Validating Plans in the Context of Processes and Exogenous Events

Maria Fox and **Richard Howey** and **Derek Long**
Department of Computer and Information Sciences, University of Strathclyde, Glasgow, UK
firstname.lastname@cis.strath.ac.uk

Abstract

Complex planning domains push the boundaries of the expressive power of planning domain modelling languages. Recent extensions to the standard planning languages have included expressions for temporal, metric and resource structures. Other work has also considered how process models can be incorporated into domain models. In this paper we consider the problem of expressing and validating models containing *events* which are triggered as a consequence of the action of physical processes. We focus, primarily, on the validation of plans in the context of exogenous events, discussing the modelling, semantic and implementation issues that arise. Events impact not only on plans but on domain models as a whole and we also consider the problems that arise in considering the validation of event structures in domain models.

1 Introduction

In recent years planning research has made significant strides forward from classical STRIPS planning, facing problems involving explicit temporal structure, metric fluents, conditional and derived effects (Thiebaux & Cordier 2001) and even continuous change (McDermott 2003; Shin & Davis 2004). These extensions to the expressive power of planning domain descriptions have brought into reach a far more interesting class of problems. However, many challenges remain to be confronted. One example is a relaxation of the classical assumption that all change in a problem domain occurs as a direct effect of the execution of planned activities. In many domains, the indirect consequences of actions are equally important, triggered as a result of the processes initiated by the actions of the executive. These triggered *events* can represent opportunities to be exploited by a planner or threats to be avoided. In either case, a planner must have access to a representation of the events and the basis for determining whether and when they occur in order to assess their impact on a plan.

As an example, when planning the startup procedure for a process plant (Aylett *et al.* 2001), events include the reaction of chemicals released into the same vessel, the potential danger to the integrity of vessels caused by increasing pressure, the flow of fluids through connecting pipes caused by opening valves, explosive reactions caused by vessels exceeding safe threshold temperatures and so on. Some of the events are to be exploited and others to be avoided.

In this paper we consider the problem of modelling events in order to make them accessible to planning systems. We examine, in particular, how events can be given a semantic interpretation within the existing framework for domain representation (Fox & Long 2003) and show how this interpretation can be handled in the validation of plans that interact with domains that include potential events. We show that the cost of plan validation grows only linearly with the number of events that are triggered in the execution of a plan. This work contributes to the problem of accurate modelling of complex domains and to the validation and verification of domain models and plans. In this paper we emphasise the validation of plans, but we can use the output of the plan validation process to support the verification of the underlying planning domain model. This is an issue of increasing importance, as planning researchers aspire to apply their work to more complex problems relevant to real applications.

The semantics of events we describe is implemented in the automatic plan validation tool, VAL (Howey, Long, & Fox 2004). In this paper we discuss the practical difficulties in validation presented by the introduction of events and the solutions we have adopted. These solutions are not only relevant to the validation of plans, but can also provide the foundation of the mechanisms required by planners that are intended to manage events.

Although the introduction of processes and events into the PDDL2.1 framework offers significant expressive power, we do not propose that every process or event in the physical domain should be represented directly in the model: only processes or events with which the executive can usefully interact need be modelled, while all other processes can be abstracted just as they are currently.

2 Motivation

In many potential application domains for planning there are complications that cannot be captured in classical planning domain descriptions. Various extensions of classical planning have been proposed and explored, such as planning under uncertainty and planning with partial observability, with the intention of moving planning towards more realistic application domains. In the real-time control systems community there is a focus on controlling systems that are driven

by physical processes as well as controller actions (Henzinger 1996). Systems of this kind offer a major challenge to the planning community, both as a huge opportunity for the application of deliberative and planned control but also in stretching beyond the limits of current expressive capabilities of classical planning languages. Two extensions that are important for representing such systems are the expression of continuous processes and the expression of system responses to situations. A system response is a state change that is triggered not by an action on the part of the executive, but by mechanisms inherent to the physical system.

For example, if a ball is dropped onto a floor its velocity changes at the point of impact with the floor. The change is not brought about directly by the executive, but by a mechanism within the physical environment: bouncing. As can be seen in this example, the event is a consequence of an interaction between a process (the falling of the ball) and its environment (the position of the ball relative to the floor). In many real-time control problems there are situations similar to this. It is important to model such a situation in a planning domain because, if a planner is to attempt to control a system like this, it is necessary for the planner to be able to predict and plan around events that are triggered by the actions (or inactions) that are planned for the executive.

PDDL2.1 already has the expressive power to represent events that occur simply as a consequence of the passage of time (events such as sunrise and sunset) (Fox, Long, & Halsey 2004). Events that are triggered by activities initiated by the executive cannot be modelled without extending the language. In earlier work (Fox & Long 2003; Howey, Long, & Fox 2004) Fox, Long and Howey examined how some forms of continuous change can be modelled and some of the consequent problems for plan validation. Events represent another step in the progress towards modelling and planning with models of rich physical systems.

Further examples motivating the need to model event-behaviours in planning domains include:

- In the control of an orbiting observation satellite, the event of the heaters being triggered to protect sensitive parts of the satellite that are in shadow because of the orientation of the satellite, requiring planned activities to work within constrained energy supply levels.
- In a logistics-style domain, the event of a driver having reached maximum safe driving hours under European law and having to take a break, requiring planned activity to either work with the necessary delay in transit or else to have provided for the availability of a relief driver at an appropriate driver-exchange site.
- The event of a fermentation vessel reaching a temperature at which its contents undergo a chemical reaction.

In this paper we do not consider non-deterministic effects or unpredictable events. It might appear that this constraint implies that events could be modelled using conditional effects or delayed effects (Bacchus & Kabanza 2000a). In fact, this is not the case, because the interaction with processes means that events can occur at arbitrary times, not linked directly to the execution of any actions.

At this point we can offer a preliminary, though informal, definition of an event.

Definition 2.1 *An event is an instantaneous change in logical and/or numeric state, brought about by the interaction between a physical process and the environment.*

We will refine this definition in the following sections.

3 Semantic issues raised by events

If events were only ever the consequence of discrete changes caused by actions then it would be sufficient to model them using conditional effects of the actions that could cause them (although it might be tedious to do so). The more interesting case is when events can be caused by continuous change. In earlier work Fox, Long and Howey described how continuous change is modelled in PDDL2.1 (Fox & Long 2003; Howey & Long 2003; Howey, Long, & Fox 2004).

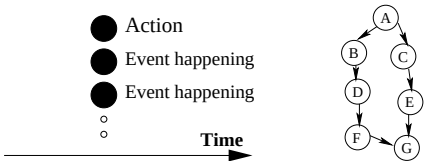
Events were first proposed as an extension for PDDL in (Fox & Long 2002) as part of an enriched language, PDDL+, along with a formal semantics in terms of hybrid automata (Henzinger 1996). This approach is attractive as it is a widely accepted model of mixed discrete-continuous activity. In models of hybrid automata, there is no distinction between the activities of an executive and the events triggered by the world. As a consequence, any events that trigger in a particular trajectory are recorded explicitly in the trajectory. In contrast, because a plan describes only the activities under control of the executive, and events are necessarily triggered by the environment and therefore not under the control of the executive, they are not explicitly recorded. Instead they must be inferred by examining the trajectory of the planned activities of the executive. This difference gives rise to certain semantic difficulties when constructing a mixed discrete-continuous model of a planning domain. In this paper we do not attempt to give the details of a formal semantics, but instead concentrate on discussing the problems that arise in defining such a semantics and our proposals for resolving them.

3.1 Modelling events

A simple proposal for modelling events is to use the format of an action, with pre and postconditions, but to label events to distinguish them from actions and to prevent them from being selected by a planner. This proposal is essentially consistent with the hybrid automaton model, treating events as semantically equivalent, in their effects, to actions.

The intuitive meaning of events is deceptively simple: an event is triggered when its preconditions become true, affecting the state as if it were an action applied at that instant. Unfortunately, this simple intuition is quickly undermined by several problems:

- If the precondition of an event holds over a continuous interval, what does it mean?
- If the effect of an event causes the precondition of another event to become true, what happens?
- If the preconditions of several events all become true at the same time, what happens?



(a) Event happenings (b) Event chain

Figure 1: Events triggered at the same time point.

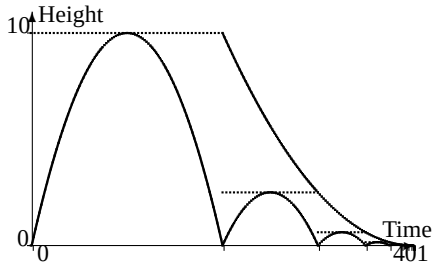


Figure 2: Graph showing the height of a bouncing ball

An answer to the first question is to propose that events are triggered only when their preconditions change from being unsatisfied to being satisfied. However, this requires expensive monitoring of the preconditions of events that have triggered to determine if and when they become false again. In practice, events make most sense when they change the current situation into one in which the event preconditions are no longer satisfied. We therefore introduce this as a formal constraint on events, removing the problem of checking when preconditions become false. The validator highlights any events that fail to falsify their own preconditions.

To answer the other two questions, let us consider events that are triggered at a given point of time. After the execution of an action (or a number of actions) all events that have their preconditions satisfied are executed together as an *event happening* (see figure 1 (a)). These events might trigger another event happening and so on. The event happenings are equivalent to action happenings as defined in (Fox & Long 2003), and are therefore subject to the same execution constraints. In particular, events are subject to *mutex* constraints, which prevent their preconditions and effects from interfering. This issue is further discussed in section 3.2.

We adopt the view that events are causally linked to the actions (or events) that trigger them, so that they succeed their triggering actions (events). This means that we do not consider events in one happening to be mutex with actions that triggered them. Events are caused by state transitions, occurring after the triggering conditions are achieved and as a direct consequence of them, but there is no reason for there to be a delay between the cause and its effect. Therefore, events have to occur at the same time as their causes trigger them. A consequence of this is that event happenings can occur at the same instant as the action happenings that trigger them, but are, nevertheless, considered to occur after the actions that caused them. To address this, we allow multiple happenings to be sequenced at a single time point. This idea is similar to the semantics proposed by Bacchus and Kabanza for actions in TLPlan (Bacchus & Kabanza 2000b) and, by McDermott, for OPTOP (McDermott 2003). The key difference is that we only allow event happenings to stack at the same instant, never actions.

3.2 Cascading Events at a Single Time Point

When multiple events are triggered at a single time point, it is important to resolve precisely what is the outcome. The causal relationship between actions or events that bring about conditions and the events that are triggered by those conditions implies an ordering. However, a single action might trigger multiple events and these obey no implied ordering. In this case, if the triggered events could interact in any way, the final state will depend on the order in which the events are enacted. We consider this situation to be unpredictable: the precise outcome is non-deterministically dependent on the event ordering. Since we are only interested, in this work, in deterministic domains, situations in which events are unordered and interacting are ruled meaningless. To check whether events interact it is only necessary to check whether they are mutex, according to the standard definition (Fox & Long 2003). We note that in complex cascades of interacting events, the pairs that are not ordered must be carefully identified in the partially ordered collection (see figure 1 (b), where a cascade of events follows a single action, A. The pairs (B,C), (B,E), (D,C), (D,E), (F,C) and (F,E) must all be checked for mutex relations.).

3.3 Infinite Sequences of Events

Allowing events to occur ordered at the same time point could lead to a paradoxical situation if events can trigger one another in a cyclic pattern. This situation, which we call *cyclic triggering*, leads to an infinite number of events occurring at a single time point. We eliminate this problem by requiring that no event may be triggered at a single time point more than once.

A slightly different problem can arise if a continuously changing metric fluent (that is, a primitive numeric expression or PNE is linked to an event, where it is possible for the event to be triggered repeatedly at increasing frequency, generating infinitely many events in finite time. For example, a bouncing ball strikes the ground at increasing frequency as the bounces get smaller (see figure 2). We do not have an automatic way to identify this problem and are therefore forced to rely on the domain modeller to prevent it and related paradoxes such as the Thomson Lamp. Such situations can be modelled differently to avoid these problems.

3.4 Modelling Processes

Processes are modelled using the same format as an action, except the effects may only be continuous. Processes are not explicitly part of a plan but are active when and only when their preconditions are true. Unlike events, it is sensible for a process to be active over an interval of time, since the effects are continuous and not instantaneous.

The precondition of a process should not depend on any continuously changing PNEs affected by the process itself.

Consider a tap that automatically flows into a bath whenever the bath volume is less than or equal to 100 units. The process is inactive after the volume exceeds 100 units. However, when the volume is equal to 100 the bath should continue to fill. This is an ambiguous situation: is the process active or not when the level is 100 units? To avoid this difficulty, it is sufficient to restrict process preconditions to contain no PNEs. Processes may still be started or stopped by continuously changing PNEs using intermediary events. (See section 6.2 for an example.)

After the last action of a plan is executed processes might still be active and, some time later, the goal of the plan might become unsatisfied. Since monitoring active processes could be arbitrarily costly, we adopt the position that it is sufficient that the goal of the plan should hold after the last action in the plan (and after any events it triggers), and it is the responsibility of the domain modeller to ensure that this adequately captures the intended effect of the plan. Various modelling techniques allow the modeller to express the need for goals to hold over an interval (Fox, Long, & Halsey 2004).

3.5 A Formal Definition of Events and Processes

We are now in a position to state more precisely the definition of an event and a process.

Definition 3.1 *An* event e *is an instantaneous state transition* (C, E)*, where* C *is a formula expressing the triggering condition and* E *describes the effects of the event. The event* e *is triggered in any state* S *such that* $S \models C$*. Application of the effect* E *results in a state* S' *such that* $S' \not\models C$*.*

Definition 3.2 *An* event cascade triggered in a state S *is a partially ordered finite set of events,* $EC_S = \{e_1, \ldots, e_n\}$*, with conditions* $\{c_1, \ldots, c_n\}$*, such that for every event* e_i *in* EC_S*, either* $S \models c_i$ *or the subset of events ordered before* e_i *achieves a state* S' *such that* $S' \models c_i$*.*

Definition 3.3 *An event cascade* EC_S *triggered in a state* S *is* valid *if every unordered pair of events in* EC_S *is non-mutex.*

Note that the fact that EC_S is a partially ordered set means that no event can be triggered more than once at any given time point.

Definition 3.4 *A* process p *is a precondition,* C*, and a set of continuous effects,* E*, such that the continuous effects are active for every state* S *such that* $S \models C$*.*

4 Plans that Trigger Events

Events are most interesting when they are triggered by continuous change. With the ability to model both continuous change and events it is possible to accurately model interesting real world situations that are otherwise impossible to express. For example, a simple thermostat which operates a heater to regulate temperature. When the temperature is too cold the heater is switched on and the temperature rises, when it is too hot the heater switches off and the temperature begins to drop, and so on.

The extension of the semantics of continuous effects in PDDL to include events is an extension of the *semi-simple plan* as described in (Howey, Long, & Fox 2004). The graph in figure 3 shows how events are triggered by continuously changing PNEs between two discrete happenings. Before an event is triggered the continuously changing PNEs are updated, ensuring that the state is correct at the time of execution. When an event is triggered it may, in general, affect the continuously changing PNEs. Therefore, the interval after the event to the next discrete happening must be checked for any events that may be triggered.

Events triggered by continuous effects are triggered by a continuous function of time crossing some threshold. Consequently the times of such events are calculated from the roots of continuous functions (the values where the function is zero). The value of the roots are always calculated to within a certain specified accuracy, and thus the timing of the event also. The time at which events are triggered can have a significant impact on the execution of a plan, since the ordering of events and actions may cause different events to be triggered in the subsequent plan.

Another important point is whether events are triggered at all. Consider a curve with non-negative values that only touches the axis at a point, and an event is triggered if the curve is non-positive. In this case the event would be triggered, but if the curve had to be strictly negative to trigger then the event would not be triggered. This emphasises that the realisation of the validation process in VAL is subject to the accuracy of computation of event times. Also that where events are triggered by changes in PNEs then the inaccuracies in computing their values may affect the accuracy of the logical state of the world.

The issue of plan robustness is currently being investigated by the authors, including the question of how likely it is that a plan will successfully execute if actions do not execute reliably at the times the plan specifies. Important related work includes (Gupta, Henzinger, & Jagadeesan 1997).

5 The Event Grounding Problem

In a complex domain it is implausible to check the validity of a plan in the face of the events that could be triggered if this relies on considering all groundings of events. Fortunately there are ways by which the number of groundings considered can be reduced. The problem applies equally to processes but we only consider events here for simplicity.

5.1 Efficient Event Grounding

Events can only be triggered when something in the world has changed. Therefore at least one of the parameters of

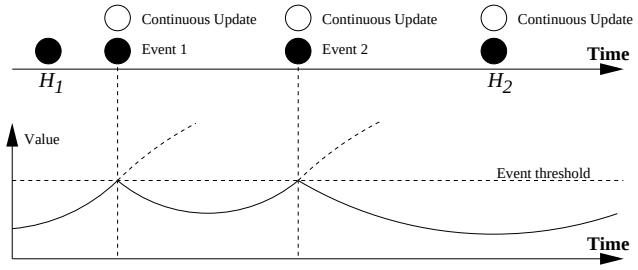


Figure 3: Events triggered by continuous change.

the event being considered must be an object that has just changed state. However, after the changed objects are considered there may still be many undefined parameters: we now consider this problem.

The grounding events problem can be managed through a map used to efficiently calculate which events are triggered at a particular point, returning a set of *parameter lists*. A relatively simple extension of this map is needed to calculate which events are triggered on an interval between discrete happenings due to continuous change.

Definition 5.1 Parameter list *A parameter list is an ordered list of object names for a given unground event written* $(p_1, p_2, \cdots, p_n)$*, where each* p_i *is an object name with the correct type corresponding to the unground event, or an undefined parameter denoted by* $\perp$*.*

An unground event with a parameter list represents a ground event. Using the undefined parameter, $\perp$, we are able to express a set of parameters, where $\perp$ could be any object. The use of $\perp$ is appropriate when a parameter does not affect the truth value of a proposition. It is important to only consider relevant subsets of parameters to avoid exponential blow up.

Definition 5.2 *Let* ψ *be a map from an event,* E*, an unground proposition,* P*, a state* S*, a set of parameter lists,* K_0*, to a set of parameter lists,* K*. Where* K *is the set of parameter lists for* E *given by* K_0 *that satisfy* P *in* S*, written:*

$$\psi(E, P, S, K_0) = K.$$

To calculate the triggered ground events of an unground event, E, in state S, we calculate $\psi(E, P_E, S, K_0)$ where P_E is the precondition of E and K_0 has one parameter list where every parameter is undefined.

Let E be an event, P an unground proposition, S a state, K_0 a set of parameter lists then we define $\psi(E, P, S, K_0)$ as follows depending on the structure of P:

Literal If P is a literal then K is the set of parameter lists derived from K_0 such that P has truth value true. We calculate this by checking in the state which literals are true and then check that these literals can be derived from K_0.

Disjunction If P is a disjunction, $\vee_i X_i$, then $\psi(E, \vee_i X_i, S, K_0) = \cup_i \psi(E, X_i, S, K_0)$. That is, the set of parameter lists given by every disjunct.

Conjunction If P is a conjunction, $\wedge_{i=1\cdots n} X_i$, then K is the set of parameter lists derived from K_0 such that P has truth value true. If we let $\xi(Q, K_0) = \psi(E, Q, S, K_0)$ then this is calculated by evaluating

$$\xi(X_n, \xi(X_{n-1}, \cdots \xi(X_2, \xi(X_1, K_0)))).$$

That is, we calculate the parameter lists derived from K_0 that satisfy X_1, then pass the result on to calculate which of these satisfy X_2, and so on.

This is the most important technique used to calculate which ground events are triggered. If each conjunct refers to a parameter with m possible values, then there are $O(m^n)$ conditions to check. Using the above method we only consider at most $O(mn)$ conditions.

Negation If P is a negation then ψ is applied to the NNF of P. The negation of a literal is slightly more problematic than a positive one, as it is not as easy to exploit the closed world assumption in the state representation. However, it is still straightforward to calculate the parameter lists.

Comparison If P is a comparison then ψ returns the list of parameters that satisfy the comparison, where the set of parameter lists is derived from K_0. Crucially, the parameters that affect the comparison cannot be considered separately since they all affect the same atomic proposition. This implies that we must test the comparison for every set of parameter lists derived from K_0 which affects the comparison. Provided comparisons are limited to only one or two PNEs then this is not a problem.

6 Examples

6.1 Grounding Events Example

Consider a domain with the following event with 20 parameters and 400 possible objects for each parameter, which results in over 10^{52} ground events.

```
(:event grounding-example-event
 :parameters (?x1 ?x2 ... ?x20)
 :precondition (and (property1 ?x1) (property2 ?x2)
        ... ...  (property20 ?x20))
 :effect (not (property1 ?x1)))
```

Suppose that in a state (`property`n `object`n) is true for $n = 2 \cdots 20$, then executing an action to add (`property1 object1`) triggers one ground event. VAL executes this plan in around 0.009 seconds. The same plan executed without the unground event present is around 0.007 seconds, showing no significant change.

However, if a conjunct has more satisfied values then more events are triggered. The table below shows timings when conjuncts have more satisfied values. In practice we only expect a few events to be executed together at any given time, so grounding events does not pose a problem. (In fact the events in this example are mutex, the chances of a huge number of non-mutex events triggering together is remote.)

No. Events Triggered	Time
1	0.009
400	0.10
160 000	30.97

6.2 Solar Power on Mars

Using events and processes it is possible to model the background behaviour of the environment. In this example we consider a model of the solar power on Mars. Although it is possible to model this situation in PDDL2.1 as described in (Fox, Long, & Halsey 2004), this model is very succinct and is, importantly, part of the domain description and not the plan. During the day it is given by a quartic (polynomial of degree 4) peaking at midday, and throughout the night it is zero, see figure 4. Our model in PDDL uses processes for the day and night, and events for sunset and sunrise:

```
(:process day-time
  :parameters ()
  :precondition (daylight)
```

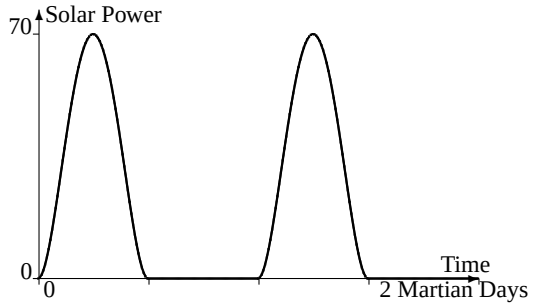


Figure 4: Solar power modelled by processes and events

```
  :effect (and (increase (daytime) (* #t 1))
    (increase (solar-power)
   (* #t (* (* (* (constantA) (daytime)) (constantB))
   (- (* (* (daytime) (daytime)) (constantB)) 1))))))
(:event sunset
  :parameters ()
  :precondition (>= (daytime) (quarterday))
  :effect (and (not (daylight))
               (assign (daytime) (-(quarterday)))))
(:process night-time
  :parameters ()
  :precondition (not (daylight))
  :effect (and (increase (nighttime) (* #t 1))))
(:event sunrise
  :parameters ()
  :precondition (>= (nighttime) (halfday))
  :effect (and (daylight) (assign (nighttime) 0)))
```

The table below shows the time VAL takes to validate an empty plan for a certain number of Martian days. The plan may be empty, but there is still a lot of processes and events to account for from the solar power model. The table shows that execution time is linear to plan length, and in practice we would not expect a plan with over 10^4 events.

These timings coupled with the grounding event timings are very encouraging in that VAL may be feasibly incorporated into planning systems to handle processes and events.

Martian Days	No. Processes and Events	Time
2	10	0.005
20	118	0.017
200	1 198	0.135
2 000	11 998	1.315
20 000	119 998	13.077
200 000	1 199 998	132.018

7 Conclusion

The first step to developing planners that are able to handle planning problems with events is to present an unambiguous semantics. This paper has discussed the semantics of PDDL with events and some of the issues arising. We believe that one of the most practical expressions of the semantics of a complex language is in an implemented form, allowing users to interact with the semantics and explore its consequences empirically. We have implemented the semantics in the automatic plan validation tool, VAL. Figures 2 and 4 illustrate example output. VAL is an important and useful tool for the development of any planner that is to handle events. Not only can it be used to validate plans produced by the planner, but its implementation provides many insights for extending planners to handle events, as well as the possibility of using VAL directly in the planning process itself.

Several problems arise in the semantics and implementation of validating plans using events, but many interesting domains can be modelled by making certain restrictions. These include avoiding cascading events leading to an infinity of events in finite time. The accuracy of PNEs must also be taken into account when they are used. We are currently investigating how to measure the robustness of plans in the face of possible inaccuracy in the timing of execution of actions and in the values of continuously valued PNEs.

The availability of the automatic plan validator, VAL, to validate plans with events is one of the first steps in building planners than can handle events. The scope of planning problems that can be captured using events is greatly increased, supporting much more accurate models of real world situations. The objective is ultimately to bring these aspects together in order to have planners capable of planning with even richer domain representations than at present.

References

Aylett, R.; Soutter, J.; Petley, G.; Chung, P.; and Edwards, D. 2001. Planning plant operating procedures for a chemical plant. *Engineering Applications of Artificial Intelligence* 14(3).

Bacchus, F., and Kabanza, F. 2000a. Using temporal logic to express search control knowledge for planning. *Artificial Intelligence* 116(1-2):123–191.

Bacchus, F., and Kabanza, F. 2000b. Using temporal logic to express search control knowledge for planning. *Artificial Intelligence* 116(1-2):123–191.

Fox, M., and Long, D. 2002. PDDL+ : Planning with time and metric resources. Technical report, University of Strathclyde, UK. Available at: `planning.cis.ac.uk/competition/`.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of AI Research* 20.

Fox, M.; Long, D.; and Halsey, K. 2004. An investigation into the expressive power of PDDL2.1. In *Proceedings of ECAI'04*.

Gupta, V.; Henzinger, T.; and Jagadeesan, R. 1997. Robust timed automata. In *HART'97: Hybrid and Real-time Systems, LNCS 1201*, 331–345. Springer-Verlag.

Henzinger, T. 1996. The theory of hybrid automata. In *Proc. of the 11th Annual Symposium on Logic in Computer Science. Tutorial.*, 278–292. IEEE Computer Soc. Press.

Howey, R., and Long, D. 2003. Validating plans with continuous effects. In *Proc. of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group*, 115–124.

Howey, R.; Long, D.; and Fox, M. 2004. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *Proceedings of 16th IEEE International Conference on Tools with Artificial Intelligence*.

McDermott, D. 2003. Reasoning about autonomous processes in an estimated-regression planner. In *Proc. of Int. Conf. on Automated Planning and Scheduling (ICAPS'03)*.

Shin, J., and Davis, E. 2004. Continuous time in a SAT-based planner. In *Proc. of AAAI-04*.

Thiebaux, S., and Cordier, M.-O. 2001. Supply restoration in power distribution systems – a benchmark for planning under uncertainty. In *6th European Conference on Planning (ECP-01)*.