Probabilistic Hybrid Action Models for Predicting Concurrent Percept-driven Robot Behavior

Michael Beetz
Dept. of Computer Science III,
University of Bonn
Roemerstr. 164,
D-53117 Bonn, Germany,
email: beetz@cs.uni-bonn.de

Henrik Grosskreutz
Department of Computer Science
Aachen University of Technology
D-52056 Aachen
Germany
grosskreutz@cs.rwth-aachen.de

Abstract

This paper develops ***Probabilistic Hybrid Action Models (PHAMs)***, a realistic causal model for predicting the behavior generated by modern concurrent percept-driven robot plans. PHAMs represent aspects of robot behavior that cannot be represented by most action models used in AI planning: the temporal structure of continuous control processes, their non-deterministic effects, and several modes of their interferences.

The main contributions of the paper are: (1) PHAMs, a model of concurrent percept-driven behavior, its formalization, and proofs that the model generates probably, qualitatively accurate predictions; and (2) a resource-efficient inference method for PHAMs based on sampling projections from probabilistic action models and state descriptions. We discuss how PHAMs can be applied to planning the course of action of an autonomous robot office courier based on analytical and experimental results.

Introduction

Most autonomous robots are equipped with restricted, unreliable, and inaccurate sensors and effectors and operate in complex and dynamic environments. A successful approach to deal with the resulting uncertainty is the use of controllers that prescribe the robot's behavior in terms of concurrent reactive plans — plans that specify how the robot is to react to sensory input in order to accomplish its jobs (cf. (McDermott 1992a; Beetz 1999)). Besides robustness foresight is another important capability of competent robots that are to perform complex and changing tasks in dynamic environments. Anticipating and forestalling possible problems requires the robots to predict what might happen when they execute their intended plans. To make such predictions they need models that represent how their concurrent reactive plans operate and might change the world.

So far most robot action planning systems use very abstract and coarse-grained models of their controllers. Most of them consider plans as partially ordered sets of atomic actions and thereby ignore powerful control abstractions for specifying synchronized and reliable behavior. These oversimplifi cations cause them to thwart many opportunities for improving the robots' behavior.

The action models used for planning also make unrealistic assumptions about the irrelevance of the temporal structure of actions and the exclusion of interferences between concurrent actions. Contrary to these assumptions, the actions of robots have extents in both space and time and the information available for planning is defi cient. The actions' extent and dependence on time require planning systems to predict when actions are executed, how long they take, how they overlap with concurrent actions, and how the world changes while they are performed. Often robot action planning systems must also be able to reason through contingencies whose likelihoods are a priori unknown and varying.

We believe that causal models for predicting the behavior generated by modern autonomous robot controllers accurately enough to foresee a signifi cant range of realistic execution problems must reflect that

... physical robot actions cause continuous change (they are movements!).
... controllers are reactive systems. The behavior and the effects depend on the situations that occur ***while*** actions are executed.
... most of the time the robot is executing multiple physical and sensing actions. Actions may interrupt or trigger each others or have combined effects.
... the robot is uncertain about the effects of its actions and the state of the environment.

In this paper we describe ***PHAMs*** that allow for the prediction of the qualitative behavior generated by ***concurrent reactive plans (CRPs)***. It models the interactions between multiple concurrent processes, the temporal structure of continuous processes, percept-driven control, and interference between effects of concurrent control processes. The main contributions of this paper are (1) PHAMs, a model of concurrent percept-driven behavior, its formalization, and proofs that the model generates probably, qualitatively accurate predictions; (2) a resource-effi cient prediction method for PHAMs based on sampling projections from probabilistic action models and state descriptions. We discuss how PHAMs can be applied to planning the course of action of an autonomous robot offi ce courier based on analytical and experimental results.

Concurrent Reactive Plans and the Specification of Navigation Behavior

We develop PHAMs in the context of a particular example: the navigation behavior of an autonomous robot. Predicting the path that the robot will take is not only necessary to predict where the robot will be. It is also a prerequisite for predicting what the robot will be able to sense. For example, whether the robot will ***learn*** that a door is open depends on the robot taking a path that passes by the door, executing the door angle estimation routine while passing the door, and the door being within sensor range. Consequently, if the robot executes a plan step only if a door is open then in the end the execution of this plan step strongly depends on the actual path the robot will take.

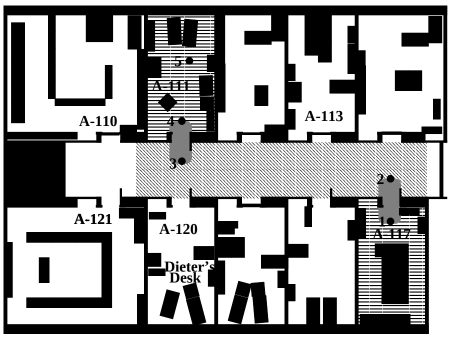

Figure 1: Graphical representation of a navigation plan.

After having seen why navigation behavior is so important for robot action planning we will now take a closer look at how navigation behavior is specified. Fig. 1 depicts an example of a navigation plan. The plan consists of two components (see (Beetz *et al.* 1998) for a more detailed description). The first one specifies a sequence of target points (the location indexed through the numbers 1 to 5 in Fig. 1) to be sequentially visited by the robot. The navigation between the target points is accomplished by a standard path planner (S. Thrun & Schmidt 1998). The second component specifies when and how the robot is to adapt its travel modes as it follows the navigation path. In many environments it is advantageous to adapt the travel mode to the surroundings: to drive carefully (and therefore slowly) within offices because offices are cluttered, to switch off the sonars when driving through doorways (to avoid sonar crosstalk), and to drive quickly in the hallways. This part of the plan is depicted through regions with different textures for the different travel modes "office," "hallway," and "doorway." Whenever the robot crosses the boundaries between regions it adapts the parameterization of the navigation system (Beetz *et al.* 1998).

CRPs are specified in an extended structured plan language (eg. RPL (McDermott 1991)) that provides high-level control structures for specifying concurrent, event-driven robot behavior. The pseudo code in Fig. 2 sketches a CRP for leaving an office that consists of two concurrent subplans: one for following the (initial part of the) prescribed path and one for adapting the travel mode. The second subplan adapts the navigation mode of the robot dynamically. Initially, the navigation mode is set to "office." Upon entering and leaving the doorway the navigation mode is adapted. The plan uses ***fluents***, conditions that are asynchronously updated based on new sensor readings. Fluents can trigger (*whenever*) and terminate (*wait-for*) plan steps.

```
execute concurrently
  execute-in-order
    GO-TO (1); wait for (go-to-completed?);
    GO-TO (2); wait for (go-to-completed?);
  execute-in-order
    with local fluents distance-to-doorway
                  ← fluent-network (| $\langle x, y \rangle - \langle x_{dw}, y_{dw} \rangle$ |)
      with local fluents entering-dw?-fl ← distance-to-doorway $<$ 1m
                         entering-hw?-fl ← distance-to-doorway $>$ 1m
        execute-in-order
          SET-NAVIGATION-STRATEGY(office); wait for (entering-dw?-fl);
          SET-NAVIGATION-STRATEGY(doorway); wait for (entering-hw?-fl);
          SET-NAVIGATION-STRATEGY(hallway)
```

Figure 2: Simple navigation plan: pseudo code

Projecting Concurrent Reactive Plans

So, how can the operation of CRPs and their effects be predicted? Numerical simulation methods, an obvious candidate, are very expensive in terms of computational resources and produce results too detailed for high-level planning. Computationally more economical alternatives are methods for symbolic temporal projection that only predict those discrete events that are relevant for determining the robot's course of action.

Suppose we would like to symbolically predict the navigation behavior that is generated by the navigation plan that is depicted in Fig. 1. The most parsimonious causal model is the one that represents the resulting behavior as a single atomic action that has a single effect, namely changing the robot's location from its current one to the destination of the navigation plan. More detailed representations hierarchically decompose the navigation behavior into smaller actions like leaving the room, passing the doorway, going over the hallway, and entering the office. This kind of model can represent that entering the office is only possible if the corresponding door is open. If the door is closed the execution of the navigation plan will result in the robot standing in front of the closed door.

Probabilistic Causal Models of CRPs

In this paper we develop PHAMs as a novel kind of symbolic causal models, which are more realistic. PHAMs can make qualitatively accurate predictions of the behavior generated by CRPs. This is possible because PHAMs can model

- continuous change,
- reactive control processes,
- multiple interacting control processes, and
- uncertainty about physical behavior and world state.

We will discuss the importance of these features and the issues they raise in the remainder of this section.

Continuous Change and Reactive Control Processes. CRPs activate and deactivate control processes and thereby continuously change states like the robot's position. At the same time, sensing processes continually measure and estimate relevant states (eg, the robot's position or the opening angle of a door) and signal when these estimated states satisfy conditions that are monitored by the CRP. As a consequence, signals sent by the sensing processes and feedback

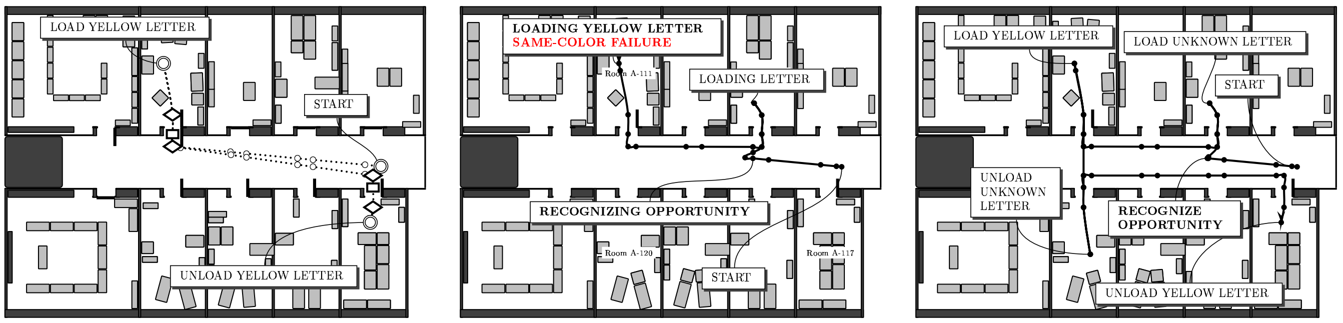

Figure 3: Different projected scenarios for the plan discussed in Section "Example Scenario."

from the control processes cause the activation and deactivation of control processes.

The reactive nature of CRPs has an important consequence for the required expressiveness of causal models. For reactive control, the events that have to be predicted for a navigation action depend not only on the action itself but also on the monitoring processes that are active and wait for conditions that the continuous effects of the navigation action might cause. Suppose that the robot controller runs a monitoring process that stops the robot as soon as it passes an open door. In this case the planner must predict "robot passes door" events for each door the robot passes during a continuous navigation action. These events then trigger a sensing action that estimates the door angle, and if the predicted percept is an "open door detected" then the navigation process is deactivated. Other discrete events that might have to be predicted based on the continuous effects of navigation include entering and leaving a room, having come within one meter of the destination, etc.

Multiple interacting control processes. Because in the end all control processes set voltages for the robot's motors, the possible modes of interference between control processes are limited. If they generate signals for the same motors the combined effects are determined by the so-called *task arbitration scheme* (Arbib 1998). The most common task arbitration schemes are (1) behavior blending (where the motor signal is a weighted sum of the current input signals) (Konolige *et al.* 1997); (2) prioritized control signals (where the motor signal is the signal of the process with the highest priority) (Brooks 1986); and (3) exclusion of concurrent control signals through the use of semaphores. Any of these schemes can be handled by an appropriate projection algorithm for combined effects. In the CRPs considered in this paper we exclude multiple control signals to the same motors through semaphores (see (Beetz 1999) for a motivation).

Thus the only remaining type of interference between different processes is the superposition of movements caused by different motors. Consider for example a control mode in which the robot turns its camera at a speed of 0.5π per second and travels at a speed of 40cm per second along the hallway. Again the robot is to stop as soon as it perceives an open door. To predict when and where the robot will stop the planner has to consider the superposition of the continuous effects of the navigation and moving camera processes.

Uncertainty. There are various kinds of uncertainty and non-determinsm in the robot's actions and the environment that a causal model should represent. It is often necessary to specify a probability distribution over the average speed and the displacements of points on the paths to enable models to predict the range of spatio-temporal behavior that a navigation plan can generate.

Another important issue is to model probability distributions over the occurrence of exogenous events. In most dynamic environments exogenous events such as opening and closing doors might occur at any time. This can be modelled, for example, using Poisson distributed exogenous events. Modelling such events is almost impossible within classical frameworks.

The Expressiveness and Inferential Power of PHAMs

Figure 3(left) shows a projected execution scenario for a CRP and the different kinds of events that are predicted. The events depicted by rhombs denote events where the CRP changes the direction and the target velocity of the robot. The squares denote the events entering and leaving offices. The small circles denote the events start and finished passing a door, which are predicted because a concurrent monitoring process estimates the opening angles of doors ***while*** the robot is passing them.

To illustrate the inferential power of PHAMs let us consider an autonomous robot courier. The robot is to deliver a yellow letter from room A-111 to A-117 and another letter for which the color is not known from A-113 to A-120 (*cmd-2*). Suppose the robot carries out *cmd-2* opportunistically. That is, if it detects that A-113 is open it will interrupt and reconsider its plan under the premise that the steps for accomplishing *cmd-2* are executable.

Executing this plan might yield mixing up letters because the robot might carry two letters with the same color. To find out whether a plan will probably work or might yield mixing up letters, the robot must predict the possible courses of action and their results. Suppose the robot's belief state assigns probability *p* for the value true of random variable open-A113. The belief state also contains probabilities for the colors of letters on the desk in A-113.

Wrt. this belief state a number of different execution scenarios for a given CRP are possible. Fig. 3 shows three of them. The first one, in which A-113 is closed, is pictured in Fig. 3(left). In the scenarios in which office A-113 is open the controller is projected to recognize the opportunity and to change its intended course of action. The resulting schedule asks the robot to first enter A-113, and pickup the letter for cmd-2, then enter A-111 and pick up the letter for cmd-1, then deliver the letter for cmd-2 in A-120, and the last one in A-117. This category of scenarios can be further divided into two categories. In the first subcategory shown in Fig. 3(middle) the letter to be picked up is yellow. Performing the pickup thus would result in the robot carrying two yellow letters and therefore an execution failure is signalled. In the second subcategory shown in Fig. 3(right) the letter has another color and therefore the robot is projected to succeed by taking for all these scenarios the same course of action.

Based on estimations of the probabilities of the different scenarios the planner can decide whether the plan should be revised and how (Beetz, Bennewitz, & Grosskreutz 1999).

Modeling Reactive Control Processes and Continuous Change

Before we model the navigation behavior generated by CRPs we will first introduce some vocabulary for describing the systems operation. We consider the interpretation of the CRP and the robot together with its operating environment as two dynamic systems and describe the state of the robot and its navigation system through a number of state variables including state variables *x*, *y*, and *o* describing the robot's position and orientation in the environment and the variables *v* and ω for the robot's translational and rotational speed. The navigation plans exert control on the state variables *v* and ω in order to achieve a state in which the robot is at its destination. There are also variables representing the state of the environment, one for the opening angle of each door and the locations of the letters to be delivered.

Because the concurrent reactive plan activates and deactivates control processes asynchronously we need action models that can represent both discrete and continuous behavior. In the following, we will use the vocabulary of *hybrid systems* (Alur, Henzinger, & Wong-Toi 1997; Alur, Henzinger, & Ho 1996) to represent this aspect of concurrent reactive plans. Hybrid systems are continuous variable, continuous time systems with a phased operation. Within each phase, called *control mode*, the system evolves continuously according to the dynamical law of that mode, called *continuous flow*. Thus the state of the hybrid system can be thought of as a pair — the control mode and the continuous state. The control mode identifies a flow, and the continuous flow identifies a position in it. With each control mode there are also associated so-called *jump conditions*, that specify the conditions that the discrete state and the continuous state together must satisfy to enable a transition to another control mode. Once the transition occurs, the discrete state and the continuous state are changed abruptly. The *jump relation* specifies the valid settings of the system variables that might occur during a jump. Then, until the next transition, the continuous state evolves according to the flow identified by the new control mode.

When considering the interpretation of CRPs as a hybrid system the control mode is the set of active control processes and their parameterization. The continuous state is the state of the system variables *x*, *y*, and *o*. The continuous flow describes how these state variables change as a result of the combination of the continuous effects of the active control processes. We assume that the state variables change only linearly over time. Thus for each control mode *v* and ω are constant. Linear flow conditions are sufficient because the robot's paths can be approximated accurately enough using polylines (Beetz & Grosskreutz 1998). They are also computationally much easier and faster to handle. The jump conditions are the conditions that are monitored by monitoring processes which activate and deactivate control processes and the termination of control processes such as arriving at an intermediate point of the navigation path.

Thus the interpretation of a navigation plan according to the hybrid systems model works as follows. The hybrid system starts at some initial state $\langle cm_0, x_0 \rangle$. The state trajectory evolves with the control mode remaining constant and the continuous state x evolving according to the flow condition of cm. When the continuous state satisfies the transition condition of an edge from state cm to a state cm' a jump can be made to state cm'. During the jump the continuous state may get initialized to a new value x'. The new state is the pair $\langle cm', x' \rangle$. The continuous state x' evolves according to the flow condition of cm'.

For now we simply pretend that the hybrid automata can be generated for arbitrary CRPs and assume that the hybrid automata are given. In the section "Probabilistic sampling-based projection" we will see that only a small fraction of the hybrid automaton is needed, which can be constructed on the fly.

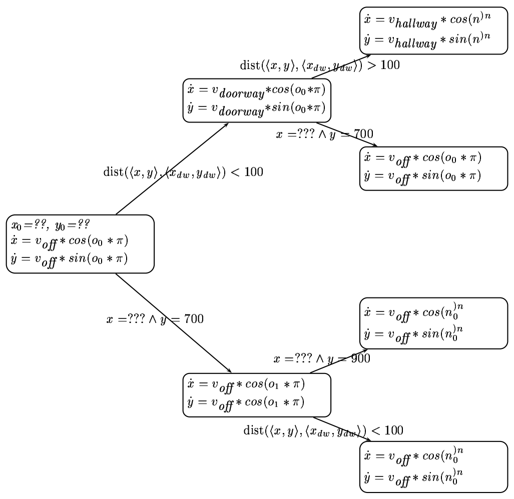
$\dot{x} = v_{hallway} * cos(n)^n$
$\dot{y} = v_{hallway} * sin(n)^n$
$dist(\langle x,y \rangle, \langle x_{dw}, y_{dw} \rangle) > 100$
$\dot{x} = v_{doorway} * cos(o_0 * \pi)$
$\dot{y} = v_{doorway} * sin(o_0 * \pi)$
$x = ??? \wedge y = 700$
$\dot{x} = v_{off} * cos(o_0 * \pi)$
$\dot{y} = v_{off} * sin(o_0 * \pi)$
$dist(\langle x,y \rangle, \langle x_{dw}, y_{dw} \rangle) < 100$
$x_0 = ??, \ y_0 = ??$
$\dot{x} = v_{off} * cos(o_0 * \pi)$
$\dot{y} = v_{off} * sin(o_0 * \pi)$
$x = ??? \wedge y = 700$
$\dot{x} = v_{off} * cos(n_0)^n$
$\dot{y} = v_{off} * sin(n_0)^n$
$x = ??? \wedge y = 900$
$\dot{x} = v_{off} * cos(o_1 * \pi)$
$\dot{y} = v_{off} * cos(o_1 * \pi)$
$dist(\langle x,y \rangle, \langle x_{dw}, y_{dw} \rangle) < 100$
$\dot{x} = v_{off} * cos(n_0)^n$
$\dot{y} = v_{off} * sin(n_0)^n$

Figure 4: Hybrid automaton.

Fig. 4 depicts the interpretation of the first part of the navigation plan shown in Fig. 2. The interpretation is represented as a tree where the nodes represent the control modes of the corresponding hybrid system and the node labels the continuous flow. The edges are the control mode transitions labeled with the jump conditions. The robot starts executing the plan in room A-117. The initial control mode of the hybrid system is the root of the state tree depicted in Fig. 4. The initial state represents the state of computation where the first control processes of the two parallel branches are active, that is the processes for going to the intermediate destination **1** and maintaining the "office mode" as the robot's travel mode. The flow specifies that while the robot is in the initial control mode the absolute value of the derivative of the robot's position is v_{off}. The hybrid system transits into one of the subsequent states when either the first intermediate destination is reached or when the distance to the doorway becomes less than one meter. The transition condition for the upper edge is that the robot has come sufficiently close to the doorway, for the lower edge that it has reached the first intermediate destination. For the upper edge, the hybrid system goes into the state where the robot goes to the intermediate destination **2** while still keeping the office mode as its current travel mode. In the other transition the robot changes its travel mode to doorway and keeps approaching the first intermediate destination. The only variables that are changed through the control mode transitions are the velocity of the robot and its orientation. Both settings are implied by the flow condition of the respective successor states.

The Probabilistic Rule Language

In the last section we have introduced the basic vocabulary to represent the interpretation of CRPs and continuous and reactive control processes. We can see that hybrid systems represent all possible sequences of computational state transitions of the navigation plan. Some of the state sequences, however, are physically not possible. For example, changing the travel mode from office to doorway after the second intermediate destination has been reached. In addition, CRPs generate different mode sequences with different frequencies (probabilities). If the goal of action planning is to make the plan more robust then the planner should reason about the conditional probabilities of state sequences with respect to the robot's beliefs. In this section we will introduce McDermott's probabilistic rule language (1994) as a means for representing probabilistic effects of robot actions and nondeterministic exogenous events.

In McDermott's rule language causal models are represented by

$p{\rightarrow}e$ rule *name*	$e{\rightarrow}p$ rule *name*
while *cond*	if *cond*
with an avg spacing of τ	then with probability θ
occurs *ev*	event *ev*
	causes *effs*

Exogenous event rules ($p{\rightarrow}e$) generate over any interval in which *cond* is true, random "Poisson-distributed" events *ev* with an average spacing of τ time units (see (McDermott 1994)). **Effect rules** ($e{\rightarrow}p$ rules) specify that whenever event *ev* occurs and *cond* holds, then with probability θ the effects of *ev* are specified by *effs*. The effects have the form *p*, causing the proposition *p* to hold in the resulting world state and clip *p*, terminating the persistence of *p*.

A consistent set of probabilistic rules together with a set of dated plan generated events define a probability distribution over execution scenarios. An execution scenario describes how the execution of a robot controller might go, that is, how the environment changes as the plan gets executed. It is represented as a *timeline*, a linear sequence of *occurrences* where an occurrence is a dated *event*. The events of occurrences cause new *world states* that are characterized by the set of propositions that hold in them. Actions are events that are caused by the robot.

Before proving properties of our model we must first introduce McDermott's semantics of possible worlds.

Definition 1 *A* ***world state*** *is a function from propositions to* $\{T,F,\perp\}$ *and is extended to boolean formulas in the usual way. An* ***occurrence*** *$e@t$ is a pair $c = (e,t)$, where e is an event and t a time ($t \in \Re_+$). An* ***occurrence sequence*** *is a finite sequence of occurrences, ordered by date. Its duration is the date of the last occurrence. A* ***world of duration L****, where $L \in R_+$, is a complete history of duration L, that is, it is a pair (C,H), where C is an occurrence sequence with duration $\leq L$, and H is a function from $[0,L]$ to world states. In $H(0)$ all propositions are mapped to F, and if $t1 < t2$ and $H(t1) \neq H(t2)$ then there must be an occurrence $e@t$ with $t1 \leq t \leq t2$.*

We use the following abreviations: $(A \downarrow t)(W)$, "A after t in W", to mean that there is some $\delta > 0$ such that $\forall t' : t < t' < t + \delta \Rightarrow H(t')(A) = T$. $(A \uparrow t)(W)$, "A before t in W" is similarly defined, but the upper bound for t' includes t. $A \uparrow c$ and $A \downarrow c$ refer to the date of the occurrence c.

Definition 2 *If T is a set of rules as defined above, $Exog$ is an occurrence sequence, and L is a real number $\geq duration(Exog)$, then an* ***L-Model of T and Exog*** *is a pair (U,M), where U is a set of worlds of duration L such that $\forall (C,H) \in U : Exog \subset C$, and M is a probability measure on U that obeys the following restrictions: $(A{\downarrow}t)$, $(A{\uparrow}t)$ and $e@t$ are considered as random variables. $\overline{A}$ is the "annihilation" of a conjunction A, that is the conjunction of the negations of the conjuncts of A.*

1. ***Initial blank state:*** $\forall A \in P : M(A \uparrow 0) = 0$.
2. ***Event-effect rules:*** *If T contains a rule instance $A/e\overset{r}{\rightarrow}B$ then for every date t, require that, for all nonempty conjunctions C of literals from B: $M(C{\downarrow}t|E@t \wedge A{\uparrow}t \wedge \overline{B}{\uparrow}t) = r$.*
3. ***Event-effect rules when the events don't occur:*** *Suppose B is an atomic formula, and let $R = \{R_i\}$ be the set of all instances of $./.{\rightarrow}B'$ rules whose consequents contain B or $\neg B$. If $R_i = A_i/E_i\overset{p_i}{\rightarrow}C_i$, then let $D_i = A_i \wedge \overline{C_i}$, Then $M(B{\downarrow}t|B{\uparrow}t \wedge N) = 1$ and $M(B{\downarrow}t|\neg B{\uparrow}t \wedge N) = 0$ where $N = (\neg E_1@t \vee \neg D_1) \wedge (\neg E_2@t \vee \neg D_2) \wedge ....$*
4. ***Event-occurrence rules:*** *For every time point t such that no occurrence with date t is in $Exog$ and every event E, such that there is exactly one instance $A\overset{d}{\rightarrow}E$ with $M(a{\uparrow}t) > 0$ require*
$$lim_{dt\rightarrow 0}\frac{M(\textit{occ. of class E between t and t+dt}|A{\uparrow}t)}{dt} = 1/d$$
$$lim_{dt\rightarrow 0}\frac{M(\textit{occ. of class E between t and t+dt}|\neg A{\uparrow}t)}{dt} = 0$$
if there exists no so such rule, require
$$lim_{dt\rightarrow 0}\frac{M(\textit{occ. of class E between t and t+dt})}{dt} = 0$$
5. ***Conditional independence:*** *If one of the previous clauses defines a conditional probability $M(\alpha|\beta)$, which mentions times t, then α is conditional independent, given β, of all other random variables mentioning times on or before t. That is, for arbitrary γ mentioning times on or before t, $M(\alpha|\beta) = M(\alpha|\beta \wedge \gamma)$.*

McDermott (1994) shows that this definition yields a unique probability distribution M.

Probabilistic Hybrid Action Models

Let us now explain the predicates and rules that represent the operation of CRPs and their physical effects.

Conceptualization of PHAMs

McDermott's probabilistic rule language provides us with mechanisms for modelling probabilistic effects and Poisson distributed exogenous events. We still need to model another kind of uncertainty for projecting CRPs: probabilistic control mode transitions, which enable us to model, among other things, the variance of behavior exhibited by real robots when controlled by the same configuration of control routines. To do this we extend the basic representation of hybrid systems by introducing probability distributions over jump relations.

We will use the following notations to describe the interpretation of navigation plans using the vocabulary of hybrid systems: *jumpCondition(cm,e,c)*, *jumpSuccessor(e,cm',probRange)*, *jumpRelation(cm',$\vec{vals}$, $\vec{flows}$)*, and *probRange(e,max)*. *jumpCondition(cm,e,c)* represents that control mode *cm* is left along edge *e* when the condition *c* becomes true. A jump *e* causes the automaton to transit probabilistically into a successor mode. To represent that the transitions are mutually exclusive we specify the probabilities through probability ranges. Thus, each possible successor is associated a probability range *probRange*. For reasons that are explained below we represent the probability ranges such that their relative sizes are preserved, the sum of their ranges is 1, and that their boundaries have the form $\frac{i}{2^n}$, where *i* and *n* are integers. The predicate *probRange(e,*2^n*)* defines the sum of the ranges. A possible transition with a probability range $[\frac{i}{2^n}, \frac{j}{2^n}]$ is represented as *jumpSuccessor(e,cm',[i,j])*. The predicate *jumpRelation(cm',$\vec{vals}$, $\vec{flows}$)* means that upon entering control mode *cm'* the system variables and flows are initialized as specified by $\vec{vals}$ and $\vec{flows}$.

The following example illustrates the above predicates to describe transition conditions etc.

jumpRelation(cm_0,⟨2400,800⟩,⟨30,80⟩)
jumpCondition(cm_0,e_1,$y \geq 900$)
jumpSuccessor(e_1,cm_1,(0,11))
jumpSuccessor(e_1,cm_2,(12,15))
probRange(e_1,16)
jumpRelation(cm_1,⟨ ⟩,⟨30,80⟩)
jumpRelation(cm_2,⟨ ⟩,⟨40,76⟩)
...

The robot starts at position $\langle 800, 2400\rangle$ in control mode cm_0 in which the robot leaves the lower office on the right. In this control mode the robot moves with 30cm/s into the x-direction and with 80cm/s in the y-direction. The navigation system leaves control mode cm_0 as soon as the y coordinate becomes greater than 900 by performing the transition e_1. If the system performs the transition e_1 then with 75% (12/16) probability the system transits into control mode cm_1 and with 25% (4/16) probability into the mode cm_2. The flow condition in cm_1 is $\dot{x} = 30$ and $\dot{y} = 80$ and in cm_2 $\dot{x} = 40$ and $\dot{y} = 76$.

To predict the state of a hybrid automaton we use the predicates *mode(cm)* and *startTime(cm,t)* to represent that the current control mode is *cm* and that *cm* started at time *t*. We use *flow($\vec{flow}$)* and *valuesAt(t_i,$\vec{val}_i$)* to assert the flows and values of system variables for given time points. Further, the values of system variables can be inferred for arbitrary time points through interpolation on the basis of the current flow and the last instances of *valuesAt(t_i,$\vec{val}_i$)*. This is done using the predicate *stateVarsVals*:

$$stateVarVals(\vec{vals}) \equiv \text{valuesAt}(t_0,\vec{vals}_0) \wedge now(t) \\ \wedge flow(\vec{flow}) \wedge \vec{vals} = \vec{vals}_0 + (t - t_0)\vec{flow}$$

where *now(t)* specifies that *t* is the current time (see below).

These predicates are used to define the occurrence sequence *Exog* and a set of rules *T*, which according to Definition 2, imply an L-model of *T*, where *L* is a duration sufficiently large for the prediction of a plan's effects. The L-model is the probability distribution over possible execution scenarios that reflects the execution of a CRP.

Exog consists of the "initialization" event, that causes the initial state of the execution scenario, which contains the specification of the hybrid automaton, the initial control mode together with the initial values of the state variables and flows, and a set of clock ticks that occur every dt_{clock} time units to include a notion of time into the scenarios.

Probabilistically sampled control mode jumps. Let us now discuss our approach to predicting the changes of the control mode of the automaton over time. The complication with the prediction of control mode is that the hybrid automaton defines mutually exclusive control mode transitions that occur with different probabilities. In the light of definition 2 this means that the distribution over execution scenarios of duration *L* must reflect the probability distributions over control mode transitions. As a means of formulating the rules that represent control mode transitions we will use the predicate *randomlySampledSuccessorMode(e,cm)*:

$$randomlySampledSuccessorMode(e,cm) \equiv \\ probRange(e, max) \wedge randomNumber(n, max) \\ \wedge jumpSuccessor(e, cm, range) \wedge n \in range$$

In order to probabilistically sample values from probability distributions we have to axiomatize a random number generator that asserts instances of the predicate *randomNumber(n,max)* that was used above. We do this by formalizing a *randomize* event.[1]

Lemma 1 *At any time point randomNumber has exactly one extension randomNumber(r,max) where r is an unbiased random between* 0 *and* max.

Proof: Let max* be the largest *probRange* extension and *randomBit(i,value)* the i-th random bit. The start event that causes the initial state timeline causes *randomBit(i,0)* $\forall 0 \leq i \leq \log max^*$. Thereafter, a $randomize$ event is used to sample their value:

[1]McDermott ((1994), page 26) discusses the usefulness of, and the difficulties in realizing, nondeterministic exclusive outcomes. Therefore in his implementation he escapes to Lisp and uses a function that returns a random element.

```
e→p rule RANDOMIZE
if randomBit(i,val) ∧ negation(val,neg)
then with probability 0.5
     event randomize
     causes randomBit(i,neg) ∧ clip randomBit(i,val)
```

Whenever a rule queries for a random number it resamples the number in an analogous way. Note, the probability of $randomBit$, after a randomize event is conditionally independend of its former value. The Lemma holds per induction over the number of randomize events. □

The Probabilistic Causal Rules of PHAMs

Using the rule language described above and the predicates introduced in the last section, we will now specify the causal model of CRPs. As the semantics of McDermott's algorithm is not full first-order, we treat the following rules as schemata standing for all their ground instances.

Rule MODE-JUMP causes a control mode transition as soon as the jump condition *cond* becomes true.

```
p→e rule MODE-JUMP
while mode(cm) ∧ jumpCondition(cm,cond,edge)
        ∧ stateVarsVal($\vec{vals}$) ∧ satisfies(vals,cond)
with an average spacing of $\tau$ time units
occurs jump(edge)
```

Rule JUMP-EFFECTS specifies the effects of an jump event on the control mode, system variables, and the flow.

```
e→p rule JUMP-EFFECTS
if randomlySampledSuccessorMode(edge,cm)
     ∧ initialValues(cm,$\vec{val}$) ∧ flowCond(cm, $\vec{flow}$) ∧ now(t)
     ∧ mode(cm$_{old}$) ∧ flow(flow $_{old}$) ∧ valuesAt(t$_{old}$,val$_{old}$)
then with probability 1.0
     event jump(edge)
     causes mode(cm) ∧ flow( $\vec{flow}$) ∧ valuesAt(transTime,$\vec{val}$)
              ∧ clip mode(cm$_{old}$) ∧ ...
```

The *now* predicate is updated according to the CLOCK-TICK(*?t*) event by clipping the previous time and asserting the new one. Note, the time differs at most dt_{clock}time units from the actual time.

```
e→p rule CLOCK-RULE
if now(t$_o$)
then with probability 1.0
     event clock-tick(t)
     causes now(t) ∧ clip now(t$_o$)
```

Exogenous events are modelled using rules of the following structure: $\boxed{exoEventCond(cm,con,ev)}$

```
p→e rule CAUSE-EXO-EVENT
while mode(cm) ∧ exoEventCond(cm,cond,ev)
        ∧ stateVarsVal($\vec{vals}$) ∧ satisfies(vals,cond)
with an average spacing of $\tau$ time units
occurs exoEvent(ev)
```

The effects of exogenous event rules are specified by rules of the following form.

```
e→p rule EXO-EVENT-EFFECT
if exoEffect(ev,$\vec{val}$)) ∧ valuesAt(t$_o$,val$_o$) ∧ now(t)
then with probability 1.0
     event exoEvent(ev)
     causes valuesAt(t,$\vec{val}$) ∧ clip valuesAt(t$_o$,$\vec{val}_o$)
```

Properties of PHAMs

We have seen in the last section that a PHAM consists of the rules above and a set of facts that constitutes the hybrid automata representation of a given CRP as it has been introduced in section "Conceptualization". In this section we investigate whether PHAMs make the "right" predictions.

Because McDermott's formalism does not allow for modelling instantaneous state transitions we can only show that control mode sequences in execution scenarios are probably approximately accurate. In our view, this is a low price for the expressiveness we gain through the availability of Poisson distributed exogenous events.

Lemma 2 *For each probability ϵ and delay δ, there exists a τ (average delay of the occurrence of an event after the triggering condition has become true) and a dt_{clock} (time between two subsequent clock ticks) such that whenever a jump condition becomes satisfied, then with probability $\geq 1-\epsilon$ a jump event will occur within δ time units.*

Proof: *Let t be the time where the jump condition is fulfilled. If $\tau \leq \delta/(2\log(1/\epsilon))$ and $dt_{clock} \leq \delta/2$ then at most $\delta/2$ time units after t the antecedent of rule* MODE-JUMP *is fulfilled. The probability that no event of class $jump(cm')$ occurs between $t+\delta/2$ and $t+\delta$ is $\leq e^{-\delta/(2\tau)} = e^{-log(1/\epsilon)} = \epsilon$, so with probability $\geq 1-\epsilon$ such an event will occur at most δ time units after t.* □

This implies that there is always a non-zero chance that control mode sequences are predicted incorrectly. This happens when two jump conditions become true and the jump triggered by the later condition occurred before the other one. However, the probability of such incorrect predictions can be made arbitrarily small by the choice of τ and dt_{clock}.

The basic framework of hybrid systems does not take the possibility of exogenous events into account and thereby allows for proving strong system properties such as the reachability of goal states from arbitrary initial conditions or safety conditions for the system behavior (Alur, Henzinger, & Wong-Toi 1997; Alur, Henzinger, & Ho 1996). For the prediction of robot behavior in dynamic environments these assumptions, however, are unrealistic. Therefore, we only have a weaker property, namely that only between immediate subsequent events the predicted behavior corresponds to the flows specified by the hybrid system.

Lemma 3 *Let W be an execution scenario, $e_1@t_1$ and $e_2@t_2$ be two immediate subsequent events of type jump or exoEvent, and cm be the control mode after t_1 in W. Then, for every occurrence e@t with $t_1 < t \leq t_2$ W(t)(stateVarVals($\vec{vals}$)) is unique. Further, $\vec{vals} = \vec{vals}_1$ + (t - t_1) * flow(cm), where $\vec{vals}_1$ are the values of the state variables at t_1.*

Proof Idea: *There are only two classes of rules that affect the value of valuesAt and flow: rule* JUMP-EFFECTS*, and rule* EXO-EVENT-EFFECT. *These rules always clip and set exactly one extension of the predicates, thus together with the fact that the initial event asserts exactly one such predicate, the determined value is unique.*

During the interval t_1 to t_2 the extension of stateVarVals evolves according to the flow condition of mode cm due to the fact that flow is not changed by rule EXO-EVENT-EFFECT. *Thus it remains as* *initially set by rule* JUMP-EFFECTS*, which asserts exactly the flow corresponding to cm. The proposition then follows from the assumption of a correct axiomatization of addition and scalar-vector multiplication.*

□

Another important property of our representation is that jumps are predicted according to the probability distributions specified for the hybrid automaton.

Lemma 4 *Whenever a jump along an egde e occurs, the successor state is chosen according to the probability distribution implied by probRange and jumpSuccessor.*

Proof Idea: *This follows from lemma 1 and Rule Jump-Effects.*

□

Using the lemmata we can state and show the central properties of PHAMs: (1) the predicted control mode transitions correspond to those specified by the hybrid automaton; and (2) the same holds for the continuous predicted behavior in between exogenous events; (3) Exogenous events are generated according to they probabilities over a continuous domain (this is shown in (McDermott 1994)).

Theorem 1 *Every sequence of $mode(cm)$ occasions follows a branch $(cm_i), ..., (cm_j)$ of the hybrid automaton.*

Proof Idea: *Each occasion mode(cm) must be assert by rule* JUMP-EFFECTS*. Therefore there must have been a jump(e) event. Therefore, there must have been a jumpCondition from the previous control mode to cm.*

□

Theorem 2 *For every ϵ there exists a τ and a dt_{clock} such that with probability $\geq 1 - \epsilon$ the $\vec{vals}$ of $stateVarVals$ occasions between two immediate subsequent exogenous event follow a state trajectory of the hybrid automaton.*

Proof Idea: *The proof is based on the property that jumps occur in their correct order with an arbitrary high probability. In particular, we can choose δ as a function of the minimal delay between jump conditions becoming true. Then, the jumps to successor modes occur with arbitrarily high probability (Lemma 2). Finally, according to Lemma 3 the trajectory of stateVarVals between transitions is accurate.*

□

Probabilistic Sampling-based Projection

We have now shown that PHAMs define probability distributions over possible execution scenarios with respect to a given belief state. The problem of using PHAMs is obvious. Nontrivial CRPs for controlling robots reliably require hundreds of lines of code. There are usually several control processes active, many more are dormant waiting for conditions that trigger their execution. The hybrid automata for such CRPs are huge, the branching factors for mode transitions are immense. Let alone the distribution of execution scenarios that they might generate. The accurate computation of this probability distribution is prohibitively expensive in terms of computational resources.

In this section we investigate how we can make effective and informative predictions on the basis of PHAMs that can be performed at a speed that suffices for prediction-based online plan revision.

Recently, probabilistic sampling-based inference methods have been proposed to infer information from such complex distributions quickly and with bounded risk (Fox *et al.* 1999; Thrun 1999). To predict the effects of executing CRPs we use an extension of the XFRM projector (McDermott 1992b) that employs the RPL interpreter (McDermott 1991) together with PTOPA, McDermott's algorithm for probabilistic temporal projection (1994). The projector takes as its input a CRP (written in RPL), rules for generating exogenous events, a set of probabilistic rules describing the effects of events and actions, and a (probabilistic) initial state description. The projector works exactly like the interpreter, except that instead moving the robot it uses its causal models to predict the combined effects of the control processes that are active at a time. McDermott (1994) gives a proof that PTOPA projects random execution scenarios sampled from the probability distribution implied by the given probabilistic models.

For the projection of complex CRPs we do not need to represent the complete hybrid automaton for a CRP in advance. Rather the PHAM projector extends the XFRM projector so that it generates the parts of the PHAM that are needed to project a sample scenario on the fly (see (Beetz & Grosskreutz 1998) for details). To do so, the PHAM projector extracts the jump conditions *jumpCondition* from the conditions the interpreter is waiting for (for example, the ***wait for*** statements in the CRP shown in Fig. 2). It also signals predicted control mode jumps to and thereby tells the interpreter component of the projector to continue with the projection of the subsequent plan steps. *jumpRelation* and *jumpSuccessor* are generated using probabilistic models of low level control processes and their physical effects. The projector probabilistically samples from the probability distributions defined for the initial state, exogenous events, probabilistic effects, and probabilistic control mode transitions.

Advantages of applying probabilistic sampling-based projection to the prediction of the effects of CRPs are that they work independently of the branching factor of the modes of the hybrid automaton and that it only constructs a small part of the complete PHAM.

So what kinds of prediction-based inferences can be drawn from samples of projected execution scenarios? The inference that we found most valuable for online revisions of robot plans is: do projected execution scenarios drawn from this distribution satisfy a given property p with a probability greater than θ? A robot action planner can use this type of inference to decide whether or not it should revise a plan to eliminate a particular kind of flaw: it should revise the plan if it believes that the flaws likelihood exceeds some threshold and ignore them otherwise. Of course, such inferences can be drawn based on samples only with a certain risk of being wrong. Suppose we want the planner to classify any flaw with probability greater than θ as to be eliminated and to ignore any flaw less likely than τ. We assume that flaws with probability between τ and θ have no large impact on the robot's performance. How many execution scenarios should the plan revision module to classify flaws correctly with a probability greater than 95%? Fig. 5 shows the number of necessary projections to achieve $\beta = 95\%$ accuracy. For a detailed discussion see (Beetz, Bennewitz, & Grosskreutz 1999).

	θ					
	1%	10%	20%	40%	60%	80%
$\tau = .1\%$	1331	100	44	17	8	3
$\tau = 1\%$	$\perp$	121	49	17	8	3
$\tau = 5\%$	$\perp$	392	78	22	9	3

Figure 5: Number of scenarios to get 95% accuracy.

The probabilistic sampling-based projection mechanism will become extremely useful for improving robot plans during their execution once the execution scenarios can be sampled fast enough. At the moment a projection takes a couple of seconds. The overhead is mainly caused by recording the interpretation of RPL plans in a manner that is far too detailed for our purposes. Through a simplification of the models we expect an immediate speed up of up to an order of magnitude. It seems that with a projection frequency of about 100 Hz one could start tackling a number of realistic problems that occur at execution time continually.

Evaluation

Generality. PHAMs are capable of predicting the behavior generated by flexible plans written in plan execution languages such as RAP (Firby 1987) and PRS (Georgeff & Ingrand 1989). To do so, we code the control structures provided by these languages as RPL macros. To the best of our knowledge PHAMs are the first realistic symbolic models of the sequencing layer of 3T architectures, the most commonly used software architectures for controlling intelligent autonomous robots (Bonasso *et al.* 1997). The use of PHAMs would enable 3T planning systems to make more realistic predictions of the robot behavior that will be generated from their abstract plans. PHAMs are also capable of modeling different arbitration schemes and superpositions of the effects of concurrent control processes (see argument in section "Multiple Interacting Control Processes").

Experimental Results. PHAMs have been in practical use for planning delivery tours of an autonomous robot office courier during their execution. The prediction-based scheduler has run for several hours on the real robot and for many hours on the robot simulator. Beetz & Grosskreutz (1998) experimentally investigate the accuracy of predictions and the time resources needed for making predictions in a predecessor of the PHAM projector. We found that our causal models were capable of predicting the qualitative behavior, that is the sequence of locations that the robot will visit and the actions that are performed, very well. The prediction of the durations of delivery tours, however, is very difficult because of the navigation behavior of state-of-the-art robot control systems exhibiting very high variances. Beetz & Belker (2000) discuss the problems of the behavior variance in the context of learning better navigation plans.

Beetz, Bennewitz, & Grosskreutz (1999) have performed a set of experiments with online prediction-based revisions for the delivery tours of an autonomous robot office courier, that uses PHAMs. As part of our experimental evaluation we have made up five scenarios in which the computation of good schedules required prediction-based online replanning. In those scenarios the controller using PPSD outperformed another one that used a clever situation-based rescheduling method by about 8% (see (Beetz, Bennewitz, & Grosskreutz 1999) for details). The scenarios required the use of realistic models of CRPs such as those provided by PHAMs. Fig 6(left) shows a situation that could not be handled by the situation-based scheduler but prediction-based online rescheduling avoided the problem (Fig 6(right)).

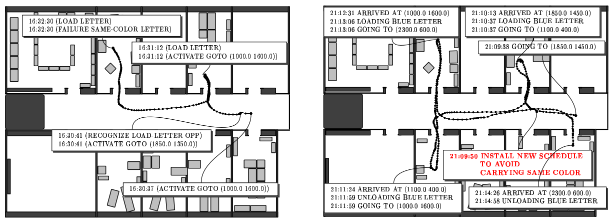
16:32:30 (LOAD LETTER)
16:32:30 (FAILURE SAME-COLOR LETTER)
16:31:12 (LOAD LETTER)
16:31:12 (ACTIVATE GOTO (1000.0 1600.0))
16:30:41 (RECOGNIZE LOAD-LETTER OPP)
16:30:41 (ACTIVATE GOTO (1850.0 1350.0))
16:30:37 (ACTIVATE GOTO (1000.0 1600.0))
21:12:31 ARRIVED AT (1000.0 1600.0)
21:13:06 LOADING BLUE LETTER
21:13:06 GOING TO (2300.0 600.0)
21:10:13 ARRIVED AT (1850.0 1450.0)
21:10:37 LOADING BLUE LETTER
21:10:37 GOING TO (1100.0 400.0)
21:09:38 GOING TO (1850.0 1450.0)
21:09:50 INSTALL NEW SCHEDULE TO AVOID CARRYING SAME COLOR
21:11:24 ARRIVED AT (1100.0 400.0)
21:11:59 UNLOADING BLUE LETTER
21:11:59 GOING TO (1000.0 1600.0)
21:14:26 ARRIVED AT (2300.0 600.0)
21:14:58 UNLOADING BLUE LETTER

Figure 6: Replanned delivery tour.

Related Work

PHAMs represent external events, probabilistic action models, action models with rich temporal structure, concurrent interacting actions, and sensing actions in the domain of autonomous mobile robot control. There are many research efforts that formalize and analyze extended action representations and develop prediction and planning techniques for them. We know, however, only of approaches that address subsets of the aspects listed above.

Various sorts of uncertainty have been integrated into the action representations of planning systems, which however assume that actions have a simple temporal structure (e.g. (Draper, Hanks, & Weld 1994)). Blythe (1995) gives an informative comparison of planning approaches for dynamic and uncertain domains. Hanks et al. (1995) presents a framework for representing probabilistic information, and exogenous and endogenous events for medical prediction problems. They do not deal with context-specific causal models. In recent years (PO)MDP planning has has become a very active research field (Boutilier, Dean, & Hanks 1998). (PO)MDP planners aim at precomputing the best decisions under uncertainty for all situations that an agent might face. So far, this approach has been feasible only for small or at least well-structured state spaces and has difficulties in handling concurrent and continuous actions.

Planning with action models that have rich temporal structure has also been investigated intensively. IxTeT (Ghallab & Laruelle 1994) is a planning system that has been applied to robot control and reasons about the temporal structure of plans to identify interferences between plan steps and resource conflicts. Other temporal planners include the TIME-LOGIC (Allen *et al.* 1990) and FORBIN (Dean, Firby, & Miller 1988). Allen & Ferguson (1994) gives an excellent and detailed a discussion of important issues in the representation of temporally complex and concurrent of actions and events without considering probabilistic information. Interferences between the effects of concurrent actions have to be axiomatized explicitly. Planners that predict qualitative state transitions caused by continuous events include EXCALIBUR (Drabble 1993). WEAVER (Blythe 1994) is an example of a planning system that considers exogenous events.

Conclusion

The successful application of AI planning to autonomous mobile robot control requires the planning systems to have more realistic models of the operation of modern robot control systems and the physical effects caused by their execution. In this paper we have presented a ***probabilistic hybrid action models (PHAMs)***, which are capable of representing the temporal structure of continuous feedback control processes, their non-deterministic effects, several modes of their interferences, and exogenous events. We have shown that PHAMs allow for predictions that are, with high probability, qualitatively correct. We have also shown that powerful prediction-based inferences such as deciding whether a plan is likely to cause a flaw with a probability exceeding a given threshold can be drawn fast and with bounded risk. Finally, we have demonstrated using experiments carried out on a real robot and a robot simulator that robots that decide based on predictions generated from PHAMs can avoid execution failures that robots without foresight cannot.

We believe that equipping autonomous robot controllers with concurrent reactive plans and prediction-based online plan revision based on PHAMs is a promising way to improve the performance of autonomous service robots through AI planning both signifi cantly and substantially.

Acknowledgements We would like to thank an anonymous reviewer who pointed us to the work on hybrid systems.

References

Allen, J., and Ferguson, G. 1994. Actions and events in interval temporal logic. Technical Report 521, University of Rochester, Computer Science Department.

Allen, J.; Kautz, H.; Pelavin, R.; and Tenenberg, J., eds. 1990. *Reasoning about Plans*. San Mateo, Cal: Morgan Kaufmann.

Alur, R.; Henzinger, T.; and Ho, P. 1996. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering* 22(3):181–201.

Alur, R.; Henzinger, T.; and Wong-Toi, H. 1997. Symbolic analysis of hybrid systems. In *Proceedings of the 37th IEEE Conference on Decision and Control*.

Arbib, M. 1998. *Behavior based Robotics*. MIT Press.

Beetz, M., and Belker, T. 2000. Environment and task adaptation for robotic agents. submitted for publication.

Beetz, M., and Grosskreutz, H. 1998. Causal models of mobile service robot behavior. In *Fourth International Conference on AI Planning Systems*, 163.

Beetz, M.; Bennewitz, M.; and Grosskreutz, H. 1999. Probabilistic, prediction-based schedule debugging for autonomous robot office couriers. In *Proceedings of the 23rd German Conference on Artificial Intelligence (KI 99)*. Springer Verlag.

Beetz, M.; Burgard, W.; Fox, D.; and Cremers, A. 1998. Integrating active localization into high-level control systems. *Robotics and Autonomous Systems* 23:205–220.

Beetz, M. 1999. Structured reactive controllers —a computational model of everyday activity. In *Proceedings of the Third International Conference on Autonomous Agents*.

Blythe, J. 1994. Planning with external events. In de Mantaras, R., and Poole, D., eds., *Procs. of the 10th Conf. on Uncertainty in Artificial Intelligence*, 94–101. Morgan Kaufmann Publishers.

Blythe, J. 1995. AI planning in dynamic, uncertain domains. In *Extending Theories of Action: Formal Theory and Practical Applications: Papers from the 1995 AAAI Spring Symposium*, 28–32. AAAI Press, Menlo Park, California.

Bonasso, P.; Firby, J.; Gat, E.; Kortenkamp, D.; Miller, D.; and Slack, M. 1997. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence* 9(1).

Boutilier, C.; Dean, T.; and Hanks, S. 1998. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of AI research*.

Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* 14–23.

Dean, T.; Firby, J.; and Miller, D. 1988. Hierarchical planning involving deadlines, travel time and resources. *Computational Intelligence* 4(4):381–398.

Drabble, B. 1993. Excalibur: a program for planning and reasoning with processes. *AI Journal* 62:1–40.

Draper, D.; Hanks, S.; and Weld, D. 1994. Probabilistic planning with information gathering and contingent execution. In Hammond, K., ed., *Proc. 2nd. Int. Conf. on AI Planning Systems*. Morgan Kaufmann.

Firby, J. 1987. An investigation into reactive planning in complex domains. In *Proc. of AAAI-87*, 202–206.

Fox, D.; Burgard, W.; Dellaert, F.; and Thrun, S. 1999. Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence*.

Georgeff, M., and Ingrand, F. 1989. Decision making in an embedded reasing system. In *Proc. of the 11 th IJCAI*, 972–978.

Ghallab, M., and Laruelle, H. 1994. Representation and control in ixtet, a temporal planner. In Hammond, K., ed., *Second International Conference on AI Planning Systems*, 61–67.

Hanks, S.; Madigan, D.; and Gavrin, J. 1995. Probabilistic temporal reasoning with endogenous change. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*.

Konolige, K.; Myers, K.; Ruspini, E.; and Saffiotti, A. 1997. The saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence* 9(2).

McDermott, D. 1991. A reactive plan language. Research Report YALEU/DCS/RR-864, Yale University.

McDermott, D. 1992a. Robot planning. *AI Magazine* 13:55–79.

McDermott, D. 1992b. Transformational planning of reactive behavior. Research Report YALEU/DCS/RR-941, Yale University.

McDermott, D. 1994. An algorithm for probabilistic, totally-ordered temporal projection. Research Report YALEU/DCS/RR-1014, Yale University.

S. Thrun, et al. 1998. *Map Learning and High-Speed Navigation in RHINO*. in: AI-based Mobile Robots. MIT Press.

Thrun, S. 1999. Monte carlo pomdps. In *Procs. of Conference on Neural Information Processing Systems (NIPS)*. to appear.