Causal Models of Mobile Service Robot Behavior

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

Abstract

Temporal projection, the process of predicting what will happen when a robot executes its plan, is essential for autonomous service robots to successfully plan their missions. This paper describes a causal model of the behavior exhibited by the mobile robot RHINO when running concurrent reactive plans for performing office delivery jobs. The model represents aspects of robot behavior that cannot be represented by most action models used in AI planning: it represents the temporal structure of continuous control processes, several modes of their interferences, and various kinds of uncertainty. This enhanced expressiveness enables XFRM (McD92; BM94), a robot planning system, to predict, and therefore forestall, various kinds of behavior flaws including missed deadlines whilst exploiting incidental opportunities. The proposed causal model is experimentally validated using the robot and its simulator.

Introduction

Temporal projection, the process of predicting what will happen when a robot executes its plan, is essential for autonomous service robots to successfully plan their missions. For the projection of their plans robots must have causal models that represent the effects of their actions. Most AI planning systems use fairly simple models of their actions and restrict themselves to work on plans that are partially ordered sets of actions.

Unfortunately, in autonomous robot control we often cannot consider plans as partially ordered sets of atomic actions without thwarting many opportunities for improving the robots' behavior through planning. The actions of robots have extents in both space and time and the information available for planning is deficient. The actions' extent and dependence on time require planning systems to predict when actions are executed, how long they take, and how they overlap with concurrent actions. The restrictedness, unreliability, and inaccuracy of sensors and effectors requires that planning systems compute flexible plans and are able to reason through contingencies whose likelihoods are a priori unknown and varying. These flexible plans violate assumptions underlying most action models used for planning — in particular the assumptions about the irrelevance of the temporal structure of actions and the exclusion of interferences between concurrent actions.

Forestalling a wide range of behavior flaws typical for service robots requires the planner to make use of models that represent *continuous processes*, *exogenous events*, *interferences between concurrent behavior*, *incomplete information*, and *passive sensors*.

This paper describes a model for predicting the behavior generated by a robot controller for office delivery jobs. The controller is designed for robust and efficient execution of delivery plans on the autonomous mobile robot RHINO (see Fig. 1), an RWI B21 robot. The causal model represents *various kinds of uncertainty*, the *temporal structure* and *concurrent execution* of continuous control processes, and thereby enables the transformational planning system XFRM (McD92; BM97) to forestall a variety of behavior flaws (BM94).

Fig. 1: RHINO

To predict possible execution scenarios quickly and to keep the representation of the scenarios concise the causal models of continuous control processes are (partly) generated and revised during the temporal projection of the processes. This allows the projection algorithm to use causal models tailored for the surrounding plan and its state of execution. These context-specific causal models predict only those state transitions caused by the continuous processes that might affect the course of plan execution.

We use RHINO's navigation behavior as our principle example for sensor-driven, concurrent control processes. Navigation behavior has several advantages over other kinds of control processes: first, it's difficult to imagine having a sophisticated planner for mobile

robot applications without having an adequate model of navigation behavior; second, navigation is one of the best understood capabilities of mobile robots; and third, navigation is a suitable means for experimentally validating the causal models of concurrent continuous processes.

In this paper we proceed as follows. The next section describes the plans of the office delivery robot that are to be projected. The rule language for specifying causal models of actions is sketched in the subsequent section. The section on modeling RHINO's behavior applies the rule language to model the base navigation plans and the events that occur in the environment. Finally, we give some experimental results on the accurateness of the symbolic predictions of robot behavior.

Plans of an Office Delivery Robot

We use *structured reactive plans* **(SRPs)** to specify how the delivery robot is to respond to sensory input in order to accomplish its jobs. They are written in RPL *(Reactive Plan Language)* (McD91), a LISP-like robot control language with conditionals, loops, local variables, processes, and subroutines. RPL provides several high-level concepts (interrupts, monitors) to synchronize parallel actions, make plans reactive, and so on.

Structured Reactive Delivery Plans

To illustrate the advantages of high-level plans that specify concurrent reactive behavior, we sketch a plan that specifies how RHINO (see Fig. 1) is to deliver mail to the rooms A-120, A-113, A-121, and A-110. Initially, the planner asked the robot to perform the deliveries in the order A-120, A-113, A-121, and A-110. However, because the room A-120 is closed the corresponding delivery cannot be completed. Therefore, the planning system revises the overall plan such that the robot is to accomplish the delivery for A-120 as an opportunity. In other words, the robot will interrupt its current delivery to deliver the mail to A-120 (see Fig. 2) if the delivery can be completed.

```
WITH-POLICY WHENEVER PASSING A DOOR
                ESTIMATE DOOR ANGLE
WITH-POLICY SEQ WAIT-FOR OPEN?(A-120)
                DELIVER MAIL TO Dieter
  1. GO-TO(A-113)
  2. GO-TO(A-121) BEFORE 10:30
  3. GO-TO(A-110)
```

Figure 2: Office delivery plan

Constraints such as "whenever the robot passes a door it estimates the opening angle of the door using its laser range finders" and opportunities such as "complete the delivery to room A-120 as soon as you learn the office is open," which are necessary for carrying out the jobs opportunistically, are specified using the RPL construct WITH-POLICY. WITH-POLICY *P B* means "execute the primary activity *B* such that the execution satisfies the policy *P*." Policies are concurrent processes that run while the primary activity is active and interrupt the primary if necessary.

Events that require RHINO to perform actions such as "passing a door" are handled through fluents, program variables that signal changes of their values and thereby enable control threads to react to asynchronous events. For instance, the RPL statement WHENEVER *F B* is an endless loop that executes *B* whenever the fluent *F* gets the value "true." WAIT-FOR *F*, another control abstraction, blocks a thread of control until the fluent *F* becomes true.

When a delivery gets interrupted because the robot has detected that the door to A-120 is open, that opportunity has a side effect: it moves the robot into the office A-120. The interrupted delivery plan has therefore to be replanned before it can be continued.

This in mind, we implement the navigation routine as a loop that generates and executes base navigation plans until the robot has arrived at its destination. Interrupts are handled by terminating the current iteration of the loop and starting the next iteration in which a new navigation plan starting from RHINO's new position is generated (BBFC98).

Base Navigation Plans

Base navigation plans are also specified as concurrent reactive RPL plans. Figure 3 pictures such a navigation plan, that is automatically generated for a given destination by the SRP's navigation planner. The plan consists of two components. The first specifies a sequence of target points (the locations indexed by the numbers 1 to 5 in Figure 3) to be reached by the robot. The navigation between the target points is accomplished by a standard path planner (TBB+98).

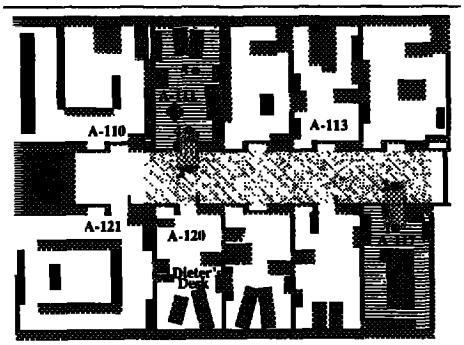

Fig. 3: Topological navigation plan

The second component specifies in detail when and how the robot is to adapt its travel modes as it follows the navigation path (FBT97). In many indoor environments it is advantageous to adapt the driving strategy

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 crosstalk between the sonars), and to drive quickly in the hallways. This part of the plan is depicted by regions with different textures for the different travel modes "office," "hallway," and "doorway." Whenever the robot crosses the boundaries between regions the appropriate travel mode is set.

Issues and Requirements

Suppose RHINO has received two pieces of evidence (1) "Dieter will be back at 10:25" and (2) "Tables are moved from room A-110 to A-117." In this situation the task of the temporal projection module is the following: given the delivery plan in Fig. 2 predict whether, in conjunction with the new evidence, the plan is still appropriate. In particular, the projection module should predict that the robot is likely to miss the deadline for the delivery to room A-121 (because it will take the opportunity to complete the delivery to A-120) and that the robot is likely to bump into a table when entering room A-110 (because it switches off sonar sensors when entering doorways).

Making these predictions requires that the robot planning system uses models of

- ***continuous processes*** for the base navigation plans that predict whether and when endogenous events, such as adaptations of the driving strategy and passing a door will occur;
- ***exogenous events*** like the opening of the door to room A-120 around 10:25 or the refurnishing of A-110;
- ***interferences between concurrent behavior*** like the interuption of a delivery plan to make use of the open door to room A-120 to deliver the mail;
- ***incomplete information*** including the duration of events, non-deterministic effects, and sensor models;
- ***passive sensors and obstacle avoidance*** such that the models predict readings from passive sensors only when they determine the course of action.

Probabilistic, Totally-Ordered Temporal Projection

We use the representation of events and their effects and the temporal projection algorithm developed by McDermott and described in (McD92; McD94). The plan projection process takes the current world model, a structured reactive plan, rules for generating exogenous events, and rules for probabilistically guessing missing pieces of the world model together with probabilistic causal models of the basic control routines and generates *execution scenarios* of the plan.

An execution scenario represents how the execution of a robot controller might go, that is, how the environment changes as the structured reactive controller gets executed. A *timeline* is a linear sequence of events and their results. Timelines represent the effects of plan execution in terms of *time instants*, *occasions*, and *events*. *Time instants* are points in time at which the world changes due to an action of the robot or an exogenous event. An *occasion* is a stretch of time over which a world state P holds and is specified by a proposition, which describes P, and the time interval for which proposition is true.

The plan projector works exactly like the plan interpreter, except that, whenever the executed plan interacts with the real world, correspondingly the projected plan interacts with the timeline. The places where this happens are the low-level plans, such as the base navigation plans: the projector guesses the results of executing these plans and asserts their effects in the form of propositions on the timeline.

The representational means for specifying causal models used in the remainder of the paper are summarized below (see (McD94)):

- **Projection rules** have the form (PROJECT *C A Es O*) and the following meaning: if routine C starts and A holds, then the events Es will occur and C will return O upon completion. The outcome o is either of the form (FINISH —*VALUES*—) if the routine is completed successfully or (FAIL —*DESCRIP*—) otherwise. This statement specifies the signal that is sent by the behavior module when the behavior is completed.
- **Effect rules** (E->P *A E r B*) specify that whenever event E occurs and A holds, then with probability r create and clip states as specified. The effects of the E->P rules have the form A, causing the occasion A to hold, (CLIP *A*), causing A to cease to hold, and (PERSIST *t A*), causing A to hold for t time units.
- **Exogenous event rules** are specified as P->E rules. (P->E *A d E*) generate over any interval in which A is true, generates random "Poisson-distributed" events with an average spacing of d time units.

McDermott (McD94) gives a formal semantics for the rule language introduced above, shows that a consistent set of rules has a unique model, and proves the algorithms for building and retrieving from the timeline to be correct. (BM97) show that the average performance of robot plans can be improved based on a small number of randomly projected scenarios.

Modeling RHINO's Behavior

After the description of the office delivery plans and the introduction of *projection*, E->P, and P->E rules as a means for representing action and change, we will now

apply these representations to describe the behavior caused by the execution of delivery plans, such as the one listed in Fig. 2.

The delivery plans control a dynamic system (cf. (DW91)): the robot RHINO in its environment (see Fig. 4). The state variables of the dynamic system that are relevant for the delivery plans are the variables X, Y representing RHINO's position, and the variables DOOR-ANGLE, representing the opening angle of the doors. The robot controller uses fluents to store the robot's measurements of these state variables (RHINO-X*, RHINO-Y*, DOOR-A120, etc.). The fluents are steadily updated through "sensing processes" such as the position tracking process (BBFC98) and a model-based procedure for estimating the opening angles of doors. The actions generated by the controller are all changes of the velocity and the heading of the robot.

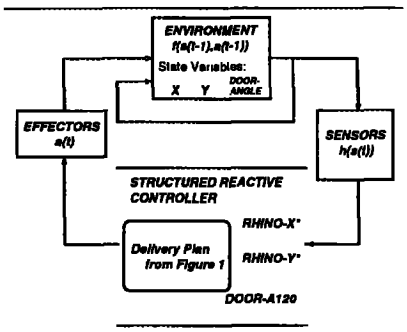

Fig. 4: Office delivery as a dynamic system

Unfortunately, dynamic system models of plan execution that describe all state transitions are too fine-grained to be useful for AI planning. On the other hand, we cannot simply make the model more coarse-grained: as we have pointed out earlier, the delivery plans specify how the robot is to respond to changes in the dynamic system, and therefore any states the dynamic system traverses might potentially change the course of action drastically.

Resolving the conflicting requirements of (1) predicting all relevant events and states and (2) both generating projections quickly and representing them concisely constitutes a difficult problem in applying AI planning to autonomous robot control. Our approach to satisfying these requirements is to construct part of the causal models of continuous processes in a context-dependent way based on an analysis of the computational state of plan execution at the start of the process.

We model the behavior generated by base navigation plans as a complex event that causes only qualitative changes of behavior (e.g. adaptations of the travel mode) and those changes of the robot's position that are relevant for the execution of the overall plan (e.g. entering the hallway or passing a door). To realize this model, the continuous processes are projected as follows: (1) analyze the computational state and extract the conditions caused by the process and waited for by concurrent branches of the delivery plan; (2) estimate when the conditions will become true and what the values of the system variables at that time will be; and (3) generate a context-dependent causal model that is used for projecting the plan.

Projecting Continuous Behavior

For efficiency reasons the process of projecting a continuous process p is divided into two phases. The first phase estimates a schedule for endogenous events caused by p while considering possible effects of p on other processes but not the effects of the other processes on p. This schedule is transformed into a context-specific causal model tailored for the plan which is to be projected. The second phase projects the plan p using the model of endogenous events constructed in the first phase. This phase takes into account the interferences with concurrent events and revises the causal model if situations arise in which the assumptions of the precomputed schedule are violated.

The projection module uses a model of the dynamic system that specifies for each continuous control process the state variables it changes and for each state variable the fluents that measure that state variable. For example, consider the base navigation plans that steadily change the robot's position (that is the variables X and Y). The estimated position of the robot is stored in the fluents RHINO-X* and RHINO-Y*:

CHANGES(BASE-NAVIGATION-PLAN, X)
CHANGES(BASE-NAVIGATION-PLAN, Y)
MEASURES(RHINO-X*, X)
MEASURES(RHINO-Y*, Y)

Extracting relevant conditions. When the projector starts projecting a base navigation plan it computes the set of pending conditions that depend on RHINO-X* and RHINO-Y*, which are the fluents that measure the state variables of the dynamic system and are changed by the base navigation plan. These conditions are implemented as fluent networks.

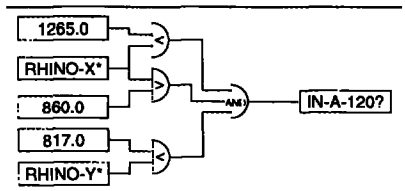

Fig. 5: Fluent network for being in room A-120

Fluent networks are digital circuits where the components of the circuit are fluents. Fig. 5 shows a fluent network where the output fluent is true, if and only if RHINO is in room A-120. The inputs of the circuit are the fluents RHINO-X* and RHINO-Y* and the circuit is updated whenever RHINO-X* and RHINO-Y* change.

Structured reactive plans are set up such that the fluent networks that compute conditions for which the plan is waiting can be automatically determined using (PROLOG-like) relational queries:

```
(SETOF ?FL-NET (AND (FLUENT ?FL)     (STATUS ?FL PENDING)
                    (CHANGES BASE-NAV-PLAN ?STATE-VAR)
                    (MEASURES ?STATE-VAR-FL ?STATE-VAR)
                    (DEPENDS-ON ?FL ?STATE-VAR-FL)
                    (FLUENT-NETWORK ?FL ?FL-NET))
       ?PENDING-FL-NETS)
```

This query determines ?PENDING-FL-NETS, the set of fluent networks ?FL-NET such that ?FL-NET is a network with output fluent ?FL. ?FL causes a plan thread to pend and depends itself on a fluent measuring a state variable ?STATE-VAR changed by the base navigation plan.

Endogenous event schedules. For each class of continuous processes we have to provide an *endogenous event scheduler* that takes the initial conditions of the process, the parameterization of the process, and the fluent networks that might be triggered by the process and computes the endogenous event schedule. The endogenous event scheduler for the base navigation plans is described in the next section. Given the kind of process (e.g., base navigation plan), the process parameters (e.g., the destination of the robot), and the pending fluent networks, the scheduler returns the sequence of predicted endogenous events, which are triples of the form $(\Delta t, (sv_1, ..., sv_n), \{ev_1, ..., ev_m\})$. Δt is the delay between the ith and the $i+1$th event in the schedule, $(sv_1, ..., sv_n)$ the values of the state variables, and $\{ev_1, ..., ev_m\}$ the events that are to take place.

If a state for which the plan is waiting, becomes true at a time instance t, then at t a PASSIVE-SENSOR-UPDATE event is triggered. PASSIVE-SENSOR-UPDATE is an event model that takes a set of fluents as its parameters, retrieves the values of the state variables measured by these fluents, applies the sensor model to these values, and then sets the fluents accordingly.

A causal model of base navigation plans. Projecting the initiation of the execution of a navigation plan causes two events: the start event and a hypothetical completion event after infinite number of time units. This is shown in the following projection rule.

```
(PROJECT (BASE-NAV-PLAN ?DEST-DESCR ?ID ?FLUENT)
  (TRUE)
  (0   (BEGIN (BASE-NAV-PLAN ?DEST-DESCR ?ID ?FLUENT))
   ∞   (END (BASE-NAV-PLAN ?DEST-DESCR ?ID ?FLUENT)))
  (FINISH))
```

The effect rule of the start event of the base navigation plan computes the endogenous event schedule and asserts the schedule the occurrence of the next endogenous navigation event as an occasion to the timeline.

```
(E->P (ENDOGENOUS-EVENT-SCHEDULE
        BASE-NAV-PLAN ?DEST-DESCR ?SCHEDULE)
      (BEGIN (BASE-NAV-PLAN ?DEST-DESCR ?ID ?FLUENT))
      1.0 (AND (PREDICTED-EVENTS ?ID ?SCHEDULE)
               (RUNNING (RHINO-GOTO ?DESCR ?ID))
               (NEXT-NAV-EVENT ?ID)))
```

The occasion (NEXT-NAV-EVENT ?ID) triggers the next endogenous event (BEGIN (FOLLOW-PATH ?HERE (?X,?Y)) ?DT ?ID)). The remaining two conditions determine the parameters of the FOLLOW-PATH event: the next scheduled event and RHINO's position.

```
(P->E (AND (NEXT-NAV-EVENT ?ID)
           (PREDICTED-EVENTS ?ID
                ((?DT (?X,?Y) ?EVS) !?REMAINING-EVS))
           (RHINO-LOC ?HERE))
      0.0 (BEGIN (FOLLOW-PATH ?HERE (?X,?Y) ?DT ?ID)))
```

The effect rule of the (BEGIN (FOLLOW-PATH ...)) event specifies among other things that the next endogenous event will occur after ?DT time units (PERSIST ?DT (SLEEPING ?ID)).

```
(E->P (RHINO-LOC ?COORDS)
      (BEGIN (FOLLOW-PATH ?FROM ?TO ?DT ?ID))
      1.0 (AND (RUNNING (FOLLOW-PATH ?FROM ?TO ?DT ?ID))
               (CLIP (RHINO-LOC ?COORDS))
               (CLIP (NEXT-NAV-EVENT ?ID))
               (PERSIST ?DT (SLEEPING ?ID))))
```

If a running follow path event has finished sleeping the (END (FOLLOW-PATH ...)) event occurs.

```
(P->E (AND (NOT (SLEEPING ?ID))
           (RUNNING (FOLLOW-PATH ?FROM ?TO ?TIME ?ID)))
      0.0 (END (FOLLOW-PATH ?FROM ?TO ?TIME ?ID)))
```

Our model of base navigation plan presented so far suffices as long as nothing important happens while carrying out the plan. However, suppose that an exogenous event that causes an object to slip out of the robot's hand is projected at time instant t while the robot is in motion. To predict the new location of the object the projector predicts the location l of the robot at the time t and asserts it in the timeline.

Qualitative changes in the behavior of the robot caused by adaptations of the travel mode are described through E->P-rules. The following E->P-rule describes the effects of the event (NAV-EVENT (SET-TRAVEL-MODE ?N)):

```
(E->P (TRAVEL-MODE ?M)
      (NAV-EVENT (SET-TRAVEL-MODE DOORWAY))
      1.0 (AND (CLIP (TRAVEL-MODE ?M))
               (CLIP (OBSTACLE-AVOIDANCE-WITH SONAR))
               (TRAVEL-MODE DOORWAY)))
```

The rule specifies that if at a time instant at which an event (NAV-EVENT (SET-TRAVEL-MODE ?N)) occurs the state (TRAVEL-MODE ?M) holds for some ?M, then the states (TRAVEL-MODE ?M) and (OBSTACLE-AVOIDANCE-WITH SONAR) will (with a probability of 1.0) not persist after the event has occurred, i.e., they are clipped by the event. The event also causes the state (TRAVEL-MODE DOORWAY) to hold until it is adapted the next time.

Endogenous Event Scheduler

We have just shown how events are projected from a given endogenous event schedule, but we have not shown how the schedule is constructed. Thus, this section describes the implementation of the endogenous event scheduler for base navigation plans. The scheduler predicts the effects of the base navigation plan

on the state variables X and Y. The endogenous event scheduler assumes the robot follows a straight path between the locations 1 to 5. As we have pointed out earlier, there are two kinds of events that need to be predicted: the ones causing *qualitative physical change* and the ones causing the *trigger conditions* that the plan is waiting for.

The qualitative events caused by the base navigation plan pictured in Fig. 2 are the ones that occur when the robot arrives at the locations 1, 2, 3, 4, and 5 in which the robot either changes its travel mode or arrives at its destination. For each of these time instants the occurrence of a SET-TRAVEL-MODE-event is predicted.

The scheduler for trigger events works in two phases: (1) it transforms the fluent network into a condition that it is able to predict and (2) it applies an algorithm for computing when these events occur. The conditions that are caused by the base navigation plan can be represented as regions in the environment such that the condition is true if and only if the robot is within this region. The elementary conditions are numeric constraints on RHINO's position or the distance of RHINO to a given target point. The scheduler assumes that RHINO-X* and RHINO-Y* are the only fluents in these networks that change their value during the execution of the plan. More complex networks can be constructed as conjunctions and disjunctions of the elementary conditions.

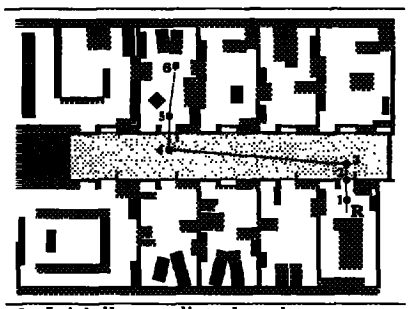

Fig. 6: Initially predicted endogenous events

The endogenous event scheduler approximates the regions by rectangular areas in order to apply faster computation methods. Consequently, circles, that is regions specified by a maximal distance to a given point, are approximated as the surrounding squares. Thus, the step performed by the endogenous event scheduler is to transform the given fluent network into a union or intersection of rectangles.

In the next step the endogenous event scheduler overlays the straightline path through the intermediate goal points of the topological navigation path (see Fig. 2) with the regions computed in the previous step. It then computes a schedule for the endogenous events by following the navigation path and collecting the intersections with the regions (see Figure 6). The result of the scheduling step is a sequence of triples of the form $(\Delta t_i, (X_i, Y_i), \{ev_1, ..., ev_n\})$.

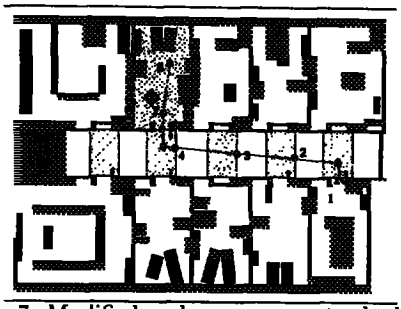

Fig. 7: Modified endogenous event schedule

Rescheduling endogenous events. One problem that our temporal projector has to deal with is that a WAIT-FOR step might be executed while a base navigation plan is projected. For example, when the robot enters the hallway, the policy that looks for the opening angles of doors when passing them is triggered. Therefore, the causal model that was computed by the endogenous event scheduler is no longer sufficient. It fails to predict the "passing a door" events.

These problems are handled by modifying the endogenous event schedule: whenever the robot starts waiting for a condition that is a function of the robot's position, it interrupts the projection of the base navigation plan, adapts the causal model of the base navigation plan, and continues with the projection. In the case of entering the hallway, a new endogenous event schedule is computed that contains endogenous events for passing doorways. This updated schedule of endogenous events is pictured in Fig. 7.

Projecting Exogenous Events

One type of exogenous event is an event for which we have additional information about their time of occurrence, such as the event that Dieter will be back from lunch around 12:25. These kinds of events are represented by an E->P rule together with a E->P rule. The E->P rule specifies that the (START) event causes the state (BEFORE-DIETERS-DOOR-OPENS) to hold and persist for ?TIME time units. The event (DIETERS-DOOR-OPENS) is triggered as soon as (BEFORE-THE-DOOR-OPENS) no longer holds.

```
(E->P (AND (ABOUT ?TIME 12:25)
           (DIFFERENCE ?TIME *NOW* ?WAIT-FOR))
      (START)
      1.0 (PERSIST ?WAIT-FOR (BEFORE-THE-DOOR-OPENS)))
(P->E (NOT (BEFORE-THE-DOOR-OPENS))
      0.0 (DIETERS-DOOR-IS-OPENED))
```

Passive Sensors and Obstacle Avoidance

Collision avoidance is not modeled except in situations in which the robot is told about objects that are moved

around. In this case the endogenous event scheduler adds a region corresponding to the object. If the region blocks the way to the destination — that is the robot cannot move around the region — then a POSSIBLE-BUMP-EVENT is generated. The effect rule for a possible bump event specifies that, if the robot has activated sensors that can detect the object, the base navigation plan fails with a failure description "path blocked." Otherwise a bump event is generated. For example, since sonar sensors are the only sensors placed at table height, the collision avoidance module can avoid a collision with a table only if the sonar sensors are active. Thus, to predict a bump, the projector has to determine how long the sonar sensors have been switched off before the possible bump event occurs.

Experimental Results

We have validated our causal model of base navigation plans and their role in office delivery plans with respect to computational resources, qualitative prediction results, and quantitative results in a series of experiments. A typical experiment is described below.

Projecting the plan listed in Fig. 2 takes on average 5 seconds (without optimizations). The projection generates a timeline that is about 330 events long and about 1200 stack frames. Many of these events are generated through rescheduling the endogenous events (21 times). If the robot were to be solely concerned with navigation, the projection process would be too slow. An office delivery robot, however, spends much of its time interacting with people, picking up and delivering mail and therefore the planner has more than adequate time for debugging its delivery plan.

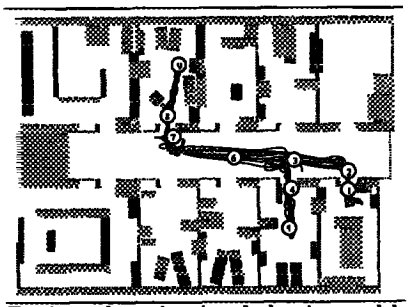

Fig. 8: Projected navigation behavior and behavior generated in the simulator

Figure 8 shows the predicted endogenous events (denoted by the numbered circles) and the behavior generated by the navigation plan in the robot simulator. The qualitative predictions of behavior relevant for plan debugging are perfect. In Fig. 8 the projector predicts correctly that the robot will exploit the opportunity to go to location 5 while going from location 1 to 9. For the two scenarios that we have discussed earlier, the projector perfectly predicts the relevant qualitative aspects of the robot's behavior: whether or not the robot bumps into the table and whether or not it misses the deadline.

We have also validated our causal models by comparing the symbolically projected behavior generated by the delivery plans, with the behavior they generate in the robot simulator and crosschecked the behavior of the simulated with the real robot in a shorter series of experiments. The predictions for the time needed to perform a single navigation task (taking about 1 to 2 minutes) are often off by 10 to 20%. These inaccuracies are mainly due to the high variances in navigation behavior. Because the grain size for time in everyday activity (deadlines, when people leave and come back, etc.) is in the range of minutes the accuracy of our predictions is more than sufficient.

Related Work

Due to space limitations we cannot give a detailed account of all aspects of related work. Related work comprises research on reasoning about action and change, probabilistic planning, numerical simulation, and qualitative reasoning.

(AF94) 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. (HMG95) presents a framework for representing probabilistic information, and exogenous and endogenous events for medical prediction problems. They consider smaller problems and do not have to generate context-specific causal models.

Planning algorithms, such as SNLP (MR91), have been extended in various ways to handle more expressive action models and different kinds of uncertainty (about the initial state and the occurrence and outcome of events) (e.g. (KHW95; DHW94)). These planning algorithms compute bounds for the probabilities of plan outcomes and are computationally very expensive. They also abstract away from the rich temporal structure of events by assuming discrete atomic actions and ignore various kinds of uncertainty. Partially observable Markov decision processes (POMDPs) focus on choosing optimal actions under uncertainty (KCK96). So far approximation algorithms for POMDPs are computationally too expensive to be applied to state spaces as large as ours (PR95).

Work in qualitative reasoning has researched issues in the quantization of continuous processes and focussed among other things on quantizations that are relevant to the kind of reasoning performed. (Hen73) points out the limitations of discrete event represen-

tations and introduces a very limited notion of continuous process as a representation of change. He does not consider the influence of multiple processes on state variables. (Hay85) represents events as *histories*, spatially bounded, but temporally extended, pieces in time space, and proposes that histories which do not intersect do not interact. In Forbus' Qualitative Process Theory (For84) a technique called limit analysis is applied to predict qualitative state transitions caused by continuous events. Also, work on simulation often addresses the adequacy of causal models for a given range of prediction queries, an issue that is neglected in most models used for AI planning.

Conclusion

To improve the perfomance of robot action planners we must equip them with better and more realistic models of the robots' behavior and the physics of the world. This paper has described a causal model of the behavior exhibited by the mobile robot RHINO when running concurrent reactive plans for performing office delivery jobs. The model represents the temporal structure of continuous control processes, several modes of their interferences, and various kinds of uncertainty. The model is used in FPPD (BM97), a plan revision technique that can. with high probability, forestall probable situation-specific execution failures based on randomly projecting a small number of execution scenarios.

We believe that the use of detailed models of autonomous robot behavior together with plan revision techniques such as FPPD will enable us to implement control systems for service robots that perform on average better than they possibly could without planning and that avoid categories of behavior flaws that cannot be avoided by most other planning systems (see (BM94)).

Our future work on developing causal models of robot behavior will focus on optimization, learning causal models, and modeling other interesting processes such as map building. searching for objects, vision routines, and manipulation actions.

References

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

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

M. Beetz and D. McDermott. Improving robot plans during their execution. In Kris Hammond, editor, *Second International Conference on AI Planning Systems*, pages 3–12, Morgan Kaufmann, 1994.

M. Beetz and D. McDermott. Fast probabilistic plan debugging. In *Recent Advances in AI Planning. Proceedings of the 1997 European Conference on Planning*, pages 77–90. Springer Publishers, 1997.

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

T. Dean and M. Wellmann. *Planning and Control*. Morgan Kaufmann Publishers, San Mateo, CA. 1991.

D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, March 1997.

K. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85 168, 1984.

P. Hayes. The second naive physics manifesto. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 1–36. Ablex, Norwood, NJ, 1985.

G. Hendrix. Modeling simultaneous actions and continuous processes. *Artificial Intelligence*, 4:145–180, 1973.

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

Leslie Pack Kaelbling, Anthony R. Cassandra, and James A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

N. Kushmerick, S. Hanks, and D. Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 76:239 286. 1995.

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

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

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

David McAllester and David Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, volume 2, pages 634–639, Anaheim, California, USA, July 1991. AAAI Press/MIT Press.

R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995.

S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998. to appear.