Validating Plans with Continuous Effects

Richard Howey and **Derek Long**
University of Strathclyde, UK
richard.howey@cis.strath.ac.uk derek.long@cis.strath.ac.uk

Abstract

A critical element in the use of PDDL2.1, the modelling language developed for the International Planning Competition series, has been the common understanding of the semantics of the language. The fact that this has been implemented in plan validation software was vital to the progress of the competition. However, the validation of plans using actions with continuous effects presents new challenges (that precede the challenges presented by *planning* with those effects). In this paper we review the need for continuous effects, their semantics and the problems that arise in validation of plans that include them. We report our progress in implementing the semantics in an extended version of the plan validation software.

1 Introduction

Classical planning research has focussed on logical structure of plans, with temporal structure confined to ordering constraints between activities. With a few notable exceptions, such as (Vere 1983; Penberthy & Weld 1994; Laborie & Ghallab 1995; Muscettola 1994), metric temporal structure has not been considered until recently, despite its obvious importance in practical planning problems. The third International Planning Competition made temporal planning one of the focal objectives and a number of planners achieved remarkable success in handling quite complex metric temporal planning behaviour, including MIPS (Edelkamp & Helmert 2000) and LPG (Gerevini & Serina 2002). Although the introduction of metric temporal reasoning was a considerable challenge, there remain important additional challenges. For example, in the domains used in the competition, all change was modelled using discrete effects. There are features of some domains that cannot be accurately modelled with discrete effects. In this paper we review the need for continuous effects to adequately model certain kinds of problems. Planning with continuous effects was previously discussed at the competition workshop 2003 (Howey & Long 2003) and a more in depth account is in preparation.

One of the key elements contributing to the success of the competition was the initial definition of a semantics for PDDL2.1 allowing a general understanding of what constitutes a valid plan and, crucially, the implementation of an automatic plan validator, VAL. Over 5000 plans were generated by entrants to the competition, so it was clearly essential, for evaluation of the results alone, to have an automatic validator. In fact, the role of VAL in communicating the practical significance of the semantic definitions to entrants should not be underestimated — it was a vital tool in the development process for all of the competitors. In this paper we consider the problem of extending VAL to handle continuous change. We consider the semantics on which this extension is based and explore the practical problems that are implied in implementing this to achieve automatic validation both efficiently and correctly.

2 Actions with Continuous Effects

Various authors have considered the representation of continuous effects in planning contexts. Pednault (Pednault 1986) proposed an extended representation language in which actions could initiate continuous processes such as rotation or linear motion, described by continuous functions parameterised by time. In Zeno, Penberthy and Weld (Penberthy & Weld 1994) considered actions with continuous effects described by differential equations for the evolution of continuously changing values such as fuel. Shanahan has extended the event calculus to include continuous change (Shanahan 1990), while proposing it as a model for planning. A functional representation (Trinquart & Ghallab 2001) of domain models has also been proposed. In this paper we are not too concerned with the precise syntactic representation of continuous change, although we supply examples using PDDL2.1 (Fox & Long 2002), the language developed for the series of international planning competitions (McDermott & Committee 1998). Instead we concentrate on the semantics that underlie the use of continuous effects.

We assume that time runs on a continuous real time line. Actions can instantaneously initiate continuous change in numeric-valued variables. This change will be in effect over finite intervals within the structure of a plan (we assume plans are finite structures and execute in finite time). The description of change is assumed to be given by describing the instantaneous changes made to the values of derivatives of variables (with respect to time). The use of derivatives has two advantages: the changes can be seen as instantaneous effects, despite the fact that they lead, indirectly, to continuous change, and the effects of concurrent actions that interact in their effects on variables is simplified compared with models that require explicit statement of the function describing a variable as a function of time. For example, the description of the effect of an action of driving between two locations can be described independently of whether a concurrent activity changes the velocity of the vehicle. In addition to the possibility of expressing continuous change in this way, we also assume that a plan can be constrained to maintain particular invariant conditions over certain intervals. The need to preserve invariants can arise for various reasons. In PDDL2.1 invariants can be associated with the execution of durative actions over the interval of their execution, but it is also reasonable to suppose that constraints could express safety conditions required for the successful execution of a plan, or goals over intervals.

A continuous effect can only affect metric quantities: it is not possible to change a propositional fluent continuously. A metric variable that can be changed by a continuous effect is called a *Primitive Numerical Expression (PNE)*. A durative action that has a continuous effect on a PNE changes the fluent so that the values taken once the continuous change is activated are described by a continuous function of time. That is if v is changing continuously on an interval $[t_1, t_2]$ then for each $t' \in [t_1, t_2]$ the limit $\lim_{t \to t'} v(t)$ exists and is equal to $v(t')$. It is possible for other actions to affect a PNE during the interval over which a continuous effect is changing it. In this case, the compound continuous effect will be decomposed into segments of continuous behaviour, punctuated by points of discrete change. These points can be either discrete changes in the value of the PNE itself, where an action assigns directly to the PNE, so that the value describes a discontinuous behaviour, or can be discrete changes in the rate of change so that the value describes a piece-wise continuous, but non-differentiable behaviour. The latter case occurs when an action modifies (instantaneously) the derivative of a PNE.

For the purposes of validation it is important to observe that discontinuities, either in the value of a PNE or in its derivative, can only occur at points corresponding to the instantaneous points of effects of actions — in PDDL2.1 either the starts or ends of durative actions, or the points of application of simple actions. These points are defined by the structure of a plan, so can be identified before validation needs to consider continuous effects and can be used to segment the behaviour of metric quantities within a plan into intervals of continuous and differentiable behaviour.

3 Syntax of Continuous Effects

A continuous effect of a durative action is written in the following style:

```
(increase (volume ?v) (* #t (refuel-rate
                 ?p)))
```

where `#t` represents the time over which the effect has been active. However, the whole expression must be interpreted as having a special significance that cannot be accurately captured in terms of an instantaneous effect: instead, the expression should be thought of as identifying a rate of change for the PNE on its left. In this example the volume of `v` is continuously increased at the refuel rate of the fuel pump `p`. If the fuel pump delivers fuel at a constant rate then the volume at any point during refuelling has been increaesed by the time since refuelling started, `#t`, multiplied by the refuel rate. If the refuel rate is itself changing then the behaviour is more complex, described by a differential equation. It might seem more natural to express this effect as an assertion of the form

$$\frac{d}{dt}\texttt{(volume ?v)} = \texttt{(refuel-rate ?p)}$$

but this would be inappropriate since there might be additional actions that affect the value of the volume continuously and concurrently. While it would not be inconsistent for each of those actions to assert that they had the effect of increasing volume at some rate, it would be inconsistent for any of them to assert a specific value for the overall rate of change of the volume.

Formally continuous effects are written as follows:

```
(<assign-op-t> <f-head> <f-exp-t>)
```

where

```
<assign-op-t>  ::= increase
<assign-op-t>  ::= decrease
<f-exp-t>      ::= (* <f-exp> #t)
<f-exp-t>      ::= (* #t <f-exp>)
<f-exp-t>      ::= #t
```

and `<f-head>` is a PNE and `<f-exp>` is any expression (for details see PDDL2.1 (Fox & Long 2002)).

4 Motivation

Before considering continuous effects in more detail, it is worth reviewing why one should consider it necessary to allow for them. Many domains can be modelled adequately by treating effects that are in reality continuous as though they were discrete effects at the start or end of a durative action. Unfortunately, there are problems in which this is not true and in which a plan can only be created if continuous effects are expressible and properly accounted for. To demonstrate this let us consider an example in some detail for the remainder of this section.

Consider a generator that must run continuously for 100 time units, requiring 1 unit of fuel per time unit, but with a capacity of only 60 units of fuel. Two tanks containing 25 units of fuel are available that can be emptied into the generator while it is generating, but the volume of fuel in the generator, which is initially full, cannot exceed its capacity. Only one tank may be emptied into the generator at a time. The rate of flow of fuel out of the bottom of tank is governed by Torricelli's Law: *Water in an open tank will flow out through a small hole in the bottom with the velocity it would acquire in falling freely from the water level to the hole*. The volume of fuel in the tank, V, is then shown to be given by

$$\frac{dV}{dt} = 2k(kt - \sqrt{U}), \qquad (1)$$

where k is the flow constant of the tank (this depends on the size of the hole, gravity, etc), U is the initial volume of fuel in the tank and t is time. At time 0 the tank has volume

Problem
```
(:objects generator tank1 tank2)
    (:init (= (fuel-volume generator)  60)
           (= (capacity generator)  60)
           (= (fuel-volume tank1) 25)
           (= (sqrtvolinit tank1) 5)
           (= (flow-constant tank1) 0.2)
           (= (fuel-volume tank2) 25)
           (= (sqrtvolinit tank2) 5)
           (= (flow-constant tank2) 0.4))
     (:goal (generator-ran))
     (:metric (total-time)))
```

Figure 2: Encoding in PDDL2.1 of the generator problem.

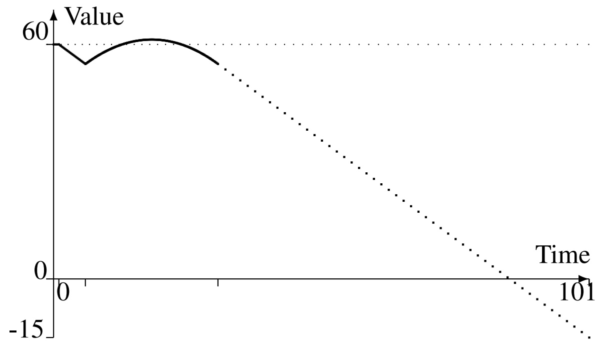

Figure 3: Graph of (fuel-level generator).

U and starts to drain. Then by time $\frac{\sqrt{U}}{k}$ the tank is empty so that these equations are only valid for time values, t, in $[0, \frac{\sqrt{U}}{k}]$. The rate at which the tank empties is fast to begin and then slows to a dribble as it finally empties. A possible encoding in PDDL2.1 using continuous effects is shown in figures 1 and 2.

The domain consists of two durative actions. One models the generator running for 100 time units, with the condition that the generator must not run out of fuel. The other durative action models the emptying of a tank of fuel into the generator for a fixed time with the condition that the generator must not overflow. The square root of the initial volume of a tank, `(sqrtvolinit ?t)`, is required for the equation describing the flow of fuel out of the tank. This must be given in the first instance since PDDL2.1 does not handle the use of square roots. The square root of the volume of the tank, `(sqrtvol ?t)`, can then be tracked by a linear function of time which then can be used to supply the square root of the initial volume of the tank in the case that the tank is partially drained and then drained later.

4.1 Calculating a Plan

Now let us consider solving this problem, clearly the first thing to do is to set the generator running for 100 time units. So let us start the generator running for 100 time units after time 0, say at time 1:

```
1: (generate generator) [100]
...
```

The generator uses one fuel unit per time unit, so the generator needs 100 fuel units to generate for 100 time units. The generator starts with 60 fuel units and each tank has 25 fuel units, so we have 110 fuel units to use, which is good. We can try to empty both tanks into the generator – but how long does each tank take to empty? From the definition of the refuelling durative action and the constants for the first tank, given in the problem definition, the rate of change of the volume of fuel of the first tank, V_1, is given by

$$\frac{dV_1}{dt} = 0.08t - 2.$$

Therefore, the volume of fuel is given by

$$V_1(t) = 0.04t^2 - 2t + 25.$$

Solving this equation we see that the tank is empty at $t = 25$. However when the model of refuelling was constructed for the domain we already observed that the tank becomes empty at the time given by the square root of the initial volume of fuel in the tank divided by the flow constant. For a planner working with this domain this fact is not obvious. Similarly the second tank takes 12.5 time units to empty.

Now let us add the refuelling action for the generator, using the first tank. We will first consider refuelling 5 time units after the generator has started running.

```
1: (generate generator) [100]
6: (refuel generator tank1) [25]
...
```

During the period of refuelling the volume of fuel of the generator, V, is given by the sum of the effects of the generator generating and the tank refuelling the generator:

$$\frac{dV}{dt} = (-1) + (2 - 0.08t).$$

Thus

$$V(t) = -0.04t^2 + t + 55 \qquad \text{for } t \text{ in } [0, 25],$$

where t is the local time for the refuelling durative action. We use the condition that V is 55 at $t = 0$ since we know that the generator has already used 5 units of fuel. The refuelling durative action has an invariant condition that the generator must not overflow which is given by

$$-0.04t^2 + t + 55 \leq 60 \qquad \text{for } t \text{ in } (0, 25),$$

which is equivalent to

$$0.04t^2 - t + 5 \geq 0 \qquad \text{for } t \text{ in } (0, 25).$$

However this condition only holds for values of t in $(0, 6.91] \cup [18.09, 25)$ (values to 2 decimal places). So, we have started to refuel the generator too soon, overflowing the generator. This is shown in figure 3. What is the earliest time that refuelling could occur? Let the volume of fuel of the generator at the start of refuelling be v_0. Then the volume of fuel in the generator is given by

$$V(t) = -0.04t^2 + t + v_0.$$

The invariant condition for the generator not to overflow is then given by

$$0.04t^2 - t - v_0 + 60 > 0 \qquad \text{for } t \text{ in } (0, 25).$$

Durative actions

```
(:durative-action generate
 :parameters (?g)
 :duration (= ?duration  100)
 :condition (over all (> (fuel-volume ?g) 0))
 :effect (and (decrease (fuel-volume ?g) (* #t 1))
      (at end (generator-ran))))

(:durative-action refuel
 :parameters (?g ?t)
 :duration  (<= ?duration (/ (sqrtvolinit ?t) (flow-constant ?t)))
 :condition (and (at start (not (refuelling ?g)))
                 (over all (<= (fuel-volume ?g) (capacity ?g))))
 :effect (and  (at start (refuelling ?g))
               (at start (assign (refuel-time ?t) 0))
               (at start (assign (sqrtvol ?t) (sqrtvolinit ?t)))
               (increase (refuel-time ?t) (* #t 1))
               (decrease (sqrtvol ?t) (* #t (flow-constant ?t)))
               (decrease (fuel-volume ?t) (* #t (* (* 2 (flow-constant ?t))
                  (- (sqrtvolinit ?t) (* (flow-constant ?t) (refuel-time ?t))))))
               (increase (fuel-volume ?g) (* #t (* (* 2 (flow-constant ?t))
                  (- (sqrtvolinit ?t) (* (flow-constant ?t) (refuel-time ?t))))))
               (at end (not (refuelling ?g)))
               (at end (assign (sqrtvolinit ?t) (sqrtvol ?t)))))
```

Figure 1: Encoding in PDDL2.1 of the generator domain using durative actions

The roots of this quadratic are

$$\frac{1 \pm \sqrt{-8.6 + 0.16 v_0}}{0.08}.$$

For the refuelling action to occur as soon as possible we want this invariant to only just hold so that the volume of fuel in the generator is at capacity. This occurs when $v_0 = \frac{8.6}{0.16} = 53.75$, and from this we know the generator must have been running for 6.25 time units for the volume of fuel to drop to this amount in the generator. Therefore the earliest we can refuel using all of the first tank is at time 7.25.

```
1: (generate generator) [100]
7.25: (refuel generator tank1) [25]
...
```

Now, we still wish to do another refuelling before the end of generating, the latest we could do this is at time 88.5 in the plan. However the volume of fuel of the generator is -2.5 by this time, meaning, of course, that the generator has since run out fuel and the plan is not valid. If we wanted to delay to the last moment then the second refuelling would occur before time 86, this being the time when the generator runs out of fuel which can be deduced from the rate at which fuel is consumed and the previous volume of fuel since the last refuelling. So our final plan could be

```
1: (generate generator) [100]
7.25: (refuel generator tank1) [25]
85.99: (refuel generator tank2) [12.5]
```

The volume of fuel of the generator for this plan is shown in figure 4.

It would be interesting to see a planner capable of producing a plan that includes the earliest or latest possible refuelling, although such a plan cannot be considered very robust. A better behaviour would be to push the refuelling actions comfortably into the generating interval.

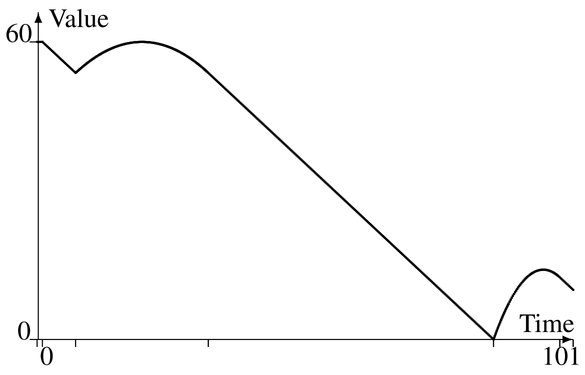

Figure 4: Graph of (fuel-level generator).

4.2 Calculating a Plan Using Bounds

A more feasible approach to calculating a plan for a problem involving continuous effects may be to use linear approximations on the continuous effects, in particular the use of step functions. For the generator problem, if we reason that a tank is only going to add 25 fuel units then we can wait until the generator has used up 25 fuel units, so that we know that no overflow could occur. The first refuelling would then take place at time 26. Next, the second refuelling can occur as soon as the first refuelling has finished, since we know that the volume of fuel in the generator is 35, so that no overflow could occur. Figure 5 shows the volume of fuel of the generator for this plan.

```
1: (generate generator) [100]
26: (refuel generator tank1) [25]
```

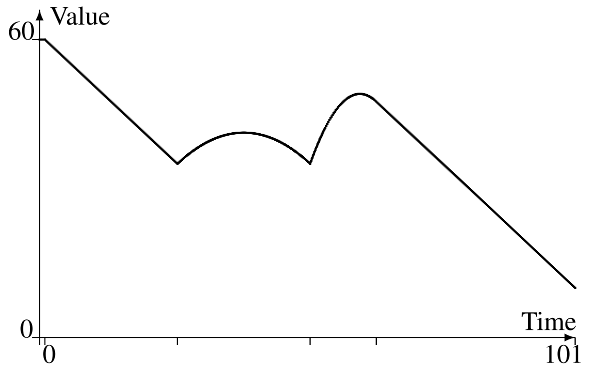

Figure 5: Graph of (fuel-level generator).

```
51.01: (refuel generator tank2) [12.5]
```

Now consider what was involved in constructing this plan. The volume of fuel in the generator was calculated as a function of time from the differential equation governing its behaviour, but with no other continuous effects interacting with the volume of fuel. We then considered the refuelling action for the first tank on its own, solving the differential equation in order to figure out how long it takes to empty the tank. A step function was then used to approximate the refuelling action, adding all of the fuel at the start of the durative action. The linear function for the generator fuel volume was then used to determine when the refuelling given by the step function could be applied. Similarly with the second tank of fuel. Notice that we did not use a step function for the generating action.

If this kind reasoning can be used to produce a plan using simple calculations to handle linear functions and linear bounds for non-linear functions then this makes the planning process much easier. However using linear bounds can greatly reduce the flexibility of durative actions within a plan, the more accurate continuous effects can be handled in the planning process the better the scope of valid plans that can be produced. In the generator example we saw that the earliest time a refuelling could occur using the first tank is at time 7.25. However, using linear bounds for the volume of fuel in the generator (by way of a step function on the refuelling action), the earliest safe time to refuel is at time 26. This reduction in flexibility might suggest that no valid plan exists at all when there are, in fact, many valid plans.

If we attempt to model the fuel-consumption of the generator with a step function we have a problem: either the fuel is consumed at the start, which suggests that a tank can simply transfer its 25 units of fuel to the generator at the start without exceeding the generator fuel capacity. Alternatively, if we put the consumption at the end then we cannot refuel the generator until it has ran out of fuel. In order to solve the problem we have to have access to an accurate (or at least a sufficiently accurate) model of the fuel in the generator throughout its generating time.

5 Semantics of Continuous Effects

We do not attempt to give a formal semantics in this section, due to limited space: a formal semantics has been developed, based on the semantics of discrete durative actions (Fox & Long 2002). The semantics of discrete durative actions can be formulated in terms of discrete state changes at the instants of change, in a familiar state-transition semantic framework, but with the addition of an embedding of the activity into a real time line. This is a straightforward extension, except for two important complications introduced by the embedding: one is to explain under what circumstances the end points of actions (when instantaneous change occurs) may coincide and the other is to account for the way in which the interaction between action execution and invariants is handled. The first of these issues is resolved by preventing action end points from coinciding if the postconditions of one of the end points include any proposition that is included in the preconditions of the other. This constraint implies that propositions are treated like shared memory in multi-processing operating systems, with actions analogous to separate processes. An action precondition demands *read-access* to all of the atomic propositions it includes, while action postconditions demand *write-access* to all of the atomic propositions they refer to. A propositional variable can support multiple coincident read-accesses, but a write-access prevents any other access to the propositional variable. An action can, of course, refer to the same proposition in both its pre- and postconditions, just as single process can read and then write to a memory, because its own memory accesses are sequenced. An alternative approach, adopted by McDermott (McDermott 2003) and Bacchus and Ady (Bacchus & Ady 2001), is to allow actions that occur simultaneously to be sequenced. We consider this approach difficult to justify, since it is not clear how, in execution, simultaneous actions could actually be guaranteed to execute in a specific order. A plan contains a finite number of actions. Thus, the management of invariants is handled by observing that it is only necessary to confirm each invariant between the finitely many discrete points of change that occur during the interval to which it applies.

The introduction of continuous change creates two further complications: continuous changes can interact and continuous change can affect values appearing in invariant conditions during the period of continuous change. A semantic account can be constructed using a timed hybrid automaton model, such as Henzinger's (Henzinger 1996). This model is attractive because it is close to the familiar state transition semantics for planning problems. The key extension to the discrete temporal model is that interacting processes are described by systems of differential equations, whose effects can be resolved at the conclusion of each interval over which they act uninterrupted. Invariants can be checked by considering the functions describing the change in PNEs over the same intervals. This is illustrated in figure 6. In (1) we show how the interval of a durative action effecting continuous change can be handled by updating the continuously changing PNEs discretely at the end of the interval and checking any invariants at this point. Each invariant check is responsible for confirming correctness over the preceding interval of continuous change. In (2) we show that if another action end point punctuates the interval then the evaluation of continuous effects and invariant checks are managed at each

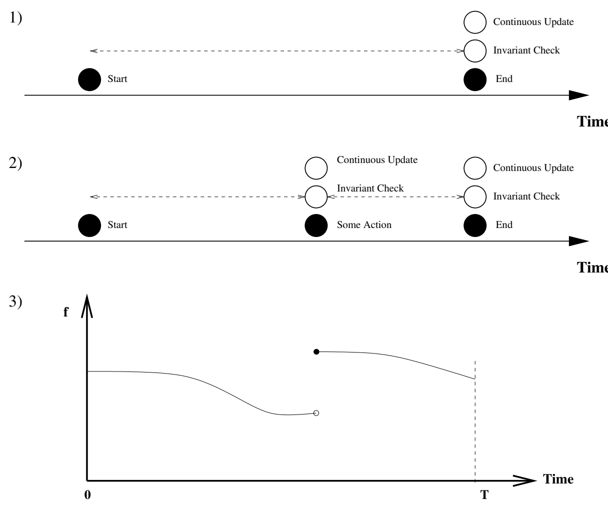

Figure 6: Durative action with continuous effects. Graph shows the values of a PNE, f, which is changing continuously during the execution of the durative action

point of change. Part (3) illustrates how discrete affects can arise, due to parallel activity during the intervals, breaking the continuous change into piece-wise continuous differential components.

6 Interacting Continuous Effects

There may be a number of continuous effects active at one time each of which additively modifies the derivative of a PNE. If a PNE has its derivative modified more than once then the derivative is given by the sum of the contributions. The rate of change of a PNE may also depend on the value of other PNEs which may themselves be continuously changing. The values of all the changing PNEs are thus given by a system of differential equations:

$$\frac{df_i}{dt} = g_i(f_1, f_2, \cdots, f_n) \qquad i \in \{1, 2, \cdots, n\},$$

where the f_i are the PNEs and the g_i are some functions depending on the set of continuously changing PNEs. PNEs that are not changing continuously are treated as constants. For example consider the following continuous effects

```
increase (distance ?c) (* #t (speed ?c))
increase (speed ?c) (* #t (acceln ?c))
```

which describe the motion of a car driving. The rate of change of the PNE for the distance of the car is given by the PNE for the speed of the car. The PNE for the speed of the car is in turn given by the PNE for the acceleration of the car. To solve these differential equations to give the functions of time describing the motion of the car we must firstly determine the acceleration, then the speed, and lastly the distance of the car.

Solving the system of differential equations that can arise is considered in sections 8.1 and 8.2.

7 Invariants

Continuous effects have their most significant effect on the validation of plans when they interact with invariants. An invariant condition must be evaluated on an interval by checking that the continuously changing PNEs that appear within it do not assume values that lead to a violation of the invariant.

7.1 One-Clause Invariants

An invariant comparison containing PNEs that are continuously changing can always be expressed as a function of time, t, that must be greater than zero (or perhaps greater than or equal to zero. If equality is used then the difference between the left hand side and the right hand side cannot vary for the invariant to hold). For example

$$t^2 + 2t + 2 > 0 \qquad \text{for } t \in (0, 5)$$

may be an invariant condition to check. If the invariant expression is linear in time we can simply evaluate the expression at the end points of the interval to confirm the condition holds. However checking an invariant condition with a non-linear expression in time it is no longer sufficient to check the condition at end points only. An invariant comparison $F(t) > 0$ on $(0, T)$, where F is some continuous function in time, t, can be checked by one of the following methods:

1. Check that $F(0) \geq 0$ and $F(T) \geq 0$ and also check that the value at any stationary points in $(0, T)$ is greater than zero.
2. Check that $F(0) \geq 0$ and $F(T) \geq 0$ and also check to see if any roots exist in $(0, T)$. (If the inequality is non-strict then care is needed in case of repeated roots).

Method 2 is chosen to check non-linear invariants in VAL. The key to the problem of checking invariants that are comparisons with non-linear expressions in t is one of finding the roots of a non-linear function. This problem is, in general, non-trivial, even in the case of polynomials. There are many algorithms to find the roots of equations but we need to be sure of finding all the roots in a given interval in every possible case. It is therefore necessary to impose certain restrictions on the invariants that may be expressed to guarantee that they may be verified on a given interval.

We are initially only considering invariant comparisons which depend on continuously changing PNEs that are given by polynomials in t. For one-clause invariant comparisons which are given by an inequality that is strict we are in fact only interested in the existence of real roots on a given open interval. Finding the roots of polynomials is considered in section 8.3.

7.2 Invariants with Disjunctions

Let A and B be two atoms that depend on time then consider the two conditions

$$A(t) \vee B(t) \text{ for all } t \in (0, T), \qquad (2)$$

$$(A(t) \text{ for all } t \in (0, T)) \vee (B(t) \text{ for all } t \in (0, T)). \qquad (3)$$

It could be the case that $A(t)$ is only satisfied on $(0, \frac{3}{4}T]$ and $B(t)$ on $[\frac{1}{4}T, T)$ so that condition (2) is satisfied but

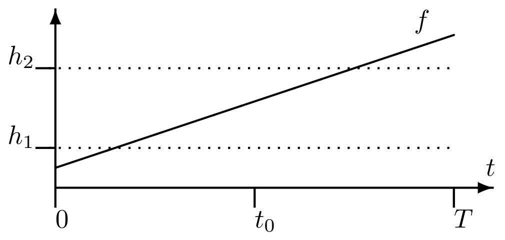

Figure 7: Example of h_1, h_2 and f. If f is required to be above h_2 *or* below h_1 across $(0, T)$, then the value at t_0 breaks the constraint.

condition (3) is not. Clearly the two interpretations are not equivalent. Suppose a durative action continuously updates a PNE, f, and there is a concurrent action (possibly the same action) with an invariant condition of the form:

$$f(t) < h_1(t) \wedge f(t) > h_2(t) \text{ for all } t \in (0, T),$$

where h_1 and h_2 are some functions that may depend on other functional expressions. The condition can then be checked by checking each comparison separately. However suppose the condition is of the form:

$$f(t) < h_1(t) \vee f(t) > h_2(t) \text{ for all } t \in (0, T),$$

then each comparison cannot be considered separately.

In the simple case where h_1 and h_2 are constants and f is linear we can no longer simply check the end points of $h_1 - f$ and $f - h_2$ to be greater than zero. This is because we could have $(h_1 - f)(0) > 0$ and $(f - h_2)(T) > 0$ which satisfies the condition at 0 and T but there could exist a point $t_0 \in (0, T)$ such that $(h_1 - f)(t_0) < 0$ and $(f - h_2)(t_0) < 0$ (see figure 7).

As an example of an invariant with disjunction consider the following:

$$(t^2 - 9t + 14 \geq 0) \vee ((t - 1 > 0) \wedge (-t + 8 \geq 0))$$

for t in $(0, 10)$. We must find the values of t in $(0, 10)$ each disjunct is satisfied on then take the union of the two and see if the result covers $(0, 10)$, which implies the result is in fact equal to $(0, 10)$. The first disjunct, $(t^2 - 9t + 14 \geq 0)$, is satisfied for values of t in $(0, 2] \cup [7, 10)$. The second disjunct is not so straightforward, the condition $(t - 1 > 0)$ is satisfied on $(1, 10)$ and $(-t + 8 \geq 0)$ is satisfied on $(0, 8]$ then taking the intersection of the two we have the disjunct being satisfied for values of t in $(1, 8]$. Now taking the union of the values that the two disjuncts are satisfied on gives $(0, 10)$ which indeed covers $(0, 10)$ showing that the invariant condition holds.

In general an invariant condition can be considered as a proposition in DNF that must hold over an interval, say $(0, T)$. To confirm the invariant, we must then find a set of intervals, C, covering $(0, T)$, such that a disjunct is satisfied in each interval in C. This is simplified if it is possible to find all the roots of all the continuous functions involved, since these points can be used as the end points of the sub-intervals forming the cover. In the case of polynomials this can be achieved (finding the roots of polynomials is considered in section 8.3). Even in the case of polynomials of polynomials, the degree of the largest polynomial is bounded giving us a tractable collection of roots and a tractable problem for validation. Using this approach means that our validation of plans cannot be more accurate than the degree of precision used in the solver. However in practice, all numerical testing, even for linear functions, is subject to the degree of precision supported by our machines (always finite), so the problem of plan validation must always be qualified by an observation of the limitation on the accuracy with which numeric constraints can be checked.

A detail of finding a covering of $(0, T)$ is important, let us call an exact covering of $(0, T)$ *fractured* if there is a value, x, other than 0 or T, such that for every $I \in C$ if $x \in I$ then x is an end point of I. (The intuition is that no interval spans between the left and right of the interval across x). If the only exact coverings that exist to satisfy a given invariant are fractured at x then this implies that at x the invariant flips from being satisfied in one way to being satisfied in another. This flip could depend on arbitrarily precise synchronisation of the activities governing the continuous change and this is unlikely to be within the power of any physical executive. For this reason, we might prefer to discount such plans as unrobust. This issue is also related to numeric precision in handling values, both in a potential executive and in the machinery expected to validate a plan.

Two things (other than the form of the functions that describe the behaviours of the PNEs that appear in the proposition) affect the complexity of invariant checking: one is the complexity of the functions that appear in the proposition itself and the other is whether or not there is more than one disjunct. The former is restricted to be (in ascending order of complexity) either simple linear functions, polynomials or other functions.

8 Solving Differential Equations and Evaluating Invariants

In this section we consider approaches to solving a system of differential equations (sections 8.1 and 8.2), which arises when validating a plan with continuous effects. Then finding the roots of polynomials is addressed (section 8.3) as required for evaluating invariant conditions. Then finally, a general method (section 8.5) using the solutions to these problems is presented for evaluating invariants that depend on continuously changing PNEs.

8.1 Numerical Methods

It is not possible to solve each system of differential equations analytically that may arise from continuous effect expressions in PDDL2.1. The subject of finding analytic solutions to a system of differential equations has a *huge* literature and will not be discussed in any detail in this paper. Analytic methods fall well short of a complete solution and can be quite complicated, too specific, too long or non-existent for some systems of differential equations. Therefore analytic methods are not well suited to complicated systems of differential equations that may arise naturally in a mathematical model. For these reasons numerical methods are very popular among scientists working with mathematical models described by differential equations. The output of numerical methods is a finite set of points which is close to the solution curve and ideally to within a given degree of accuracy. If this could be achieved then it would be simple to evaluate invariant conditions by considering this set of points. Thus if we could achieve this for our system of differential equations then this would be ideal. Unfortunately a method to do this infallibly to within a degree of accuracy for such a general system of differential equations simply does not exist. The vast amount of work on the subject is a testament to this. A more realistic and desirable result would be a set of simple restrictions that could be imposed on the system of differential equations that could be easily checked whilst allowing sufficient expressibility of differential equations. These restrictions could then guarantee that an efficient and infallible method could be used to obtain a solution to within a degree of accuracy (or at least the detection that no solution exists). However to the authors' knowledge and disappointment no such result exists. Numerical methods are ideal if the system of differential equations is known and fixed (to within certain parameters), since then numerical methods can then be applied and guaranteed to find the an approximate solution to within a degree of accuracy. In our general setting we do not have this luxury so numerical methods are not appropriate, we therefore consider certain classes of systems of differential equations that may be solved analytically whilst allowing for sufficiently interesting planning problems.

8.2 Analytic Methods

A PNE can change as a polynomial function of time or even a more complex function of time, due to more complex dependencies arising in the expression on the right-hand-side of the differential equation governing its evolution. In particular for a non-linear function to arise, the right-hand-side must include one or more of the PNEs that appear as left-hand-side elements (including the PNE that is governed by this equation itself).

For example consider the simple case where a PNE, f, varies with the rate of change given by:

$$\frac{df}{dt} = f$$

and f has a value of f_0 for $t = 0$. Then the value of f is given by

$$f(t) = f_0 e^t.$$

The following proposition shows that the values taken by PNEs are given by polynomials if certain restrictions are imposed on the differential equations.

Proposition 8.1 *Let $F = \{f_1, f_2, \ldots, f_n\}$ be a finite set of PNEs which are changing continuously on the interval $[0, T]$ given by*

$$\frac{df_i}{dt} = g_i(f_1, f_2, \ldots, f_n) \text{ for all } i \in \{1, 2, \ldots, n\}$$

where g_i is some function depending on F. The function g_i is restricted to addition, subtraction, multiplication and division on its terms and division by a PNE in F is not permitted. If the rate of change of no PNE depends on itself (either directly or indirectly) then the value of every PNE on $[0, T]$ is given by a polynomial in t.

Proof. Follows by induction on the dependency structure.

If the conditions in Proposition 8.1 were relaxed to allow division by a functional expression then a PNE could take values given by a natural logarithm. If the dependencies could contain loops then exponential functions could occur as shown above, as well as trigonometric functions and so on.

Notice that determination of the structure of the dependency sets can be carried out automatically, using syntactic analysis of the expression parse trees. To achieve this, a graph is constructed using PNEs as vertices and with directed edges between the expressions on the left of continuous updates and those on the right of the same expression. If this graph is acyclic (which is easily tested) then the differential equations can all be solved with polynomials. The only limitation is that this dependency analysis carried out purely syntactically can be conservative, in that it might be the case that dependencies actually simplify away if expressions can be symbolically manipulated to cancel terms. Since this kind of manipulation is non-trivial, we must assume that the dependencies discovered by syntactic analysis could be more restrictive than the true dependencies. It should also be observed that the dependency analysis must be considered on a case-by-case basis: each interval over which continuous change is active must be separately analysed because the domain might include expressions that, in principle, allow cycles of dependency to be constructed, but no cycles might actually appear amongst effects active in the plan itself.

The complexity of the differential equations that can be expressed far exceeds the practicality of solving them and indeed the feasibility. It is therefore necessary to impose certain restrictions on the differential equations to guarantee that they can be solved.

8.3 Polynomial Root Finding

As discussed in section 7 to evaluate the truth values of invariants it is necessary to find the roots of a polynomial on a given interval to within a given degree of accuracy. In the case of an invariant consisting of a single comparison then it is sufficient to determine only the existence or not of the roots on a given interval.

There are many methods for finding the roots of polynomials, such as Newton's Method and Graeffe's Method, to name just two, which have their advantages and disadvantages. It is a requirement of VAL to validate plans so it is not acceptable to miss the roots of polynomials which then may give the incorrect truth value of an invariant. We are then only concerned with implementing methods that will find the roots of polynomials infallibly (if it cannot guarantee to find the roots on a given interval then this should be detected).

For VAL we have chosen to implement a method based on Descartes' Rule of Signs for several reasons: it provides an infallible solution, it is efficient, and it is relatively simple to implement. This method is subject to the polynomial containing no repeated roots. Fortunately for any given polynomial we can obtain another polynomial with the same roots but without any repeated roots. This is achieved by dividing the polynomial by the greatest common divisor of itself and its derivative. To find the greatest common divisor of two polynomials the Euclidean (a.k.a. Euclid's or divisional) algorithm can be used, although the accuracy of the coefficients must be handled carefully to avoid any spurious results due to rounding in the calculations. The method that we have implemented is based on Rouillier and Zimmermann's algorithm (Rouillier & Zimmermann 2001), which is itself based on algorithms from Collins and Akritas (Collins & Akritas 1976).

8.4 Approximation and Power Series

On a given interval of time where there are a number of PNEs changing continuously, as defined by a system of differential equations, the PNEs may be given by arbitrary non-linear functions of time as mentioned in section 6. These non-linear functions of time may then appear in invariant conditions requiring that the functions be analysed for roots on a given interval, possibly to within a given degree of accuracy. It is then desirable to approximate the function in a form that can be easily used with numerical methods to compute the roots. We, of course, want to approximate the functions by polynomials, and it is a well known result that any continuous function on a closed bounded interval can be uniformly approximated on that interval by polynomials to any degree of accuracy. The most well known method of approximating continuous functions where the derivatives of all orders exist is the use of Taylor series. The Taylor series of a function can be used to obtain a polynomial approximation to within a given degree of accuracy. So, given a non-linear function of time appearing in an invariant condition we can find a polynomial approximation and proceed to check the invariant using polynomial root finding techniques. We have used this approach in VAL to handle simple exponential functions.

8.5 Plan Validation

Plan validation may be broken into segments of continuous change punctuated by a finite number of discrete changes, as discussed in section 5. These segments are given by intervals of time with the continuous change defined by a system of differential equations (as in proposition 8.1), F, and a set of conditions, Inv, which must hold over the interval. Each interval is considered by its local time written $[0, T]$. To evaluate the conditions on the given interval we use the following steps.

Step One In our answer we are assuming that we can find an analytic solution to the system of differential equations. To ensure we can find an analytic solution we impose certain conditions on the system of differential equations, such as the conditions in proposition 8.1 so that the solutions are polynomial. So for each $i = 1, \cdots, n$ we have f_i expressed in terms of t where f_i is continuous on $[0, T]$ and has derivatives of all orders.

Step Two For each atom in a proposition in Inv which is a comparison depending on F such as $h_1(t) > h_2(t)$ for some functions h_1 and h_2, we rearrange the comparison to be zero on the right hand side, $h_1(t) - h_2(t) > 0$. Then letting $g(t) = h_1(t) - h_2(t)$ we can find a polynomial approximation for g to within a given degree of accuracy as mentioned in section 8.4. Thus each comparison is given by a polynomial and a boolean value to whether it is strict or non-strict. (For equality comparisons the polynomial must equal 0 and will be considered as a boolean atom for the purposes of this method).

Step Three For each A in Inv we determine whether it holds on $(0, T)$ as follows:

- **Boolean** If A is a boolean condition then its truth value is immediate.
- **Comparison** If A is a comparison with polynomial g then we can isolate the roots of g (if any exist). The comparison holds on $(0, T)$ if the end points of g are greater than zero and no roots exist on $(0, T)$. Evaluation of g is needed at key points in $(0, T)$ in the case of a non-strict comparison (in case of repeated roots) or if the end points evaluate to zero.
- **Conjunction** If A is a conjunction $A = A_1 \wedge A_2 \wedge \cdots \wedge A_k$ for some $k \in \mathbb{N}$, then we determine if A holds on $(0, T)$ by checking each conjunct in order, 1 to k, as given by these rules (depending on whether it is a boolean, comparison etc.) to whether it holds for all values on $(0, T)$. If one conjunct does not hold on $(0, T)$ then A does not hold on $(0, T)$ and the remaining conjuncts need not be checked.
- **Disjunction** If A is a disjunction $A = A_1 \vee A_2 \vee \cdots \vee A_k$ for some $k \in \mathbb{N}$, then we determine if A holds on $(0, T)$ as follows. Firstly determine the values of t in $(0, T)$ that A_1 holds on as given below, call it J_1. If $J_1 = (0, T)$ then A holds on $(0, T)$. If $J_1 \neq (0, T)$ then we calculate the values of t that A_2 holds on, J_2, then if $J_1 \cup J_2 = (0, T)$ then A holds on $(0, T)$ and so on. If $\bigcup_{i=1}^{k} J_i \neq (0, T)$ then A does not hold on $(0, T)$.

Values of t in $(0, T)$ that a disjunct A_i holds on:

- **Boolean** If A_i is a boolean value then A_i holds on $(0, T)$ if A_i is true, otherwise it holds on the empty set.
- **Comparison** If A_i is a comparison then from its corresponding polynomial as given in step two, we firstly find the roots in $(0, T)$. These roots are used to determine the end points of a set of intervals that the comparison holds on. If the comparison is strict then the intervals will be open, otherwise the interval end points will be closed excluding 0 and T.
- **Conjunction** If A_i is a conjunction $A_i = B_1 \wedge B_2 \wedge \cdots \wedge B_k$ for some $k \in \mathbb{N}$, then we determine the values of t that A_i holds on $(0, T)$ as follows. In order, for each B_i determine the values of t in $(0, T)$ it holds on (by these meth-

ods) call it J_i. The J_i's are used to calculate $\bigcap_{i=1\cdots k} J_i$, and so if there is a conjunct B_j such that $\bigcap_{i=1\cdots j} J_i = \emptyset$ then the remaining J_i are not calculated.

- **Disjunction** If A_i is a conjunction $A_i = B_1 \vee B_2 \vee \cdots \vee B_k$ for some $k \in \mathbb{N}$, then we determine the values of t that A_i holds on $(0, T)$ as follows. In order, for each B_i determine the values of t in $(0, T)$ it holds on (by these methods) call it J_i. The J_i's are used to calculate $\bigcup_{i=1\cdots k} J_i$, and so if there is a conjunct B_j such that $\bigcup_{i=1\cdots j} J_i = (0, T)$ then the remaining J_i are not calculated.

For each conjunction and disjunction it is not always necessary to consider every conjunct or disjunct, therefore in order to save on calculation time we sort the conjuncts and disjuncts into order of estimated computational effort.

9 Conclusion

This paper examines the problem of validation of plans with continuous effects. We have implemented this approach in an extension of VAL used in the 3rd IPC (Long, Cresswell, & Howey 2003). Currently, to guarantee the validation of a plan containing durative actions with continuous effects certain restrictions have to be met: all continuous effects must be given by a polynomial or a simple exponential function of time. This condition implies that the dependency graph of the rates of change of PNEs has no loops except for self dependent loops. This condition is automatically checked and VAL identifies plans that it cannot correctly validate.

In section 8 a general framework was presented to handle more complicated continuous effects; a more thorough account is in preparation. Although VAL does not currently handle a wide range of types of functions, methods have been developed to do so and these methods have been implemented in the handling of exponential functions. The scope of the continuous functions that VAL should handle can be seen as a prerequisite to developing planners capable of producing plans with those continuous functions. Currently, to the authors' knowledge, the few planners handling continuous effects only consider linear effects. The next logical step is to develop planners capable of handling non-linear effects that are given by polynomials. This is now a much more realistic target with the availability of an automatic plan validation tool, VAL, capable of handling plans with polynomial effects. The extension of VAL to handle yet more complex functions at this stage would be considered overkill.

The validation of plans containing continuous effects is an important first step in making planners capable of planning with languages that express them. Validation depends on semantics and cannot be implemented without removing ambiguities. The availability of a validation tool is a vital first step for the community in progressing along this path.

References

Bacchus, F., and Ady, M. 2001. Planning with resources and concurrency: A forward chaining approach. In *Proceedings of IJCAI'01*, 417–424.

Collins, G., and Akritas, A. 1976. Polynomial real root isolation using Descartes' rule of signs. *SYMSAC*.

Edelkamp, S., and Helmert, M. 2000. On the implementation of Mips. In *Proc. of Workshop on Decision-Theoretic Planning, AI Planning and Scheduling (AIPS)*, 18–25. AAAI-Press.

Fox, M., and Long, D. 2002. PDDL2.1: An extension to PDDL for expressing temporal planning domains. Technical report.

Gerevini, A., and Serina, I. 2002. LPG: A planner based on local search for planning graphs. In *Proc. of 6th Int. Conf. on AI Planning Systems (AIPS'02)*. AAAI Press.

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. VAL's progress: The automatic validation tool for PDDL2.1 used in the international planning competition. In *Proc. of ICAPS Workshop on the IPC*.

Laborie, P., and Ghallab, M. 1995. Planning with sharable resource constraints. In *Proc. of 14th International Joint Conference on AI*. Morgan Kaufmann.

Long, D.; Cresswell, S.; and Howey, R. 2003. Validator for PDDL2.1 plans. Available at `www.cis.strath.ac.uk/~rh/val.html`.

McDermott, D., and Committee, A. I. 1998. PDDL–the planning domain definition language. Technical report, Available at: `www.cs.yale.edu/homes/dvm`.

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

Muscettola, N. 1994. HSTS: Integrating planning and scheduling. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling*. San Mateo, CA: Morgan Kaufmann. 169–212.

Pednault, E. 1986. Formulating multiagent, dynamic-world problems in the classical planning framework. In Georgeff, M., and Lansky, A., eds., *Proc. of the Timberline Oregon Workshop on Reasoning about Actions and Plans*.

Penberthy, J., and Weld, D. 1994. Temporal planning with continuous change. In *Proc. of 12th National Conf. on AI*.

Rouillier, F., and Zimmermann, P. 2001. Efficient isolation of a polynomial real roots. Technical Report 4113, Rinria.

Shanahan, M. 1990. Representing continuous change in the event calculus. In *Proceedings of ECAI'90*, 598–603.

Trinquart, R., and Ghallab, M. 2001. An extended functional representation in temporal planning: Towards continuous change. In *Proc. of ECP-01*.

Vere, S. 1983. Planning in time: Windows and durations for activities and goals. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 5.