a new term $State(\tau)$ denoting the situation at time τ. An event history is then represented by a conjunction of *Happens* predicates, and two new axioms link the *State* term to the event history in the intuitive way (see (Miller & Shanahan 1994) for details). The first axiom says that the situation at time t is $Result(a1, State(t1))$ if action $a1$ completes at time $t1$ and no other action happens between $t1$ and t:

$$\begin{aligned}(State(t) &= Result(a1, State(t1))) \\ &\leftarrow [Happens(a1, t0, t1) \wedge t0 < t1 \wedge t1 < t \wedge \\ &\quad \neg\exists a2, t2, t3[Happens(a2, t2, t3) \wedge \\ &\quad [a1 \neq a2 \vee t1 \neq t2] \wedge t1 \leq t2 \wedge t2 < t1]]\end{aligned}$$

The second axiom describes the initial situation:

$$State(t) = S_0 \leftarrow \neg\exists a1, t1, t2[Happens(a1, t1, t2) \wedge t1 < t]$$

The circumscription policy is amended to minimise *Absit* at a high priority and then to minimise *Ab* and *Happens* in parallel, allowing *Result* and *State* to vary. This intuitively represents the assumption that no events take place that are not explicitly represented in the domain. If we let T stand for the conjunction of the domain definition axioms (*Holds* predicates), the frame axioms and necessary uniqueness-of-names axioms, and let N stand for the narrative predicates, then this policy can be stated as

$$CIRC(T \wedge N; Absit; Ab, Happens, Result, Holds, S_0, State)$$
$$\wedge CIRC(T \wedge N; Ab; Happens, Result, Holds, S_0, State)$$

Where $CIRC(T; \pi, \theta; \xi\psi)$ denotes the parallel circumscription of π and θ in a theory T with ξ and ψ allowed to vary. Miller and Shanahan show that the circumscription policy they introduce entails the circumscription policy of Baker (Baker 1991), which allows useful deductions to be made about states of the world. The proof of this assumes that the narrative description is a finite conjunction of atomic *Happens* predicates. For example, if we add to the laundry domain the following narrative predicates:

Happens(Wash, 1, 4)
Happens(Hang-out, 5, 6)
we can prove *Holds(Clean, State(8))*.

Overlapping actions

Finally, predicates are added to specify the effects of action occurrences which overlap in time. In most cases we might expect that two action occurrences that overlap will not alter each other's effects, and that is the default case in Miller and Shanahan's language. However the predicate *Cancels* is included to specify when the effects of actions that take place concurrently will be altered, and an axiom is added to derive the fluents that hold after concurrent actions are taken. This language allows a much richer set of interactions than is supported in Weaver, and I describe axioms below that restrict this model. For this reason the full model is not presented here. Details can be found in (Miller & Shanahan 1994).

A new function symbol *Duration* maps actions to numbers, denoting the duration of an action type, which must be positive. Two operators are used to combine action types. $\alpha1; \alpha2$ represents the action of $\alpha1$ immediately followed by $\alpha2$ and $\alpha1\&\alpha2$ represents the actions performed concurrently. Two function symbols represent ways to decompose operators in time. $Head(\alpha, \delta)$ represents the first δ time units of action type α and $Tail(\alpha, \delta)$ represents the remainder of α. Any such action derived from α will be termed a "horizontal component" of α, and the predicate $Slice(\alpha1, \alpha)$ is defined to mean that $\alpha1$ is a horizontal component of α. The predicate $Part(\alpha1, \alpha)$ means that $\alpha1$ is derived from α by taking some sequence of horizontal and vertical components. An equivalence operator $\approx$ is introduced to represent that two different ways of writing an action are identical. These symbols are defined by the following axioms:

$$Result(a1; a2, s) = Result(a2, Result(a1, s))$$
$$a1 \approx a2 \rightarrow Result(a1, s) = Result(a2, s)$$
$$a \approx Head(a, d); Tail(a, d)$$
$$Part(a1, a2) \Leftrightarrow [Slice(a1, a2) \vee \exists a3, a4[Slice(a1, a3) \wedge a2 \approx a3\&a4]]$$

The predicate $Cancels(\alpha2, \alpha1, f, s)$ represents that action $\alpha2$ stops $\alpha1$ from having its usual effect on fluent f when the two actions are performed concurrently in situation s. Axioms are included so that in this case any equivalent action to $\alpha2$ will also cancel $\alpha1$, as will any horizontal component.

The additional axiom for defining effects of actions is

$$\begin{aligned}&holds(f, Result(a, s)) \leftarrow \\ &\quad [a \approx a1\&a2 \wedge Holds(f, Result(a1, s)) \wedge \\ &\quad \neg\exists a3[Part(a3, a2) \wedge Cancels(a3, a1, f, s)]]\end{aligned}$$

In order to reason about narratives with overlapping events, axioms are added that assert that the occurrence of an action is equivalent to the occurrence of all its components, horizontal or vertical. The predicate $\approx$ is circumscribed at the highest priority, then *Absit* and then *Cancels*, and finally *Ab* and *Happens* in parallel. One more predicate, $Happens^{\dagger}(\alpha, \tau 1, \tau 2)$, is introduced to represent that action α takes place in the time period from $\tau 1$ to $\tau 2$ and *no other* action takes place in this time period, defined by the following axiom:

$$Happens^{\dagger}(a, t1, t2) \Leftrightarrow$$
$$[Happens(a, t1, t2) \wedge$$
$$\neg \exists a1, t2, t4[Happens(a1, t3, t4)$$
$$\wedge t1 \leq t3 \wedge t4 \leq t2 \wedge \neg Part(a1, a)]]$$

Finally the axiom relating the *State* and *Result* functions is altered as follows:

$$State(t) = Result(a1, State(t1)) \leftarrow$$
$$[Happens^{\dagger}(a1, t1, t2) \wedge t \geq t2 \wedge$$
$$\neg \exists a3, t3, t4[Happens(a3, t3, t4) \wedge t2 < t4 \wedge t4 \leq t]]$$

Modifications to Miller and Shanahan's Model

A small number of additions and restrictions are made to the language of narratives described in (Miller & Shanahan 1994) in order to provide a model for the Weaver planner, and they are described here. First, a predicate is defined that captures the notion of the control that the agent has over its domain. Second, limits are placed on the way concurrent events can interact to simplify the planner's task. Third, a notion of conditional narratives is introduced, to represent conditional plans in such a way that Baker's circumsciptive policy still works correctly.

External events

My aim is to describe a planner for domains in which external events may take place that are beyond the control and prior knowledge of the planning agent, although I assume the agent knows the types of external events that may occur, and their consequences on the world. For instance, a robotic agent may know that it shares its domain with other robotic agents and with humans. The other agents can move around, pick things up and leave doors open, but the planning agent does not know exactly which of the possible events might occur, or when.

In order to model domains with external events I make use of an extension Miller and Shanahan introduce to their language in (Miller & Shanahan 1994): the minimisation of the *Happens* predicate in the circumscription policy is replaced by a *Happens** predicate, that only captures a subset of the events. In this case I am interested in whether events are under the control of the planning agent.

I introduce a new predicate $Agent(\alpha)$ to represent that the action type α is under the control of the planning agent. When an action type is under the control of the planning agent, one can reasonably assume that the action does not take place without the agent's knowledge. This is not true for events ϵ for which $\neg\ Agent(\epsilon)$ is true. The predicate $Agent(f)$ is also used to denote that the fluent f is only altered by actions α for which $Agent(\alpha)$. In what follows I will refer to event types a for which $Agent(a)$ is true as action types and those for which it is not true as external event types.

Specifically, we add the sentences

$$Happens(a, t1, t2) \wedge Agent(a) \rightarrow Happens^*(a, t1, t2)$$

$$(\neg \forall s[Holds(f, Result(a, s)) \Leftrightarrow Holds(f, s)]) \rightarrow$$
$$\neg Agent(f) \vee Agent(a)$$

and follow a circumscription policy of minimising *Absit* at a high priority and then minimising *Ab* and *Happens** in parallel, allowing *Happens*, *Result*, $S0$ and *State* to vary.

For example, suppose the laundry domain is augmented with a new event type *Wind-blows*, with the following characterisation:

$$Holds(Windy, Result(Wind\text{-}blows, s))$$
$$Holds(Dry, Result(Wind\text{-}blows, s)) \leftarrow Holds(Hanging, s)$$

and suppose the following controllability predicates are added:

$$Agent(Wash)$$
$$Agent(Hang\text{-}out)$$
$$\neg Agent(Wind\text{-}blows)$$
$$Agent(Clean)$$

Then given the previous narrative description, it is still possible to prove $Holds(Clean, State(8))$ but it is no longer possible to prove $\neg Holds(Dry, State(8))$ since the circumscription policy admits models in which one or more events of type *Wind-blows* take place.

Concurrent events and Weaver

The proof of $Holds(Clean, \text{State}(8))$ in the previous section relies on minimising the *Cancels* predicate so that the effects of *Wind-blows* are assumed to be independent from the effects of the action types, should they overlap. Interesting external events will not always be independent in their effects from the actions available to the agent, and so in general some mechanism is required to make the effects of these events precise. Weaver does not currently support the explicit specification of *Cancels* predicates in the domain, instead it requires that no two events that alter the same fluent can overlap. This is expressed as a limitation on the *Happens* predicate, rather than the *Cancels* predicate, because it is more convenient to reason about the times of completion using *Happens*. In the laundry example, there are no altered fluents in common between the *Wind-blows* event and any of the actions, so no restrictions need to be added.

The motivation for allowing multiple external events that may not interact with each other is the potential for efficiency in planning algorithms rather than for representational power. In this model, the external events can be split up into smaller units, and the disjointness restriction enforces a degree of structure. The fact that different sets of events may be combined in different states enables a more

parsimonious representation. The ways in which a planner may exploit this structure are discussed in the next section.

Balanced conditional narratives

When external events are present in the domain, the planner has incomplete knowledge about the initial state or there are non-deterministic action effects, a plan with conditional steps may succeed where any non-branching plan would fail. Such a plan has steps that are only executed when certain facts are true of the world at the time of executing the plan.

Since plans in Weaver are translated into narratives in this model, conditional plans are represented by conditional narratives. For example:

$$\begin{array}{rcl} \mathit{Happens}(\mathit{Wash}, 1, 2) & & \\ \mathit{Happens}(\mathit{Hang\text{-}out}, 2, 3) & \leftarrow & \mathit{Holds}(\mathit{Windy}, \mathit{State}(2)) \\ \mathit{Happens}(\mathit{Spin\text{-}dry}, 2, 3) & \leftarrow & \neg\mathit{Holds}(\mathit{Windy}, \mathit{State}(2)) \end{array}$$

Here the domain of section has been extended with the action *Spin-dry*, under agent control, which dries the laundry without the wind. Note that there is no model of the domain and this narrative in which both *Hang-out* and *Spin-dry* take place.

In order to show $\mathit{Holds}(\mathit{Dry}, \mathrm{State}(3))$ the proof that circumscription of narratives entails Baker's circumscription policy must be extended to conditional narratives, but this is simple to do. Miller and Shanahan's proof takes an arbitrary model M of

$$CIRC[T \wedge N; \mathit{Ab}, \mathit{Happens}; \mathit{Result}, \mathit{S0}, \mathit{State}] \quad (1)$$

where T is a domain description and N is a narrative description, or a conjunction of atomic statements of the form $\mathit{Happens}(\alpha, \tau 1, \tau 2)$. They show that M must also be a model of Baker's system,

$$CIRC[T; \mathit{Ab}; \mathit{Result}, \mathit{S0}] \quad (2)$$

The proof works by deriving a contradiction: since M is a model of T, if it is not a model of 2, then there is a model $\bar{M}$ that is strictly preferred to M. From $\bar{M}$ a model that is strictly preferred to M according to 1 can be found, which violates the original assumption.

This proof can easily be generalised to allow N to contain a finite number of conditional occurrences of the form $\mathit{Happens}(\alpha, \tau 1, \tau 2) \leftarrow \mathit{Holds}(f, \tau 0)$, where $\tau 0 \leq \tau 1$, by observing that any model M of (1) is a model of some unconditional narrative formed from N according to the actions which occur in the M's interpretation of *Happens*. Now the original proof can be re-used to show that M is also a model of (2).

Minimisation of *Happens** with arbitrary conditional narratives, however, can produce non-intuitive results. This is because a minimal model could include extra actions, either external events or those under the agent's control, in order to ensure that the preconditions of the conditional occurrences do not hold. Consider a subset of the previous narrative:

$$\begin{array}{rcl} \mathit{Happens}(\mathit{Wash}, 1, 2) & & \\ \mathit{Happens}(\mathit{Hang\text{-}out}, 2, 3) & \leftarrow & \mathit{Holds}(\mathit{Windy}, \mathit{State}(2)) \end{array}$$

From this narrative it is possible to prove $\neg\mathit{Holds}(\mathit{Windy}, \mathit{State}(2))$! This is because in a minimal model for *happens**, no *Wind-blows* events take place, so the precondition will not be satisfied and only the *Wash* action takes place. Despite its appeal to the paranoid, models in which external events depend on the agent's conditional future plans are undesirable. For this I introduce the concept of *balanced* conditional narratives, which have the property that whenever the narrative includes a conditional occurrence of the form $\mathit{Happens}(A, t1, t2) \leftarrow P$ where P is a combination of statements of the form $\mathit{Holds}(f, \mathit{State}(t1))$ using disjunction, conjunction and negation, then the set of all such statements in the narrative that refer to $\mathit{State}(t1)$ must form a partition of the possible situations that hold at that state. In other words, whatever the combination of fluents that hold at time $t1$, there must be exactly one action that will be taken. The narrative above with three occurrence statements is clearly a balanced conditional narrative. For the remainder of this paper, all conditional narratives are assumed to be balanced.

The restriction to balanced conditional narratives does not reduce the expressibility of the language if we include the action type *Void* in the domain, which is not mentioned in the domain theory. Then conditional occurrences of the *Void* action can be added to the narrative simply to achieve balance.

Planning

Having introduced the language used to model the class of problems for which the Weaver planner is designed, I will describe the planning algorithm and sketch a proof that the planner is sound, i.e. that plans produced by the planner in response to an initial state and goal description will achieve the goal from the initial state.

Weaver is based on Prodigy 4.0, with extensions to create conditional plans. In the following description of Prodigy 4.0 I follow (Fink & Veloso 1995). Prodigy is given an initial state description $S0$ and a goal G, a first-order sentence in the fluents of the domain. On completion, Prodigy returns a plan, a totally-ordered sequence of actions $(a_i | 1 \leq i \leq n)$ such that

$$\mathit{Holds}(G, \mathit{Result}(a_n, \mathit{Result}(a_{n-1}, \ldots \mathit{Result}(a_1, S0)))) \quad (3)$$

At any point in its execution, the state of Prodigy is described by its current incomplete plan. An incomplete plan has two parts, a head-plan and a tail-plan, depicted in Figure 1. The head-plan is a totally ordered sequence of action types, representing the prefix of Prodigy's current preferred plan. The tail-plan is a partially ordered set of actions, built by a backward-chaining algorithm.

Prodigy begins execution with both the head-plan and tail-plan empty. At each point in its execution, Prodigy applies an execution simulation algorithm to its head-plan, resulting in a simulated state *Current*. If $\mathit{Holds}(G, \mathit{Current})$ is true, Prodigy terminates and returns the head-plan as its plan. Otherwise, one of two steps is non-deterministically chosen: (1) pick an action type from the tail-plan that has

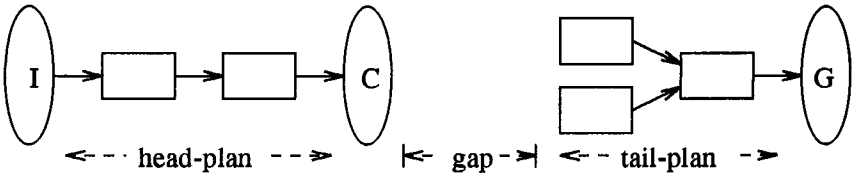

Figure 1: Representation of an incomplete plan.

no children and move it to the end of the head-plan, or (2) add a childless action type a to the tail-plan as the child of an existing step. The action a must be such that *Agent*(a) is true. These steps determine Prodigy's search space, along with a specific algorithm to determine the set of action types considered to add to a tail-plan in step (2).

In the absence of external events, the potential completeness of the search space is easy to see: with a simple algorithm in step (2) that only adds action types to the empty tail-plan and considers all possible action types, all possible sequences of actions of increasing length can be generated in the head-plan. In practice, a back-chaining algorithm will be used so that the tail-plan is a goal tree and back-jumping techniques will further reduce the search space. For more details, see (Fink & Veloso 1995).

In order to prove the soundness of the algorithm, i.e. that when a plan is returned one can prove equation 3, it is not necessary to know anything about the back-chaining algorithm used to generate the tail-plan, or the details of the search strategy. All that is important is that the execution simulation applied to the head-plan makes correct predictions about the state resulting from an actual execution of the head-plan. Execution simulation maintains a model of the state by applying the changes prescribed by the domain model for the action types in the head-plan, and no others, to a model of the initial state. Therefore a lemma used by Kartha (Kartha 1995) to prove that Baker's circumscription policy and Pednault's ADL language are equivalent for a class of domains can be adapted to show that the plans produced by Prodigy are correct with respect to the action language developed here, for the same class of domains and in the absence of external events.

Thus in the absence of external events, if a narrative is expressed as

$$\bigwedge_{i=1}^{n} Happens(a_i, t_i, t'_i)$$

where $t_i \leq t'_i \leq t_{i+1}$ for $1 \leq i \leq n$ and where the plan $(a_i | 1 \leq i \leq n)$ is returned by Prodigy to achieve goal G from initial state S_0, then we can prove

$$\forall t \geq t'_n [Holds(G, State(t))]$$

Planning with external events

Weaver makes use of Prodigy to produce plans ignoring external events, then determines external events relevant to the plan and considers the different outcomes that can arise from the plan based on the possible outcomes of external events. Based on this analysis, Weaver then re-directs Prodigy to produce a new plan, making two different kinds of modification. First, new actions may be added to the tail-plan whose effects are chosen to stop external events from having effects that interfere with the plan (rather than only to achieve the preconditions of other actions). Second, the planner can be applied to two or more cases of the execution simulation, accounting for different occurrences of external events, and combining to produce a conditional plan. Weaver's overall algorithm is summarised in Figure 2. The technical details of choosing actions and managing back-chaining for conditional plans are not covered here: they can be found in (Blythe 1994) and (Blythe 1996).

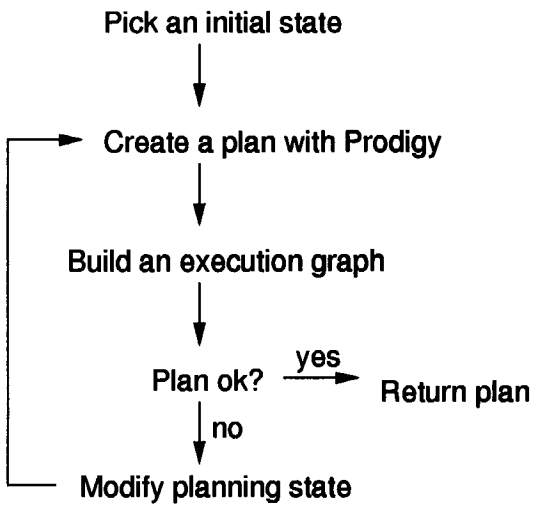

Figure 2: Summary of Weaver's algorithm

In this section I sketch a proof that the plans produced by Weaver are correct when the initial state is certain and there are no actions with nondeterministic effects.[1] In order to show this it is necessary to describe Weaver's plan evaluation module in more detail.

Weaver evaluates a plan by constructing an *execution graph*. The execution graph is a directed graph each of whose nodes is labelled with a fluent and a time, such that no two nodes share the same fluent and time. The arcs are labelled with either the symbol *persistence* or the symbol

[1] Weaver is correct without these assumptions, but the proof is more complicated.

effect. Assuming that the plan is represented by the narrative

$$N = \bigwedge_{i=1}^{n} Happens(a_i, t_{i-1}, t_i)$$

where $t_0 < t_1 < \ldots < t_{n-1} < t_n$, then the possible times in the node labels are the t_i. First, the plan consisting of just these actions is simulated by Prodigy. Then the graph is constructed as follows:

1. For each fluent g_i in the goal statement, the node (g_i, t_n) is added.
2. At each time t_i where $i > 0$, for each fluent f such that there exists a node (f, t_i),
   (a) if the value of f does not change between $State(t_{i-1})$ and $State(t_i)$, a node (f, t_{i-1}) is added and an arc (labelled *persistence*) is added between (f, t_{i-1}) and (f, t_i).
   (b) for each predicate of the form $Holds(f, Result(a_i, s)) \leftarrow P$ in the planning domain T, for each fluent p_j mentioned in P, a node (p_j, t_{i-1}) is added to the graph, and an arc (labelled *effect* is added from (p_j, t_{i-1}) to (f, t_i), regardless of whether f changes from $State(t_{i-1})$ to $State(t_i)$

Conditional plans are dealt with by creating an execution graph for each branch of the plan. Weaver uses the execution graph to search for potential flaws in a candidate plan by examining each link. If a link $[(f, t_i) \rightarrow (f, t_{i+1})]$ is labelled *persistence* then Weaver looks for external events e such that $Holds(\neg f, Result(e, s)) \leftarrow P$ is a statement in T and P is consistent with the fluent values in the nodes for time period t_p given by

$$t_p = \max\{t_j | t_j < t_i - \text{duration}(e)\}.$$

In order to show that the event may cause the plan to fail, Weaver examines the preconditions P to see if they are actually applicable in the simulated state, or if some chain of events can make them so. While it is my ultimate aim to show that this technique is correct, and that Weaver works for nondeterministic actions as well, in this paper I prove a weaker result:

Lemma 1 *Let the domain T, consisting of action types and external events, contain no non-deterministic effects and a completely specified initial state $S0$. Let the narrative N be such that Weaver discovers no persistence-threatening events in the execution graph for goal G, then G is entailed by the circumscription policy of (1).*

Proof sketch:

Suppose there is a model M of (1) with N and T defined for this domain and narrative such that the goal statement G is not true in $State(t_n)$. Then there must be an earliest time t_k for which the value of at least one fluent in the nodes (f, t_k) is different from that in the simulated narrative. $t_k > t_0$ since the initial state is fixed. Consider the links to (f, t_k) from nodes with time label t_{k-1}. If there is no persistence link, then there is an effect link for a_k corresponding to an expression $Holds(f, Result(a_k, s)) \leftarrow P$ in T such that the preconditions P are satisfied in $State(t_{k-1})$ but the result does not hold in $State(t_k)$. This contradicts that M is a model of $T \wedge N$.

Therefore there must be a persistence link $(f, t_{k-1}) \rightarrow (f, t_k)$ such that f has the same value in $State(t_{k-1})$ in M as in a model in which no external events occur, but a different value in $State(t_k)$. Thus M must include some external event occurrence $Happens(e, t', t)$ with $t_{k-1} < t < t_k$ that changes the value of f. But then Weaver would have found this event type in its examination of the execution graph for N. ∎

The purpose of the execution graph is to focus Weaver's attention on the external events that are relevant to the plan. If a model M differs from the model with no external events only in events that do not affect any of the persistence links in the execution graph, then the goal will be satisfied in M. This graphical structure allows some of the independence of events and of action outcomes that may exist in the domain to be exploited for efficient planning.

If Weaver does discover persistence-threatening events, it is still possible to prove in some circumstances that the plan satisfies the goal. For each such event, if the preconditions P of the event are consistent with the state when the event begins, any extra fluents from P are added to the execution graph at the time point for this state. All event sequences may be considered that take place prior to this time stage and may make the preconditions of the event be satisfied. If no sequence of events is found that makes at least on persistence-threatening event possible, the plan is correct, by an extension of the previous proof. Such a search may be prohibitively expensive in general, but Weaver is able to use the effects of the events to restrict its search at each time point for events that may alter the state for relevant fluents only.

Note that one of the desired features mentioned in the introduction is satisfied: since external events are modelled in the same form as the actions, a planner can reason about their effects and causes in order to make use of them to achieve some goal. Although the planner cannot directly cause an event e to take place, it may be possible to produce a state s in which e would have a desired effect if it did take place. Such a plan could not be proved to satisfy the goal in the language of the previous section, but there may be a non-empty set of models in which the plan satisfies the goal. Ways to make use of plans that sometimes satisfy their goals are mentioned in section . Plans using external events can be created in Prodigy by relaxing the constraint that $Agent(a)$ be true for all actions a added to the tail-plan.

I note in passing that the concept of a *goal specification* must be refined for planners in domains with external events. Two possible refinements for the notion of reaching a goal state are (1) reaching a state s such that the goal condition is true in s and all subsequent states, or (2) reaching a state s where the goal condition is true, regardless of its value in subsequent states. Weaver uses the second, weaker definition but the arguments about efficiency made in this paper hold for either definition.

An extended example

In order to talk about planning with external events, I introduce an example planning domain concerned with reading a book in peace in my house. This requires being in a different room from my cat, which demands attention. I allow function terms to represent fluents and actions and represent the domain as follows.

First the actions *Getbook*, *Go*, *Close* and *Catmoves* are specified:

$$
\begin{aligned}
&Holds(Havebook, Result(Getbook, s))\\
&Holds(In(x), Result(Go(x), s)) \leftarrow Room(x)\\
&Holds(Cat(x), Result(Catmoves(x), s)) \leftarrow\\
&\qquad Room(x) \wedge Open(x) \wedge In(x)\\
&\neg Holds(Open(x), Result(Close(x), s)) \leftarrow Room(x)
\end{aligned}
$$

Next the domain objects:

$$Room(x) \Leftrightarrow [x = Kitchen \vee x = Diningroom]$$

Third, the initial state:

$$
\begin{aligned}
&\neg Holds(Havebook, S_0)\\
&Holds(Cat(Kitchen), S_0)\\
&Holds(In(Kitchen), S_0)
\end{aligned}
$$

Fourth, information about controllable actions and action durations:

$$
\begin{aligned}
&Agent(Getbook) \wedge Agent(Go(x)) \wedge Agent(Close(x)) \wedge\\
&Duration(Getbook) = 1 \wedge Duration(Close) = 1 \wedge\\
&Duration(Catmoves) = 1 \wedge Duration(Go) = 1
\end{aligned}
$$

and finally the goal is specified:

$$
\begin{aligned}
Goal \Leftrightarrow \exists sx[&Room(x) \wedge Holds(In(x), s)\\
&\wedge \neg Holds(Cat(x), s) \wedge Holds(Havebook, s)]
\end{aligned}
$$

Similarly to the example from the laundry domain, the two-step plan represented by the narrative $Happens(Getbook, 0, 1) \wedge Happens(Move(Diningroom), 1, 2)$ does not achieve the goal in every model of the domain. This is because there exists a model that includes the event occurrence $Happens(Catmoves(Diningroom), 1, 2)$. However, the three-step plan represented by $Happens(Close(Diningroom), 0, 1) \wedge Happens(Getbook, 1, 2) \wedge Happens(Move(Diningroom), 2, 3)$ does solve the plan. Weaver is able to find this plan, by modifying the plan above, and also a conditional plan that can be proved to satisfy the goal. This conditional plan is

$$
\begin{aligned}
&Happens(Getbook, 0, 1)\\
&Happens(Move(Diningroom), 1, 2) \leftarrow Holds(Cat(Kitchen), 1)
\end{aligned}
$$

Figure 3 shows the execution graph created by Weaver for this plan, which it uses to search for relevant events. Since it finds that events of type *Catmoves* are relevant, shown in the annotated execution graph in figure 4, lemma 1 cannot be used to show that the plan achieves the goals. The reason the plan does achieve its goals is that whenever these events can take place, the conditions under which the event changes the value of $Cat(Diningroom)$ are not satisfied. Weaver has added arcs labelled *stop-event* to show why these conditions do not hold.

Efficiency issues for the planner

A backward-chaining planner, whether partial-order or otherwise, must be able to evaluate a candidate plan for some goal efficiently. Whether goals can be satisfied transiently or must be satisfied permanently, the planner needs to effectively compute the effects of a plan across many branches of a tree of future states. In this section I describe three techniques that can improve the efficiency of this process. First, given a particular plan the independence implicit in the multiple events model can be used to ignore those events that cannot affect the outcome of the plan with respect to goal satisfaction. This approach effectively collapses many models of the plan's effects by ignoring unimportant differences. Second, a planner may attach information to outcomes of actions and events specifying which it believes are more likely than others. These can be combined according to laws of probability or Dempster-Shafer evidence combination, for example, and can often allow a planner to have a high confidence in plan success while examining a relatively small fraction of the possible futures. Third, efficiency gains can come from a careful management of the time spent searching for candidate plans versus the time spent evaluating them. It can often be the case that a candidate plan can be ruled out before even all the relevant external events have been identified. I describe the trade-offs involved in choosing where to spend computational effort below.

Exploiting independence in multiple events

The number of models for a given candidate plan is exponential in the plan's length, as the number of external events that may take place increases. While all the actions chosen by the planner are presumably relevant to the plan, many of the external events that can take place will have no relevance, because the fluents they alter do not affect the actions in the plan or the final goal. If these events can be ignored, the size of the tree can be exponentially reduced. The new set of models considered will not uniquely specify states that arise while the plan is executed, but will capture enough details to ensure the plan is correct.

The execution graph filters out some of the events that cannot affect the plan, and achieves this to a limited degree. In fact Weaver examines events that might threaten a persistence link further by checking if its preconditions are completely satisfied in its starting state, allowing for other

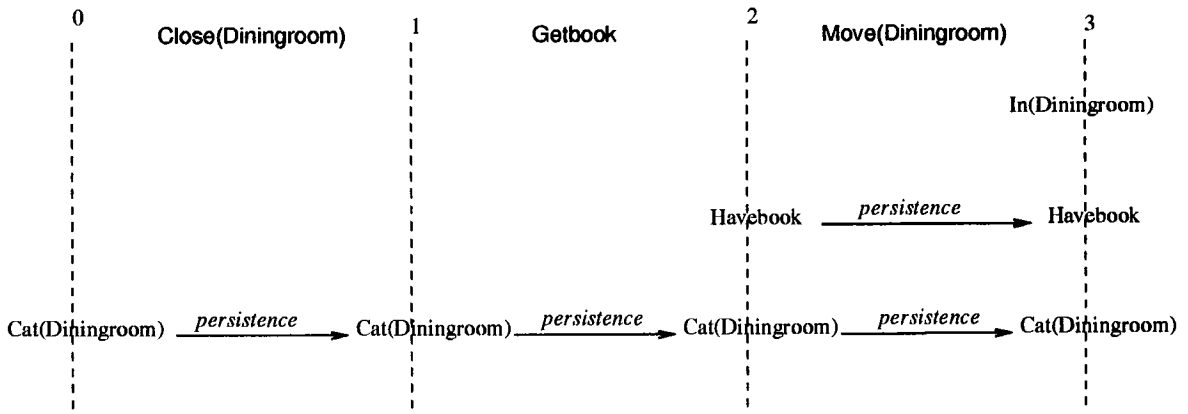

Figure 3: The first stage of creation of the execution graph, before external events are found. The graph contains no arcs labelled *effect* because there are no conditions necessary for the actions to achieve the effects in the graph.

events that might take place. Proving that this more strict filter for events is correct requires an extension to the proof presented here. I have also investigated a means to filter out irrelevant events based on a static analysis that can be made from the domain before planning begins (Blythe 1996).

The power of this approach depends on the use of small, relatively independent events, which may take place concurrently. This representation choice is therefore important to the model satisfying the third criterion mentioned in the introduction, allowing relevant events to be found efficiently. While the representation doesn't guarantee this, it does allow it.

Using a probabilistic model of the non-deterministic outcomes

In models of real-world domains, some of the possible outcomes of actions or events may often be considerably more likely than others, and in events with many possible outcomes, a small number of them may occur a high proportion of the time. This information can be used by a planner to both evaluate a plan, to within some error margin, more quickly, and to create plans more quickly by focussing on the more important scenarios in a candidate plan. The likelihoods of individual event or action outcomes can be combined to provide a likelihood that some property will hold after executing a plan, including goal satisfaction.

Knowledge about independence of events and actions can again be used to find more efficient techniques than summing branch probabilities over all possible branches of the event tree. Even though an event is relevant to the plan, it may not be relevant to every step, and in fact the goal structure of the plan leads directly to an independence structure for the events and actions that can be used to compute the probability of goal success in a Bayesian net (Kushmerick, Hanks, & Weld 1993). Indeed, the execution graph described in section is implemented as a Bayesian net in Weaver. Within the model of events described here, the use of probabilities is viewed as a technique for efficient planning rather than as a fundamental property of the action model. Thus non-deterministic planners that make no use of probabilities, such as Cassandra (Pryor & Collins 1993), could be extended to use the same action model.

Interleaving planning and plan evaluation

Since computational work is required to evaluate the effects of a plan, there is a spectrum of possible planning approaches based on the proportion of time spent searching for plans and the proportion of time spent evaluating plans. At one end of the spectrum, a planner might attempt to create a plan by backward chaining on actions and events, and ignoring the effects of external events until a complete plan is produced. At this point the relevant events are determined, using the techniques of section , and a reduced event tree or Bayesian net might be used to find the probability that the plan succeeds. (It will probably be low.) I refer to such a planner as a "greedy planner".

At the other end of the spectrum, a planner might reason about the external events that may affect each action when it is chosen. Since the earlier actions chosen by a backward chaining planner appear later in the plan, these event structures would not be identical to the final event tree corresponding to the appropriate point in the plan since the possible states are unknown. However some elements of the tree can be determined ahead of time, and these might allow the planner to prune useless partial plans much earlier than the greedy-planner would. I refer to this planner as a "greedy evaluator".

A truly efficient planner for dynamic domains will operate somewhere in the middle of this spectrum. The exact point depends on the number and the impact of the external events in the domain. In a domain with a large number of events which frequently turn out to be irrelevant, a more greedy planner will be appropriate. In a domain where external events are closely coupled with actions (for example their preconditions are satisfied by the effects of the action) and frequently interfere with plans, a more greedy evaluator will be appropriate.

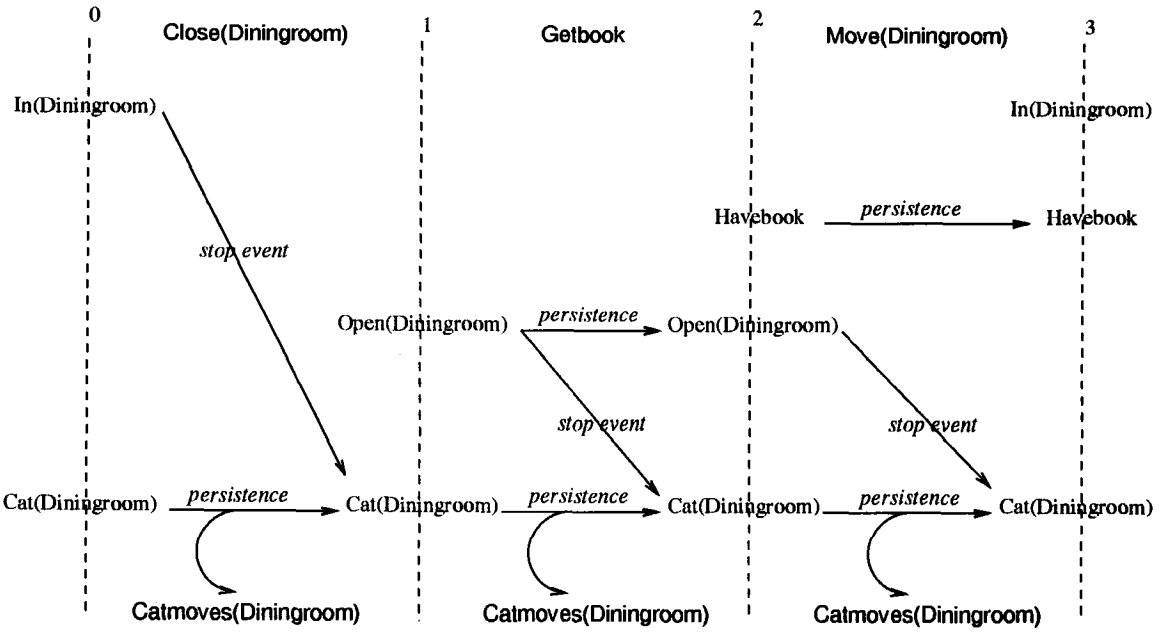

Figure 4: The execution graph, after events have been examined. The rounded lines show the events whose preconditions were added to the graph. These lines are not part of the graph itself.

Discussion

I have described a model for actions in dynamic domains, where the effects of events beyond the agent's direct control can be modelled as well as those of its actions. The model satisfies some useful criteria for the construction of efficient planners in these domains. The model of external events allows them to be incorporated in plans by backward chaining planners. It also allows efficient reasoning about the events relevant to a particular plan by exploiting their independence structure. The model can be described by a modified version of narratives in the situation calculus, described in (Miller & Shanahan 1994). Although it does not make full use of Miller and Shanahan's descriptions of simultaneous events and is less expressive than others that address similar issues, for example (Shanahan 1995), (Baral 1995), it is one of only a few currently used in an implemented planner. The model of non-deterministic action used here is similar to that of the Buridan planner (Kushmerick, Hanks, & Weld 1993).

There is a need both for more analysis of models like these and of appropriate planning algorithms and techniques, and for more empirical studies of such domains and planners. Weaver is currently a greedy planner that iteratively improves its plan. That is, it constructs a complete plan without considering external events, and then evaluates the plan with respect to the external events. It then attempts to repair its plan by either adding steps before events to make them inapplicable or adding steps after events to recover form their effects. In future work I will use the Weaver architecture as a platform to investigate the tradeoff between time spent planning and time spent evaluating plans in dynamic domains.

This short paper has ignored many interesting aspects of planning in dynamic, non-deterministic domains. In particular I have assumed complete observability and I assumed the planner creates entire plans before executing them. The model of time used here is very simple and there has been no account of temporal goal satisfaction (where constraints are placed on the time interval over which the goal is satisfied). However the language serves to illustrate some issues for planners and for action languages which will remain important in languages that handle these features.

References

Baker, A. B. 1991. Nonmonotonic reasoning in the framework of the situation calculus. *Artificial Intelligence* 49:5.

Baral, C. 1995. Reasoning about actions: Non-deterministic effects, constraints, and qualification. In *Proc. 14th International Joint Conference on Artificial Intelligence*, 2017–2024. Montréal, Quebec: Morgan Kaufmann.

Blythe, J. 1994. Planning with external events. In de Mantaras, R. L., and Poole, D., eds., *Proc. Tenth Conference on Uncertainty in Artificial Intelligence*, 94–101. Seattle, WA: Morgan Kaufmann.

Blythe, J. 1996. Decompositions of markov chains for reasoning about external change in planners. In Drabble, B., ed., *Proc. Third International Conference on Artificial Intelligence Planning Systems*. University of Edinburgh: AAAI Press.

Fink, E., and Veloso, M. 1995. Formalizing the prodigy planning algorithm. In Ghallab, M., and Milani, A., eds.,

New Directions in AI Planning, 261–272. Assissi, Italy: IOS Press.

Kartha, G. N. 1994. Two counterexamples related to baker's approach to the frame problem. *Artificial Intelligence* 61:379–391.

Kartha, G. N. 1995. *A Mathematical Investigation of Reasoning About Actions*. Ph.D. Dissertation, University of Texas at Austin.

Knoblock, C. A. 1993. *Generating Abstraction Hierarchies: An Automated Approach to Reducing Search in Planning*. Boston: Kluwer.

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.

Miller, R., and Shanahan, M. 1994. Narratives in the situation calculus. *Journal of Logic and Computation* 4(5).

Pryor, L., and Collins, G. 1993. Cassandra: Planning for contingencies. Technical Report 41, The Institute for the Learning Sciences.

Shanahan, M. 1995. A circumscriptive calculus of events. *Artificial Intelligence* 75(2).