Switches and Jumps in Hybrid Action Systems

Mauno Rönkkö

Turku Centre for Computer Science, Dept. of Computer Science,
Åbo Akademi University,
Lemminkäisenkatu 14A, FIN-20520 Turku, Finland,
`mronkko@abo.fi`

Anders P. Ravn

Department of Information Technology,
Technical University of Denmark,
Bldg. 344, DK-2800 Lyngby, Denmark,
`apr@it.dtu.dk`

Turku Centre for Computer Science
TUCS Technical Report No 152
December 1997

ISBN 952-12-0126-6
ISSN 1239-1891

Abstract

Hybrid action systems extend a conventional action system with a guarded differential equation — a *differential action* — that defines evolution of continuous variables while the guard remains true. The new action is given a weakest liberal precondition semantics, which is illustrated by examples from a train model.

The example motivates a discussion of priorities in a system of iterated differential and ordinary actions. The conclusion is a proposal for a standard form with urgency of discrete actions. The result is compared with Branicky's classification in "may" and "must" jumps or switches. The asynchronous "may" switches can be modelled as interrupts of evolutions.

Keywords: weakest liberal precondition, actions, hybrid systems

TUCS Research Group
Programming Methodology Research Group

1 Introduction

Hybrid systems are mathematical models for dynamical systems that exhibit behaviours which alternates at aperiodic instants of time between smooth evolution and discrete transition. Typical examples of such systems are found in the area of hierarchical, computer based control of complex, composite physical systems like modern vehicles, chemical plants, or integrated manufacturing facilities. A description and design for such systems requires a combination of theories and techniques from the fields of control and software engineering.

In this paper we use the established *Action Systems* framework [2] extended with a differential action [9] to define and discuss various combinations of discrete transitions and continuous evolutions. The main result is a characterisation of the different jumps in Branicky's unified model [5] by iterated, non-deterministically composed guarded discrete and differential actions.

Overview.

We start by defining actions, including differential actions in Section 2. Their effect is illustrated by a train example. In Section 3 we study alternation of differential and discrete actions. At the end we introduce action systems and identify a class of action systems that are suitable for modelling and analysis of hybrid systems. Next, Section 4 investigates the relation between hybrid action systems and Branicky's unified model for hybrid systems [5]. The link is illustrated by a hysteresis example. Finally, follows the conclusion in Section 5.

2 Actions

An action is any statement in Dijkstra's guarded command language [8]. Also the pure guarded commands can be used as actions. In a system, the actions operate on a fixed set of variables, who when assigned values identify a state in the system. Therefore, predicates over the variables provide a convenient way of specifying and reasoning about states in a system. Such a predicate is called a condition.

Given an action A, we may desire a *postcondition* q to hold after the execution. The meaning or the semantics for the action A can thus be given by a mapping between predicates, for instance the *weakest liberal precondition* transformer, 'wlp (,)', which maps q to $\mathrm{wlp}\,(A,\ q)$. This predicate describes the largest set of states from which A either reaches the postcondition q or fails to terminate.

Termination is defined by another transformer, 't'. The set of states from which an action A terminates, i.e. reaches an arbitrary state, is given by $\mathrm{t}A$.

When reasoning about sequential programs termination is most useful, so one uses the *weakest precondition* transformer, which Dijkstra [8] defines by $\mathrm{wp}\,(A,\ q) \ = \ \mathrm{wlp}\,(A,\ q) \ \wedge \ \mathrm{t}A$.

The weakest precondition defines a basic property for any action: The set of states in which an action is enabled, i.e. the states from which an action A may start, is given by $\mathrm{g}A \ = \ \neg\mathrm{wp}\,(A,\ \mathit{false})$ or equivalently $\neg\mathrm{wlp}\,(A,\ \mathit{false}) \ \vee \ \neg\mathrm{t}A$. I.e. it is ensured that either the action terminates with some result or it never terminates.

2.1 Elementary and composite actions

In this paper we use capital letters such as X, Y, E, and F to denote a vector of variables and other components. We use subscripts when referring to the components in a vector.

We shall ignore typing of variables and expressions and assume that all expressions used in definitions and examples are defined appropriately.

Let $q[E/X]$ denote the textual substitution of free variables X with expressions E in a predicate q. Also, let p be a predicate. The definitions for assert and other elementary actions, also given in [4, 6], are

$$\begin{array}{lcl}
\mathrm{wlp}\,(\{p\},\ q) & \hat{=} & p \ \Rightarrow \ q \\
\mathrm{wlp}\,(\mathbf{abort},\ q) & \hat{=} & \mathit{true} \\
\mathrm{wlp}\,(\mathbf{skip},\ q) & \hat{=} & q \\
\mathrm{wlp}\,(X := E,\ q) & \hat{=} & q[E/X]
\end{array}$$

Note that *abort* never terminates, whereas an $\{\}$, **skip**, and an assignment always terminate.

Let, as before, A and B be actions, and p be a predicate. The sequential composition, the composition by a non-deterministic choice, as well as prefixing an action with a guard are defined as

$$\begin{array}{lcl}
\mathrm{wlp}\,(A; B,\ q) & \hat{=} & \mathrm{wlp}\,(A,\ \mathrm{wlp}\,(B,\ q)\,) \\
\mathrm{wlp}\,(A \ [\!]\ B,\ q) & \hat{=} & \mathrm{wlp}\,(A,\ q) \ \wedge \ \mathrm{wlp}\,(B,\ q) \\
\mathrm{wlp}\,(p \rightarrow A,\ q) & \hat{=} & p \ \Rightarrow \ \mathrm{wlp}\,(A,\ q)
\end{array}$$

Termination is defined using analogous formulas.

The iteration has the liberal semantics

$$\mathrm{wlp}\,(\ \mathbf{do} \quad A \ \ \mathbf{od},\ q) \quad \hat{=} \quad \forall\, n \geq 0 \ \cdot \ \mathrm{wlp}\,(A^n,\ \mathrm{g}A \ \vee \ q)$$

where $A^{n+1} \ \hat{=} \ A; A^n$ with $A^0 \ \hat{=} \ \mathbf{skip}$.

For the iteration the termination is $\mathrm{t}(\ \mathbf{do} \quad A \ \ \mathbf{od}) = \exists\, n \geq 0 \ \cdot \ \neg\,\mathrm{g}A^n$.

2.2 Differential Action

The differential action [9] causes given variables X to evolve according to a system of differential equations as long as a guard e holds. This guard condition may use the variables X. We use the standard convention of writing a system of differential equations as $\dot{X} \ = \ F(X)$, where the $\dot{X}$ is a component-wise first derivative of X.

Definition 1. Let e be a guard and q be a postcondition, both may speak of the variables X. The differential action $e: \dot{X} = F(X)$ has the semantics

$$\begin{array}{l} \mathrm{wlp}\,(e: \ \dot{X} = F(X),\ q) \ \widehat{=} \\ \quad \exists\, \Phi \ \cdot \ \Phi(0) = X \ \wedge \ \dot{\Phi} = F(\Phi) \ \wedge \\ \quad\quad (\forall\, \tau \geq 0 \ \cdot \ (e \ \vee \ q)[\Phi(\tau)/X] \ \vee \ \exists\, 0 \leq \delta < \tau \ \cdot \ \neg e[\Phi(\delta)/X]) \end{array}$$

In the semantics, the first two conjuncts require the existence of the solution Φ for the differential equation $\dot{X} = F(X)$. The third conjunct isolates the first point in time, where the guard ceases to hold.

Termination is defined exactly by the existence of such an instant:

$$\begin{array}{l} \mathrm{t}\ e: \ \dot{X} = F(X) \ \widehat{=} \\ \quad \exists\, \Phi \ \cdot \ \Phi(0) = X \ \wedge \ \dot{\Phi} = F(\Phi) \ \wedge \\ \quad\ \exists\, 0 \leq \delta \ \cdot \ \neg e[\Phi(\delta)/X] \end{array}$$

Note that if the action terminates at δ then $q[\Phi(\delta)/X]$ holds.

An example of travelling train. Modelling of a system with different kinds of continuous behaviour requires use of several differential actions. An example of such a system is a travelling train, which accelerates to travelling velocity and travels with that velocity up to a deceleration point, from where on the train decelerates to a full stop.

Let x be the position of the train and v its velocity. The phase, where the train accelerates to travelling velocity tv, is modelled as a differential action

$$v < tv: \ \dot{x}, \dot{v} = v, 1$$

The next phase, where the train travels with constant velocity up to a deceleration point d, is modelled as

$$x < d: \ \dot{x} = v$$

The last phase, where the train decelerates to a full stop, is modelled as

$$v > 0: \ \dot{x}, \dot{v} = v, -1$$

In this example, the phases take place consecutively. In order to model the entire system, we could combine these three actions sequentially. However, this requires that we know, what the sequence is. Another, more general way, is to iterate the actions and let the actions themselves take the right sequence. In the case of a travelling train, we would have

$$\begin{array}{lllll} \mathbf{do} & x < d \ \wedge \ v < tv & \rightarrow & v < tv: \ \dot{x}, \dot{v} = v, 1 & \mathbf{od} \\ [\!] & x < d \ \wedge \ v = tv & \rightarrow & x < d: \ \dot{x} = v & \\ [\!] & x \geq d \ \wedge \ v > 0 & \rightarrow & v > 0: \ \dot{x}, \dot{v} = v, -1 & \end{array}$$

This interleaving of continuous evolutions *switches* x between different controlling differential equations. Another typical hybrid phenomenon is *jumping*, where a continuous evolution is followed by a discrete change. We could for instance add a "restart" action to the program above:

$$x > d \ \wedge \ v = 0 \quad \rightarrow \quad x := 0$$

At the end of the journey, the train jumps back to start. Our objective in the following is to investigate a standard form for expressing such phenomena with iteration of differential and ordinary actions.

3 Hybrid Alternation

There are several ways to model hybrid alternation, but not all of them are equally suitable for modelling the alternation in a hybrid system. We investigate different forms of alternation, where time is assumed to progress during differential actions, but not during any other actions.

The alternations are studied in a standard form $\mathbf{do} \quad A_1 \ [\!] \ A_2 \ [\!] \ \ldots A_n \ \mathbf{od}$ where all the actions $A_1, \ldots A_n$ are free of non-deterministic composition. An iteration that is not in this form is translated to the standard form by atomicity refinement [1]. For example, an iteration

$$\begin{array}{lllll} \mathbf{do} & p & \rightarrow & (g \rightarrow A_1 \ \ [\!] \ \ \neg g \rightarrow A_2) \\ [\!] & \neg p & \rightarrow & A_3 \\ \mathbf{od} & & & \end{array}$$

where p and g are predicates and A_1, A_2, and A_3 are actions without non-deterministic choice, is translated to the standard form by propagating the guard p in the first action yielding

$$\begin{array}{llll} \mathbf{do} & p \ \wedge \ g & \rightarrow & A_1 \ \mathbf{od} \\ [\!] & p \ \wedge \ \neg g & \rightarrow & A_2 \\ [\!] & \neg p & \rightarrow & A_3 \end{array}$$

A differential action describes how the variables evolve when the guard is enabled. It also states that when the guard does not hold, the differential action has no effect on the variables. This is called stuttering. When thinking of how a hybrid system behaves, we are mainly interested in the stutter free behaviour. An infinite stuttering is seen as termination. Therefore, we introduce a notational short form for a stutter free differential action, which is $e :\rightarrow \dot{X} = F(X)$ with the meaning of $e \rightarrow e : \ \dot{X} = F(X)$. The stutter free differential action has the property that it is enabled only when the evolution guard holds, that is $\mathrm{g}(e :\rightarrow \dot{X} = F(X)) \quad = \quad e$.

3.1 Strict alternation

In strict alternation continuous evolution is given a chance for execution after every discrete change.

Assuming that the discrete changes D are modelled by a non-deterministic composition of ordinary actions, the continuous evolutions DA are modelled by a non-deterministic composition of stutter free differential actions, and that two auxiliary boolean variables *disc* and *cont* are available, the strict alternation takes the standard form

$$\begin{array}{lllll} \{disc \ \wedge \ \neg cont\}; & & & & \\ \ \ \mathbf{do} & disc \ \wedge \ \mathrm{g}D & \rightarrow & D; \ disc, cont := false, true \\ \ \ [\!] & cont \ \wedge \ \mathrm{g}DA & \rightarrow & DA; \ disc, cont := true, false \\ \ \ \mathbf{od} & & & \end{array}$$

The assertion in front of the iteration ensures that the discrete change may take place before the continuous evolution.

The strict alternation has a major drawback, it cannot model switching. Between continuous evolutions there must be a discrete change, otherwise the system terminates. Moreover, the strict alternation allows continuous evolution only if some discrete change has occurred. However, if no continuous evolution is enabled, the discrete changes may takes place successively.

The effect of the continuous evolution may be observed only at that point of time, when the corresponding differential action terminates. Also, the number of observing ordinary actions is limited; there can be only one observation by one enabled ordinary action. This means that, if two actions were enabled, they would be competing for the execution, even if they do not disable each other, because the continuous evolution taking place immediately after the first discrete change could alter the state space in such a way that the other ordinary action would become disabled.

The strict alternation terminates, when the system cannot jump anymore, $(\neg disc \ \vee \ \neg\,\mathrm{g}D) \ \wedge \ (\neg cont \ \vee \ \neg\,\mathrm{g}DA)$.

3.2 Prioritised strict alternation

In this form of alternation the continuous evolution is given a chance for execution after all discrete changes have taken place.

The prioritised strict alternation has the standard form

$$\begin{array}{llll} \{\neg cont\}; & & & \\ \ \ \mathbf{do} & \mathrm{g}D & \rightarrow & D; \ cont := true \\ \ \ [\!] & cont \ \wedge \ \neg\,\mathrm{g}D \ \wedge \ \mathrm{g}DA & \rightarrow & DA; \ cont := false \\ \ \ \mathbf{od} & & & \end{array}$$

Like in strict alternation, the effect of the continuous evolution may be observed only after the execution of the corresponding differential action. However, the prioritised strict alternation allows several observations at the same time. Any enabled actions are executed before a new continuous evolution takes place, provided that the actions do not disable each other. Therefore, this model has the potential to execute discrete changes forever.

Since the discrete changes are assumed to be timeless, the infinite execution would correspond to time holding still. This, in turn, is interpreted as aborting.

The termination of prioritised strict alternation is slightly more complicated than in the strict alternation. The prioritised strict alternation terminates, when no discrete change is enabled and some continuous evolution has already been executed, that is $\neg\,\mathrm{g}D \,\wedge\, (\neg \mathit{cont} \,\vee\, \neg\,\mathrm{g}DA)$. This underlines the drawback of this model; in cannot model switching. Moreover, the alternation must start with a discrete change.

3.3 Prioritised alternation

In prioritised alternation the discrete changes have a priority over continuous evolutions. However, the discrete changes may take place only when no continuous evolution is being executed.

The prioritised alternation has the standard form

$$
\begin{array}{lllll}
\mathbf{do} & \mathrm{g}D & \rightarrow & D \\
[\!] & \neg\,\mathrm{g}D & \rightarrow & DA \\
\mathbf{od} & & &
\end{array}
$$

Like in the prioritised strict alternation, the effect of the continuous evolution may be observed by several discrete actions after the execution of the differential action. The prioritised alternation has also the potential for the discrete changes to abort.

In comparison to the prioritised strict alternation the continuous evolutions are more independent from the discrete changes. The alternation may start with a continuous evolution, even if there are no enabled discrete changes, and the continuous evolutions may appear successively. This means that the prioritised alternation can model both switching and jumping in hybrid systems.

The prioritised alternation terminates, when no more jumping or switching occurs, that is both discrete changes and continuous evolutions are disabled, $\neg\,\mathrm{g}D \,\wedge\, \neg\,\mathrm{g}DA$. This is seen by transforming the iteration to the normal form. The action $\neg\,\mathrm{g}D \rightarrow DA$ has the same semantics as the action $\neg\,\mathrm{g}D \,\wedge\, \mathrm{g}DA \rightarrow DA$.

3.4 Interrupting prioritised alternation

In interrupting prioritised alternation, the discrete changes have unconditional priority over continuous evolutions. In other words, the discrete changes occur whenever enabled, even when the continuous evolution modelled by a differential action has not terminated.

Let DI be a non-deterministic composition of stutter free differential actions of form $\neg\,\mathrm{g}D \,\wedge\, e :\rightarrow \dot{X} = F(X)$. The prioritised alternation has the standard form

$$\begin{array}{lllll} \textbf{do} & \mathrm{g}D & \rightarrow & D \\ [\!] & \neg\,\mathrm{g}D & \rightarrow & DI \\ \textbf{od} \end{array}$$

Unlike in the previous forms of alternations, the effect of the continuous evolution is observable at points of time defined by the discrete action guards. An enabled discrete action is executed immediately when it becomes enabled. In order to ensure this, the states, in which the discrete actions are enabled, are excluded by the evolution guard. The discrete changes may execute forever causing the alternation to abort.

Like the prioritised alternation, the interrupting prioritised alternation can model both switching and jumping.

The interrupting prioritised alternation terminates, when both discrete changes and continuous evolutions are disabled, that is $\neg\,\mathrm{g}D \wedge \neg\,\mathrm{g}DI$. This is seen by transforming the iteration to the normal form.

3.5 The chosen form of alternation

The prioritised alternation is chosen as the form of hybrid alternation to be used in hybrid action systems. The prioritised alternation has the simplest standard form and it is not as limited as the strict alternation and the prioritised strict alternation. The interrupting prioritised alternation gives too much priority to discrete actions, it weakens the evolution guard causing the continuous evolution to vanish sometimes. This phenomena is illustrated by the following example:

Consider an adaptive bottle filler, where the filling is stopped depending on how full the previous bottle was. Let x be the liquid level in the current bottle, p be the liquid level in the previous bottle, and $f(x,p)$ be the decision function to stop the filling. The adaptive bottle filling is captured by a differential action $f(x,p) :\ \dot{x} = 1$. We know that when the bottle is empty, the value of x is zero. However, because we do not know how full the bottle will be, we would like to model the change of bottle as an action $x > 0 \rightarrow p, x := x, 0$, which is to be executed right after the bottle has been filled. If we use the interrupting prioritised alternation to combine these actions we get

$$\begin{array}{llll} \textbf{do} & x > 0 & \rightarrow & p, x := x, 0 \\ [\!] & x \leq 0 \ \wedge\ f(x,p) & \rightarrow & x \leq 0 \ \wedge\ f(x,p) :\ \dot{x} = 1 \\ \textbf{od} \end{array}$$

which would hardly fill any bottle, because $x \leq 0 \ \wedge\ f(x,p)$ becomes false when there is something in the bottle. The same problem does not appear in the prioritised alternation (End of example).

For the prioritised alternation we give the short form

$$\textbf{alt}\ D\ \textbf{with}\ DA \qquad (1)$$

with the meaning $\textbf{do}\ \ \mathrm{g}D \rightarrow D\ [\!]\ \neg\,\mathrm{g}D \rightarrow DA\ \ \textbf{od}$. This form of alternation is used for identifying hybrid action system that are used in modelling and analysis of hybrid systems.

4 Jumps and Switches in the Unified model

Branicky gives the dynamics for the unified model with the help of jump sets in [5]. Here jump means a jump from one state of the system to another. Since in hybrid action systems the values of the variables define the state of the system, a jump set may model both switching and jumping in the system. The used jump sets are autonomous jump sets A, controlled jump sets C, and corresponding jump destination sets T. Informally, the system evolves continuously until it enters either A or C. If the state is in A the system must make a jump into a destination state T according to autonomous jump transition map G, which relates states in A to states in T with the current control input. If the entered state is in C the system may choose to change the state into a state in T. An illustration of the dynamics is shown in Figure 1. In the unified model there are also jump delay maps, which express the time delay associated to each jump in the system.

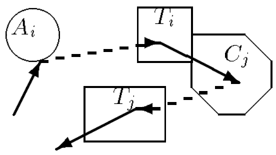

Figure 1: Example dynamics of the unified model. The dashed lines represent jumps and the solid lines continuous evolutions.

Next we describe the behaviour of a hybrid action system in terms of the jump sets. These we obtain from the hybrid alternation (1) that is present in a hybrid action system.

Autonomous jump sets. Let the continuous behaviour in a hybrid alternation be a non-deterministic composition of a general form

$$\begin{array}{l} \quad e_1 \ \wedge \ g_1 :\rightarrow \dot{X} = F_1(X) \\ [\!] \ \ldots \\ [\!] \ e_n \ \wedge \ g_n :\rightarrow \dot{X} = F_n(X) \end{array}$$

where for all $i = 1..n$ the e_i speaks only of evolving variables X and g_i is free of X. A continuous evolution in the system (solid arrow in Figure 1) is modelled by a differential action in the composition above. For each differential action $i = 1..n$, the autonomous jump set A_i is defined by the predicate $\neg e_i \ \wedge \ g_i$, which specifies all those states, to which the corresponding evolution may lead. Together all the individual autonomous jump sets form the autonomous jump sets A of the system.

This covers the case of finite collections of jump sets. For countable infinite jump sets, one might encode the current index as a program variable ci and augment the guards by the condition $ci = i$, and similarly replace F_i by a function dependent on ci. The result would be one remaining differential action.

Jump transition map and jump destination sets. Let the discrete changes in the hybrid alternation be described by an action $\mathrm{g}D \rightarrow D$, where D may be a non-deterministic composition of actions. A jump transition map G_i for an autonomous jump set is computed by $\{A_i\};$ **do** $\mathrm{g}D \rightarrow D$ **od**. Thus, the corresponding jump destination set is defined by

$$\exists\, X_0 \;\cdot\; \mathrm{wlp}\,(\{A_i\};\;\; \mathbf{do}\;\; \mathrm{g}D \rightarrow D\;\; \mathbf{od},\; X = X_0)$$

The jump transition maps and the jump destination sets for individual autonomous jump sets form together the jump transition map G and the jump destination sets T of the system.

Controlled jump sets. A hybrid action system does not exhibit explicit controlled jump sets, where the evolution may change anywhere within the jump set. However, we can identify restricted controlled jump sets, where the evolution may change at some predetermined subregions.

Let the following non-deterministic composition be a part of the continuous behaviour in a hybrid alternation.

$$\begin{array}{ll} & e_1 :\rightarrow \dot{X} = F(X) \\ [\!] & \ldots \\ [\!] & e_n :\rightarrow \dot{X} = F(X) \end{array}$$

This describes uniform behaviour in the region e which is a disjunction of the evolution guards $e_1 \;\vee\; \ldots \;\vee\; e_n$. If there exists such an order for the differential actions in the non-deterministic composition that

$$\forall\, i \;\cdot\; \exists\, j \;\cdot\; \mathrm{wlp}\,(e_j :\rightarrow \dot{X} = F(X),\; e_i) = true \qquad \text{where} \;\; i = 2..n \;\; \text{and} \;\; j = 1..n,$$

the region e is path connected. In that case the differential actions describe the paths that connect the subregions. Such a path connected region e is a controlled jump set. The subregions, where the evolution may change are specified by the negations of the individual evolution guards $\neg e_1, \ldots, \neg e_n$.

The separate controlled jump sets form together the controlled jump sets C of the system. The corresponding jump destination sets are included in the jump destination sets T.

Transition delay maps. There are no transition delay maps attached to the hybrid alternation. This is due to the approach that the time is implicitly present only during the continuous evolution modelled by the differential actions. If we wish to add the notion of delay to the jumps taking place in the system, we may model the delay explicitly with an extra variable and a differential action that evolves it.

4.1 About the other alternation models

The relation between the other hybrid alternation models in Section 3 and the Branicky's unified model is more restricted than the relation of the chosen form of hybrid alternation.

As mentioned earlier, the problem with the strict alternation and with the prioritised strict alternation is that they do not allow switching. This excludes the controlled jump sets. Furthermore, these models require that the system starts by a discrete change, which is a limitation not present in the unified model.

The interrupting prioritised alternation makes the evolution guards interruptible by the discrete changes. Hence, every evolution is taken within a controlled jump set specified by the evolution guard. This is a special case in the unified model, where the dynamics is always covered by a controlled jump set.

4.2 A Hysteresis Example

We illustrate the relation between hybrid action systems and Branicky's unified model with a hysteresis example from [5]. In this example a room is controlled by a thermostat. The difference from desired room temperature is denoted d. The desire is to keep a room temperature such that $-l \leq d \leq l$ by either heating the room, $h = 1$, or cooling the room, $h = -1$, where h is used as a control variable. The heating is kept on as long as $d < l$, otherwise the thermostat changes to cooling. Similarly, the cooling is kept on as long as $-l < d$, otherwise the thermostat changes to heating. The heating is on initially. The Figure 2 illustrates the control policy.

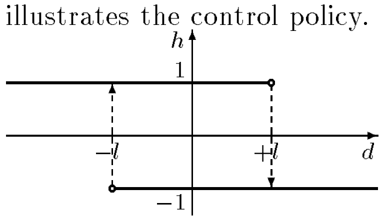

Figure 2: The control policy for the thermostat

The hybrid alternation modelling the behaviour of the thermostat has four actions. The two ordinary actions change from heating to cooling and from cooling back to heating, and the two differential actions model the continuous evolution during the heating and the cooling. Let $f(d, h)$ model the changes in the difference of the room temperature from the desired temperature. The hybrid alternation for the thermostat is

$$\begin{array}{llll}
\textbf{alt} & h = 1 \ \wedge \ l \leq d & \rightarrow h := -1 & \\
[\!] & h = -1 \ \wedge \ d \leq -l & \rightarrow h := 1 & \\
\textbf{with} & d < l \ \wedge \ h = 1 & :\rightarrow \dot{d} = f(d,h) & (1) \\
[\!] & -l < d \ \wedge \ h = -1 & :\rightarrow \dot{d} = f(d,h) & (2)
\end{array}$$

For the differential actions in this hybrid alternation we obtain the autonomous jump sets by taking the partial negation of the evolution guards, yielding

$$\begin{array}{lcl} A_1 & = & d \geq l \ \wedge \ h = 1 \\ A_2 & = & -l \geq d \ \wedge \ h = -1 \end{array}$$

For these jump sets we get the associated jump destination sets by computing the result of iterating the ordinary actions, yielding

$$\begin{array}{l} T_1 = d \geq l \ \wedge \ h = -1 \\ T_2 = -l \geq d \ \wedge \ h = 1 \end{array}$$

The connection between the autonomous jump sets and the jump destination sets is that when the evolution enters A_1 the system jumps autonomously to T_1. Similarly, the system jumps autonomously from A_2 to T_2. The same jump set characterisation was obtained for this example in [5].

5 Conclusion

In this paper we studied different models of hybrid alternation and how the priority in these models limit the hybrid phenomena that they can exhibit. From these models the prioritised alternation is proposed as the standard form, because it is the most general model.

We also compared the chosen form of alternation with the Branicky's unified model. We showed that the hybrid alternation captures the autonomous jump sets, predetermined controlled jump sets, and the jump destination sets in the unified model. The jump transition map is computed by the discrete actions and the transition delay maps are modelled with differential actions. Thus, we have shown indirectly, how to translate a system with predetermined controlled jump sets in the unified model to a hybrid alternation and vice versa. Branicky has given in [5] the link between hybrid automata and the unified model. Therefore, we can use tools designed for analysis of hybrid automata, like HyTech, for analysing hybrid alternation as well.

Since the semantics for the actions are given with predicate transformers, we can also use theorem provers, like PVS, for verifying properties in the hybrid alternation. The theorem provers are efficient especially in proving invariant properties, which makes this an attractive subject for future work.

Acknowledgements. The first author would like to thank the participants at the 9th Nordic Workshop on Programming Theory in Tallinn for their comments and suggestions.

References

[1] R. J. R. Back. *Atomicity Refinement in a Refinement Calculus FrameWork.* Reports on computer science and mathematics 141, Åbo Akademi, 1993.

[2] R. J. R. Back and R. Kurki-Suonio. Decentralization of process nets with centralized control. In *Proc. of the 2nd ACM SIGACT–SIGOPS Symp. on Principles of Distributed Computing*, pp. 131–142, 1983.

[3] R. J. R. Back and K. Sere. Stepwise refinement of action systems. *Structured Programming*,**12**, pp. 17-30, 1991.

[4] M. Bonsangue and J. N. Kok. Semantics, orderings and recursion in the weakest precondition calculus. In J. W. de Bakker, G. Rozenberg, W. P. de Roever (eds.) *Proceedings Rex Workshop on Semantics: Foundations and Applications*, LNCS 666, pp. 91-110, Springer-Verlag, 1993.

[5] M. S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control.* PhD Thesis. Massachusetts Institute of Technology, EECS Dept, 1995.

[6] M. Broy and G. Nelson. Adding Fair Choice to Dijkstra's Calculus. *ACM Transactions on Prog. Languages and Systems*, **16**,3, pp. 924-938, 1994.

[7] M. Butler, E. Sekerinski, and K. Sere. An Action System Approach to the Steam Boiler Problem. In J-R. Abrial, E. Borger, H. Langmaack (eds.) *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, LNCS 1165, Springer-Verlag, 1996.

[8] E. W. Dijkstra. *A Discipline of Programming.* Prentice–Hall International, 1976.

[9] M. Rönkkö and A. P. Ravn. *Differential Equations as Actions.* No. 109, Technical Reports, Turku Center for Computer Science, Åbo Akademi University, Finland, April 1997. Presented at *Hybrid Systems V*, Notre Dame, Indiana, USA, 1997.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**