Hybrid Action Systems

M. Rönkkö

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

A. P. Ravn

Department of Computer Science, Technical University of Denmark,
bldg. 344, DK 2800 Lyngby, Denmark,
apr@it.dtu.dk

Turku Centre for Computer Science
TUCS Technical Report No 110
May 1997

ISBN 951-650-999-1
ISSN 1239-1891

Abstract

Based on the differential action introduced in [12] we extend the action system framework into a hybrid action system framework. Hybrid action systems are to be used in modelling and analysing hybrid systems. The meaning of a hybrid action system is given as an ordinary action system. We also extended the definition of a parallel composition for action systems to handle the continuous behaviour in the hybrid action system framework.

Keywords: action systems, hybrid action systems, hybrid systems, parallel composition, linear composition, differential equations, weakest liberal precondition

1 Introduction

The continuous variables in hybrid systems make the use of interleaving model for parallel composition difficult. The interleaving model is based on the assumption that the effect of two simultaneous actions is the same as the effect of two sequential actions. This assumption is not valid in a continuously parallel model. In that model the linear additivity takes place, i.e. the effect of two simultaneous actions is twice as "big" as the effect of one action. A simple example of this is an action that disables itself after its execution. In the interleaving model there cannot be two sequential executions of this kind of actions, whereas in the continuously parallel model, there can very well be. Another problem arises, if the actions are allocated time for their performance. In this case the interleaving model returns wrong results concerning the time consumption of two simultaneous actions.

However, the interleaving model is very effective in its own domain, none time consuming discrete computations. Hence, this model suits well for handling the discrete variables in the hybrid systems.

Previously, action systems have been extended with explicit clock variables when reasoning about real-time systems [9, 10]. The discrete approximations have been used in modelling continuous phenomena in hybrid settings. However, the differential action introduced in [12] allows us to use the differential equations directly in capturing the continuous phenomena. This reduces the need for modelling the ticks of the clocks explicitly, which in turn simplifies the reasoning of real-time and hybrid systems.

The approach taken in this paper is to combine both continuous parallelism and interleaving parallelism in the same framework, called *hybrid action systems*. Hybrid action systems keep the continuous model separate from the discrete model. This allows a meaningful parallel composition for both of the models. If one wants to analyse particularly the hybrid behaviour of a hybrid action system, one has to first translate it into an ordinary action system. This translation is mechanical and it is based on a very general computational model, an abstraction from the Branicky's unified model [5]. An additional advantage of our approach is that it does not impose any further constraints. Therefore, all the existing results for action systems are also applicable.

Overview

In Section 2, we give a quick introduction to action systems. In Section 3, we use the linearity property to show that a linear composition of sets of partially defined systems of differential equations is a single set of partially defined systems of differential equations. This result is then used in Section 4 for showing that a linear composition of differential actions returns a set of differential actions, from which the applied differential action is chosen nondeterministically. In Section 5, we have a look at a general computational model for the hybrid systems, which is used in Section 6 for defining a hybrid action system. The meaning of the hybrid behaviour is given as an ordinary action system. In Section 7, we extend the parallel composition to handle the hybrid action systems. Finally, in Section 8 we have a look at various examples, where the goal is to obtain a single action system from many parallelly composed hybrid action systems.

2 Action Systems

In this section we define the actions with the weakest liberal precondition predicate transformer. We also define an ordinary action system and give the meaning of the parallel composition for several action systems.

2.1 Actions

Action systems, originally proposed by Back and Kurki-Suonio [2], are predicate transformer based systems. Action systems have been applied in modelling reactive and concurrent behaviour, see e.g.[3, 7]. Recently, the action systems framework has also been introduced for reasoning about discrete control systems [11].

An *action* in action systems is any statement in Dijkstra's guarded command language [8] including the pure guarded commands. The actions are defined with the *weakest precondition* predicate transformer $\mathrm{wp}(A,\ q)$ that returns the weakest precondition, from which the action A terminates and establishes the postcondition q. The *termination* of an action is also given by the weakest precondition as

$$\mathrm{t}A \quad = \quad \mathrm{wp}(A,\ true)$$

An action is said to be enabled, when its *guard condition* is not $false$. The guard condition is given by

$$\mathrm{g}A \quad = \quad \neg\mathrm{wp}(A,\ false)$$

The semantics for the actions can also be defined with the *weakest liberal precondition* (wlp). This is chosen here, because in reactive systems nontermination is often considered as correct behaviour. The relation between the weakest liberal precondition and the weakest precondition is given by Dijkstra [8] as

$$\mathrm{wp}(A,\ q) \quad = \quad \mathrm{wlp}(A,\ q) \quad \wedge\ \mathrm{t}A$$

Let the $q[^e/_x]$ denote the textual substitution of the free occurrencies of variables x with the expressions e in the predicate q. Furthermore, let A and B be actions, and p be a predicate. The elementary actions are defined as [4, 6]

$$\begin{array}{lcl}
\mathrm{wlp}(abort,\ q) & \widehat{=} & true \\
\mathrm{wlp}(skip,\ q) & \widehat{=} & q \\
\mathrm{wlp}(x := e,\ q) & \widehat{=} & q[^e/_x]
\end{array}$$

The semantics for composing actions sequentially, by non-deterministic choice, and by prefixing guard is defined as [4, 6]

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

The iteration has the semantics

$$\mathrm{wlp}(\ \mathbf{do}\, A\ \mathbf{od},\ q) \quad \widehat{=} \quad \forall_{n \geq 0} \,.\, \mathrm{wlp}(A^n,\ \mathrm{g}A\ \vee\ q)$$

for which the termination is given by $\mathrm{t}(\ \mathbf{do}\, A\ \mathbf{od}) = \exists_{n \geq 0}\ .\ \neg \mathrm{g}A^n$

The differential action evolves a variable x continuously as long as a boundary expression e holds. The continuous evolution is determined by a differential equation $\dot{x} = f(x)$, for which the existence of the solution is required. The semantics of the differential action is given as [12]

$$\begin{array}{lcl}
\mathrm{wlp}(\dot{x} = f(x)\ \mathbf{inv}\ e,\ q) & \widehat{=} & \exists_{\Phi}\ .\ \Phi(0) = x\ \wedge\ \dot{\Phi} = f(\Phi)\ \wedge \\
& & (\forall_{\tau \geq 0}\ .\ (e\ \vee\ q)[^{\Phi(\tau)}/_x]\ \vee \\
& & \quad \exists_{0 \leq \delta < \tau}\ .\ \neg e[^{\Phi(\delta)}/_x])
\end{array}$$

For the derivation of the semantics for the differential action, see also [12].

2.2 Action systems

An action system $\mathcal{A}$ is an initialized block of form

$$\mathcal{A} \ = \ |[\ \mathbf{var}\ \ x\ ;\ I;\ \ \mathbf{do}\ \ A_1 \,[\!]\ \ \ldots\ [\!]\ \ A_m\ \mathbf{od}]|:z$$

where x are the *local* variables and *import list* z includes all the *global* variables that are referred in the action system. The local variables are visible only within that action system where they are defined. A globally visible variable is denoted with a star (e.g. i^*) at the declaration time. The global variables are available to all action systems, but they must be made explicitly visible by including them into the import list z. Therefore, any of those global variables that are not refered in the z are also not visible to the action system $\mathcal{A}$. The local and the visible global variables form the *state variables* of $\mathcal{A}$. All of these variables are distinct and associated with an explicit type.

All the local variables are initialized by the action I. Because no preference is given to any of the variables, the initialization may not refer to any uninitialized variable. For example, an initialization $a, b := b+3, a+4$, where $\{a, b\} \in x$, would be undefined, since the values of the variables themselves are undefined before the initialization. Executionally, all the action systems are assumed to be synchronized within the initialization step. This becomes apparent later, when the parallel composition for action system is defined.

After the initialization, the actions A_1 ... A_m are repeatedly executed when enabled. The selection of an enabled action for execution is non-deterministic. There are no assumptions about fairness in the selection of actions. The execution of an action is always atomic, which ensures that during the execution the values of the variables remain the same unless the action itself changes them. If two enabled actions refer to disjoint variables, their execution can be in any order or in parallel. Hence, this models parallelism by interleaving.

The repeated execution of the actions A_1 ... A_m terminates when none of them is enabled, i.e. $\neg \mathrm{g}(A_1 \parallel \ \dots \ \parallel \ A_m)$. Similarly, the repeated execution of the actions aborts when an executed action aborts.

2.3 Parallel Composition

Consider two action systems $\mathcal{A}$ and $\mathcal{B}$

$$\begin{array}{lcl}\mathcal{A} & \hat{=} & |[\ \mathbf{var}\ \ x\ ;\ I;\ \ \mathbf{do}\ \ A_1 \parallel \ \dots \ \parallel \ A_m\ \mathbf{od}]| : z \\ \mathcal{B} & \hat{=} & |[\ \mathbf{var}\ \ y\ ;\ J;\ \ \mathbf{do}\ \ B_1 \parallel \ \dots \ \parallel \ B_n\ \mathbf{od}]| : w\end{array}$$

where x is disjoint from y. The *parallel composition* $\mathcal{A} \parallel \mathcal{B}$ is defined as an action system

$$\begin{array}{lcl}\mathcal{A} \parallel \mathcal{B} & = & |[\ \mathbf{var}\ \ x, y\ ; \\ & & \quad I; J; \\ & & \quad \mathbf{do}\ \ A_1 \parallel \ \dots \ \parallel \ A_m \parallel \ B_1 \parallel \ \dots \ \parallel \ B_n\ \mathbf{od} \\ & &]| : (z \cup w) - (x \cup y)\end{array}$$

The parallel composition combines the state spaces of the two action systems, merging the global variables and keeping the local variables distinct.

Those imported global variables in $z \cup w$, that are defined in either of the actions system, i.e. $(x \cup y)$, are removed from the import list. In this sense, these global variables are associated with a unique meaning, and thus become local.

The behaviour of the parallel composition depends on how the individual action systems, the *reactive components*, interact with each other via the global variables that they reference. For instance, a reactive component does not terminate by itself. The termination is a global property of the composed action system [1].

Since the initialization actions are well defined, and non-deterministic choice between actions is both associative and commutative, the parallel composition is also associative and commutative. Therefore, the meaning of several parallelly composed action systems can be unfolded in any order without affecting the result.

3 Systems of Differential Equations

In this section we start by clarifying, how a linear composition of partial functions is obtained. We then have a look at the notation of a partially defined system of differential equations. The linear composition is applied to show that linearily composed partially defined systems of differential equations is actually a set of partially defined systems of differential equations. This result is generalized into linear composition of sets of partially defined systems of differential equations.

3.1 Linear composition

Let X denote a vector of variables $(x_1, \ldots, x_n)$, and $\mathbf{d}.X$ denote the domain of X . Let $f(X)$ and $g(X)$ be partial functions. The linear composition of these functions is a set of partial functions

$$f(X) + g(X) = \begin{cases} f(X) + g(X), & X \in \mathbf{d}.f \cap \mathbf{d}.g \\ f(X), & X \in \mathbf{d}.f \cap \overline{\mathbf{d}.g} \\ g(X), & X \in \overline{\mathbf{d}.f} \cap \mathbf{d}.g \end{cases} \tag{1}$$

The linear composition is both associative and commutative. It differs from the ordinary summation in the sense, that it is defined in the whole range of $\mathbf{d}.f \cup \mathbf{d}.g$, whereas the summation is defined only in the range $\mathbf{d}.f \cap \mathbf{d}.g$.

From here on, all the functions are assumed to be piecewise continuous partial functions. The total functions can be seen as partial functions, for which the domain is total.

3.2 Partially defined system of DE

Let $F(X)$ be a vector of functions, which all have the same domain denoted as $\mathbf{d}.f$, that is

$$F(X) = \begin{bmatrix} f_1(X) \\ \vdots \\ f_n(X) \end{bmatrix}, \quad X \in \mathbf{d}.f$$

Furthermore, let $\dot{X}$ denote the first derivate of X, that is $(\dot{x}_1, \ldots, \dot{x}_n)$. Now, a partially defined system of differential equations, in short a *system*, is written as

$$\dot{X} = F(X)$$

Note, that the number of the functions in the vector F equals to the number of the variables in the vector X.

3.3 Linear composition of systems

Similarly to $F(X)$, let $G(X)$ be a vector of functions defined in the domain $\mathbf{d}.g$. Now, a linearily composed system of both $F(X)$ and $G(X)$ is written as

$$\dot{X} = F(X) + G(X) \tag{2}$$

Based on the linear composition of functions (1), the system (2) is actually a set of systems written as

$$\begin{cases} \dot{X} = F(X) + G(X), & X \in \mathbf{d}.f \cap \mathbf{d}.g \\ \dot{X} = F(X), & X \in \mathbf{d}.f \cap \overline{\mathbf{d}.g} \\ \dot{X} = G(X), & X \in \overline{\mathbf{d}.f} \cap \mathbf{d}.g \end{cases}$$

The linear composition can also be generalized into sets of systems. Let $\mathcal{F}(X)$, and $\mathcal{G}(X)$ be sets of vectors.

$$\mathcal{F}(X) = \begin{cases} F_1(X), & X \in \mathbf{d}.f_1 \\ \vdots \\ F_i(X), & X \in \mathbf{d}.f_i \end{cases} \qquad \mathcal{G}(X) = \begin{cases} G_1(X), & X \in \mathbf{d}.g_1 \\ \vdots \\ G_i(X), & X \in \mathbf{d}.g_j \end{cases}$$

A linearily composed system of $\mathcal{F}(X)$ and $G(X)$ is again a set of systems written as

$$\begin{cases} \dot{X} &= \mathcal{F}(X) + \mathcal{G}(X), \quad X \in \bigcup_{k=1}^{i} \mathbf{d}.f_k \cap \bigcup_{l=1}^{j} \mathbf{d}.g_l \\ \dot{X} &= \mathcal{F}(X), \quad X \in \bigcup_{k=1}^{i} \mathbf{d}.f_k \cap \bigcap_{l=1}^{j} \overline{\mathbf{d}.g_l} \\ \dot{X} &= \mathcal{G}(X), \quad X \in \bigcap_{k=1}^{i} \overline{\mathbf{d}.f_k} \cap \bigcup_{l=1}^{j} \mathbf{d}.g_l \end{cases}$$

4 Linear composition of differential actions

A differential action $\dot{X} = F$ **inv** E denotes a partially defined system in the domain E. A set of systems is denoted as a non-deterministic choice of differential actions

$$\begin{array}{ll} & \dot{X} = F_1 \textbf{ inv } E_1 \\ [\!] & \dot{X} = F_2 \textbf{ inv } E_2 \\ [\!] & \vdots \\ [\!] & \dot{X} = F_i \textbf{ inv } E_i \end{array}$$

The variables z remaining constant, that is $\dot{z} = 0$, are not explicitly expressed in a differential action. Therefore, the linear composition of systems is also applicable in the differential action notation.

4.1 An Example

Let x, y, z by the variables in X. An example of a linear composition of two systems in the differential action notation is following.

$$\dot{x}, \dot{y} = f, g \textbf{ inv } e \quad + \quad \dot{y}, \dot{z} = h, u \textbf{ inv } i$$

$$= \begin{array}{llll} & \dot{x}, \dot{y}, \dot{z} = f, g + h, u & \textbf{inv} & e \wedge i \\ [\!] & \dot{x}, \dot{y} = f, g & \textbf{inv} & e \wedge \neg i \\ [\!] & \dot{y}, \dot{z} = h, u & \textbf{inv} & \neg e \wedge i \end{array}$$

The same is expressed in mathematical notation 4as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} f(X) \\ g(X) \\ 0 \end{bmatrix}, X \in e \ + \begin{bmatrix} 0 \\ h(X) \\ u(X) \end{bmatrix}, X \in i$$

$$= \begin{cases} \dot{X} \ = \begin{bmatrix} f(X) \\ g(X) + h(X) \\ u(X) \end{bmatrix}, & X \in e \cap i \\ \\ \dot{X} \ = \begin{bmatrix} f(X) \\ g(X) \\ 0 \end{bmatrix}, & X \in e \cap \overline{i} \\ \\ \dot{X} \ = \begin{bmatrix} 0 \\ h(X) \\ u(X) \end{bmatrix}, & X \in \overline{e} \cap i \end{cases}$$

5 Computational Model

Computationally, a hybrid system starts from some initial state and iterates between continuous evolution and discrete changes of the values of the variables in the system. Thus, a hybrid system is basically a set of partially defined piecewise continuous functions. This can also seen as an abstraction from the Branicky's unified model presented in [5].

Assuming that the initial state $Init$ is known, the cause for the discrete change D is known as gD, and the boundary for the continuous evolution is also known as E, the computational model for a hybrid system is represented as a sequence of actions

$$\begin{array}{lll} Init; & & \\ \mathbf{do} \ \ E \ \wedge \ \neg \mathrm{g}D & \rightarrow & \dot{X} = F \ \mathbf{inv} \ E \ \wedge \ \neg \mathrm{g}D \\ \ [\!] \ \ \mathrm{g}D & \rightarrow & D \\ \mathbf{od} & & \end{array}$$

In this model, the continuous evolution proceeds up to a point where the discrete change becomes enabled. After the discrete change, the continuous evolution continuous, provided that the boundary condition E still holds. This interaction is then repeated.

5.1 Extremes

If the discrete change in the model does not disable itself, $\mathrm{wp}(D, \neg \mathrm{g}D) = false$, the hybrid system possesses no continuous evolution nor a termination after the discrete change becomes enabled. In other words, the state of the system becomes *undefined*.

If from some point on the continuous evolution takes a direction, from which neither the boundary is reached, nor the condition for any discrete change, that is $\mathrm{wlp}(\dot{X} = F \ \mathbf{inv}\ E \wedge \neg \mathrm{g}D,\ false) = true$, the system never terminates, but the behaviour is bounded by the boundary expression excluding the discrete points, where the discrete change may occur. In this case, the system becomes *stable* with respect to the boundary expression.

If at some point the boundary for the system defined by the E is exceeded and no discrete change is enabled, the system *terminates*.

6 Hybrid Action System

We extend the action system framework given in section 2.2 into a hybrid action system framework, where the continuous evolution is separated from the discrete changes. The hybrid system framework is defined as

$$\begin{array}{lll} \mathcal{H} & \widehat{=} & |[\ \mathbf{var}\ \ x;\ \ Init; \\ & & \quad \mathbf{des}\ \ \dot{X} = F\ \mathbf{inv}\ E \\ & & \quad \mathbf{do}\ \ \mathrm{g}D\ \rightarrow\ D\ \mathbf{od} \\ & &]| : z \end{array}$$

The meaning of a hybrid action system follows from the computational model described earlier. The previously defined hybrid action system $\mathcal{H}$ has the meaning of an action system

$$\begin{array}{lll} \mathcal{H} & = & |[\ \mathbf{var}\ \ x;\ \ Init; \\ & & \quad \mathbf{do}\ \ E \wedge \neg \mathrm{g}D \ \rightarrow\ \dot{X} = F\ \mathbf{inv}\ E \wedge \neg \mathrm{g}D \\ & & \quad [\!]\ \ \mathrm{g}D \qquad\qquad \rightarrow\ D \\ & & \quad \mathbf{od} \\ & &]| : z \end{array}$$

where X includes both the local variables x and the global variables z.

7 Parallel Composition

Because the action systems model discrete systems, the interleaving model for parallelism is sufficient. However, with continuous variables the interleaving model is inadequate. Most fundamental reason is the fact that in any interval, there is an infinite number of sequential values. Therefore, an interval cannot be "stepped" through value by value with full coverage.

To overcome this problem, we need to extend the parallel operator so, that it composes linearily the continuous behaviour, but keeps the parallelism interleaving for the discrete behaviour. In this way, we can make use of the existing results for the action systems also in hybrid action systems.

Let $\mathcal{H}$ and $\mathcal{K}$ be two hybrid action systems of the form

$$
\begin{aligned}
\mathcal{H} \;\widehat{=}\; & |[\ \textbf{var}\ \ x;\ \ InitH; \\
& \quad \textbf{des}\ \ \dot{X} = F\ \textbf{inv}\ E \\
& \quad \textbf{do}\ \ \mathrm{g}H\ \ \rightarrow\ \ H\ \textbf{od} \\
&]| : z
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{K} \;\widehat{=}\; & |[\ \textbf{var}\ \ y;\ \ InitK; \\
& \quad \textbf{des}\ \ \dot{Y} = G\ \textbf{inv}\ I \\
& \quad \textbf{do}\ \ \mathrm{g}K\ \ \rightarrow\ \ K\ \textbf{od} \\
&]| : v
\end{aligned}
$$

where x and y are disjoint. The parallel composition of these hybrid action systems has the meaning

$$
\begin{aligned}
\mathcal{H} \parallel \mathcal{K} \;=\; & |[\ \textbf{var}\ \ x, y;\ \ InitH; InitK; \\
& \quad \textbf{des}\ \ \dot{X} = F\ \textbf{inv}\ E\ \ +\ \ \dot{Y} = G\ \textbf{inv}\ I \\
& \quad \textbf{do}\ \ \mathrm{g}H\ \ \rightarrow\ \ H \\
& \quad [\!]\ \ \ \mathrm{g}K\ \ \rightarrow\ \ K \\
& \quad \textbf{od} \\
&]| : z \cup v
\end{aligned}
$$

where the differential actions describing the continuous behaviour are linearily composed.

Since both interleaving composition and linear composition are associative and commutative, this extended parallel composition is also associative and commutative. This means that the composition of several hybrid action systems can be unfolded in any order without affecting the result. Moreover, this extension allows composing both action systems and hybrid action system parallelly.

8 Examples

In this section, we have a look at three examples. The first example, water tank, is a completely continuous system. The second example, train, includes both discrete and continuous behaviour. The third example, fillers, illustrates the effect of composing parallelly the same continuous component several times.

Each of these examples contain a stepwise transformation from several parallelly composed (hybrid) action systems into a single action system. It should be noted that any of the obtained action systems do not possess any complicated structures that would make the further analysis particularly difficult.

8.1 Water Tank

Water tank is a system, where a leaking container is being filled. Both leaking and filling are continuous processes within the physical limitations of the tank. In the following, we derive an action system model by using hybrid action systems to describe the different characteristics of the water tank system.

Tank

We define the tank as an action system containing the water level variable. The tank itself has no behaviour whatsoever.

$$\mathcal{T}ank \quad \widehat{=} \quad |[\ \mathbf{var}\quad y^* : I\!R; \quad y := min]| :$$

Filling

Next, we define the filling process for the tank as a hybrid action system. The tank is filled with a constant rate in as long as the water level remains under fixed maximum water level max.

$$\mathcal{F}ill \quad \widehat{=} \quad |[\ \mathbf{des}\quad \dot{y} = in\ \mathbf{inv}\ y < max\ ]| : y$$

Leaking

Similarly, the tank leaks with a constant rate out. This leaking occurs only, if the water level of the tank is higher than the minimum water level min.

$$\mathcal{L}eak \quad \widehat{=} \quad |[\ \mathbf{des}\quad \dot{y} = -out\ \mathbf{inv}\ min < y\ ]| : y$$

Water tank

The water tank is now a parallel composition of all these three characteristics.

$$\mathcal{W}Tank \quad \widehat{=} \quad \mathcal{T}ank \parallel \mathcal{F}ill \parallel \mathcal{L}eak$$

Obtaining an action system

We can unfold the meaning of the water tank into an action system. We start by unfolding the parallel compositions to obtain a single hybrid action system.

$$
\begin{array}{lcll}
\mathcal{W}Tank & = & |[& \textbf{var} \;\; y^* : I\!R; \;\; y := min; \\
& & & \textbf{des} \;\; \dot{y} = in \; \textbf{inv} \; y < max \;\; + \\
& & & \qquad \dot{y} = -out \; \textbf{inv} \; min < y \\
& &]| : &
\end{array}
$$

This hybrid action system can be rewritten by unfolding the linear composition of the systems of differential equations.

$$
\begin{array}{lcllll}
\mathcal{W}Tank & = & |[& \textbf{var} & y^* : I\!R; \;\; y := min; & \\
& & & \textbf{des} & \dot{y} = in - out \; \textbf{inv} & min < y < max \\
& & & \;[\!] & \dot{y} = in \; \textbf{inv} & y \leq min \\
& & & \;[\!] & \dot{y} = -out \; \textbf{inv} & max \leq y \\
& &]| : & & &
\end{array}
$$

Finally, we translate the hybrid action system into an ordinary action system.

$$
\begin{array}{lclll}
\mathcal{W}Tank & = & |[& \textbf{var} & y^* : I\!R; \;\; y := min; \\
& & & \textbf{do} & min < y < max \\
& & & & \qquad \rightarrow \;\; \dot{y} = in - out \; \textbf{inv} \; min < y < max \\
& & & \;[\!] & y \leq min \\
& & & & \qquad \rightarrow \;\; \dot{y} = in \; \textbf{inv} \; y \leq min \\
& & & \;[\!] & max \leq y \\
& & & & \qquad \rightarrow \;\; \dot{y} = -out \; \textbf{inv} \; max \leq y \\
& & & \textbf{od} & \\
& &]| : & &
\end{array}
$$

8.2 Travelling Train

In the following we develop an action system modelling a travelling train. We use several hybrid action systems to describe the phases during the travelling and the parallel composition to combine these phases into one hybrid action system. This example differs from the water tank example in the way that the phases are not overlapping. Hence, the resulting action system has no overlapping behaviour whatsoever.

Train

First, we define an action system that contains the variables needed in the model. These variables l, v, and st represent the location, the velocity, and the status of the travelling train. The train can either be still, accelerating, travelling with constant velocity, or decelerating. These are captured in the type of the status variable st. Initially, the train is standing by in the beginning location, b.

$$\begin{array}{lll} \mathcal{T}rain & \hat{=} & |[\ \textbf{var}\ \ l^*, v^* \ : \mathbb{R}; \\ & & \qquad\quad\ \ st^* \quad : \{still, acc, tra, dec\}; \\ & & \quad l, v, st := b, 0, still \\ & &]| : \end{array}$$

Departure

The train starts to departure whenever it is in the beginning location standing by. This is modeled by an action that sets the status of the train to acceleration. The constant acceleration of train is modelled as a system of differential equations.

$$\begin{array}{lll} \mathcal{D}epature & \hat{=} & |[\ \textbf{des}\ \ \dot{l}, \dot{v} = v, 1\ \textbf{inv}\ st = acc \\ & & \quad \textbf{do}\ \ l = b\ \wedge\ st = still \ \rightarrow\ st := acc\ \ \textbf{od} \\ & &]| : l, v, st \end{array}$$

Travelling

An action changes the status of the train from acceleration to travelling whenever the travelling velocity is reached. The travelling with the constant velocity is modelled as a differential equation.

$$\begin{array}{lll} \mathcal{T}ravelling & \hat{=} & |[\ \textbf{des}\ \ \dot{l} = v\ \textbf{inv}\ st = tra \\ & & \quad \textbf{do}\ \ v = v_t\ \wedge\ st = acc \ \rightarrow\ st := tra\ \ \textbf{od} \\ & &]| : l, v, st \end{array}$$

Arrival

When the train reaches a the location $e - \frac{v^2}{2} \leq l < e$, from which it can decelerate its velocity down to a full stop precisely at the destination location e, the status of the train is set to deceleration. The deceleration is finished, when the destination location is reached. Then, the status of the train is set to *still*. The deceleration is modelled as a system of differential equations.

$$
\begin{array}{lcl}
\mathcal{A}rrival & \hat{=} & |[\ \mathbf{des}\ \ \dot{l}, \dot{v} = v, -1\ \mathbf{inv}\ st = dec \\
 & & \quad \mathbf{do}\ \ st \neq dec\ \wedge\ e - \frac{v^2}{2} \leq l < e \ \ \rightarrow\ \ st := dec \\
 & & \quad [\!]\ \ \ e \leq l\ \wedge\ st \neq still \ \ \rightarrow\ \ st := still \\
 & & \quad \mathbf{od} \\
 & &]|: l, v, st
\end{array}
$$

Travelling train

The travelling train is a parallel composition of all these hybrid action systems.

$$
\mathcal{M}Train \ \hat{=}\ \mathcal{T}rain \parallel \mathcal{D}eparture \parallel \mathcal{T}ravelling \parallel \mathcal{A}rrival
$$

Obtaining an action system

Next, we obtain the equivalent action system for the hybrid action system $\mathcal{M}Train$. We start by unfolding the parallel composition.

$$
\begin{array}{lcl}
\mathcal{M}Train & = & |[\ \mathbf{var}\ \ l^*, v^* \ : \mathbb{R}; \\
 & & \qquad\qquad st^* \ : \{still, acc, tra, dec\}; \\
 & & \quad l, v, st := b, 0, still; \\
 & & \quad \mathbf{des}\ \ \dot{l}, \dot{v} = v, 1\ \mathbf{inv}\ st = acc \\
 & & \quad [\!]\ \ \ \dot{l} = v\ \mathbf{inv}\ st = tra \\
 & & \quad [\!]\ \ \ \dot{l}, \dot{v} = v, -1\ \mathbf{inv}\ st = dec \\
 & & \\
 & & \quad \mathbf{do}\ \ l = b\ \wedge\ st = still \ \ \rightarrow\ \ st := acc \\
 & & \quad [\!]\ \ \ v = v_t\ \wedge\ st = acc \ \ \rightarrow\ \ st := tra \\
 & & \quad [\!]\ \ \ st \neq dec\ \wedge\ e - \frac{v^2}{2} \leq l < e \ \ \rightarrow\ \ st := dec \\
 & & \quad [\!]\ \ \ e \leq l\ \wedge\ st \neq still \ \ \rightarrow\ \ st := still \\
 & & \quad \mathbf{od} \\
 & &]|:
\end{array}
$$

When translating this hybrid action system into an action system, we need the negation of the guard condition for the **do od**-loop. It is calculated to be

$$
\begin{array}{l}
(l \neq b\ \vee\ st \neq still)\ \wedge\ (v \neq v_t\ \vee\ st \neq acc)\ \wedge \\
(st = dec\ \vee\ e - \frac{v^2}{2} > l\ \vee\ l \geq e)\ \wedge\ (e > l\ \vee\ st = still)
\end{array}
$$

The action system obtained through the translation is

$$
\begin{array}{lll}
\mathcal{M}Train & = & |[\ \mathbf{var}\ \ l^*, v^* \ \ : I\!R; \\
 & & \qquad\quad st^* \quad\ : \{still, acc, tra, dec\}; \\
 & & \quad l, v, st := b, 0, still; \\
 & & \quad \mathbf{do}\ \ st = acc\ \wedge\ v \neq v_t\ \wedge\ l < e - \frac{v^2}{2} \\
 & & \qquad\qquad \rightarrow\ \ \dot{l}, \dot{v} = v, 1\ \mathbf{inv}\ \ st = acc\ \wedge\ v \neq v_t\ \wedge \\
 & & \qquad\qquad\qquad\qquad\qquad\qquad l < e - \frac{v^2}{2} \\
 & & \quad\ \ [\!]\ \ st = tra\ \wedge\ l < e - \frac{v^2}{2} \\
 & & \qquad\qquad \rightarrow\ \ \dot{l} = v\ \mathbf{inv}\ st = tra\ \wedge\ l < e - \frac{v^2}{2} \\
 & & \quad\ \ [\!]\ \ st = dec\ \wedge\ l < e \\
 & & \qquad\qquad \rightarrow\ \ \dot{l}, \dot{v} = v, -1\ \mathbf{inv}\ st = dec\ \wedge\ l < e \\
 & & \quad\ \ [\!]\ \ l = b\ \wedge\ st = still \qquad\qquad \rightarrow \quad st := acc \\
 & & \quad\ \ [\!]\ \ v = v_t\ \wedge\ st = acc \qquad\qquad \rightarrow \quad st := tra \\
 & & \quad\ \ [\!]\ \ st \neq dec\ \wedge\ e - \frac{v^2}{2} \leq l < e \ \ \rightarrow \quad st := dec \\
 & & \quad\ \ [\!]\ \ e \leq l\ \wedge\ st \neq still \qquad\qquad \rightarrow \quad st := still \\
 & & \quad \mathbf{od} \\
 & &]|:
\end{array}
$$

8.3 Fillers

In this example, we quickly explore the effect of having several similar continuous components acting in parallel. In the interleaving model, this would cause no effect at all. However, in the continuous parallelism, this makes a big difference, as we shall see.

The following is an example of having several fillers filling a container simultaneously. The filling process ends, when the container is full.

Container

We define the container as an action system containing the liquid level variable. In the beginning the container is empty

$$\mathcal{C}ontainer \ \ \widehat{=}\ \ |[\ \mathbf{var}\ \ y^* : I\!R;\ \ y := 0]|:$$

A filler

A filler is a hybrid action system. It fills the container with a constant rate, in, till the container is full, $y = max$.

$$\mathcal{F}iller \ \ \widehat{=}\ \ |[\ \mathbf{des}\ \ \dot{y} = in\ \mathbf{inv}\ y < max\ ]|: y$$

Composed system

Now, we can define a system of one container and two fillers as a parallel composition

$$\mathcal{F}illers \;\hat{=}\; \mathcal{C}ontainer \parallel \mathcal{F}iller \parallel \mathcal{F}iller$$

Obtaining an action system

We unfold this parallel composition into a single hybrid action system.

$$\begin{array}{lcll} \mathcal{F}illers & = & |[& \mathbf{var} \;\; y^* : I\!\!R;\; y := 0; \\ & & & \mathbf{des} \;\; \dot{y} = in \;\mathbf{inv}\; y < max \;\; + \\ & & & \qquad\; \dot{y} = in \;\mathbf{inv}\; y < max \\ & &]| : & \end{array}$$

This hybrid action system can be rewritten by unfolding the linear composition of the systems of differential equations.

$$\begin{array}{lcll} \mathcal{F}illers & = & |[& \mathbf{var} \;\; y^* : I\!\!R;\; y := 0; \\ & & & \mathbf{des} \;\; \dot{y} = 2 \cdot in \;\; \mathbf{inv}\; y < max \\ & &]| : & \end{array}$$

Finally, we translate the hybrid action system into an ordinary action system.

$$\begin{array}{lcll} \mathcal{F}illers & = & |[& \mathbf{var} \;\; y^* : I\!\!R;\; y := 0; \\ & & & \mathbf{do} \;\; y < max \;\; \rightarrow \;\; \dot{y} = 2 \cdot in \;\; \mathbf{inv}\; y < max \;\; \mathbf{od} \\ & &]| : & \end{array}$$

9 Conclusions

The hybrid action systems framework was extended from the action systems framework based on the general computational model for hybrid systems.

The parallel composition was extended for the use of hybrid action systems. This extension was based on the mathematical properties of the linear composition. Since the use of hybrid action systems and the extended parallel composition does not impose any further constraints, all the existing results for action systems are directly usable.

The analysis of the hybrid behaviour in a hybrid action system is obtained by translating it into an ordinary action system. As shown, this translation is mechanical.

The examples presented in this paper show how the parallel composition of hybrid action systems work also when some of the components are ordinary action systems.

References

[1] R. J. R. Back. *Refinement calculus, part II: Parallel and reactive programs.* In J. W. de Bakker, W.–P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness. Proceedings. 1989*, volume 430 of *Lecture Notes in Computer Science*. Springer–Verlag, 1990.

[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*, pages 131–142, 1983.

[3] R. J. R. Back and K. Sere. Stepwise refinement of action systems. *Structured Programming*, 12:17-30, 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, p. 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 Programming Languages and Systems*, vol. 16, no. 3, p. 924-938, May 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] C.J. Fidge, A.J.Wellings. *An action-based formal model for concurrent, real-time systems.* Technical Report 95-1, SVR Center, January 1995.

[10] R. Kurki-Suonio. *Stepwise Design of Real Time Systems.* IEEE Transactions on Software Engineering vol. 19, no. 1, pp. 56-69, January 1993.

[11] M. Rönkkö, E. Sekerinski, K. Sere. *Control Systems as Action Systems.* In Proc. of *Workshop on Discrete Event Systems (WODES'96)*, Edinburgh, UK, August 1996.

[12] M. Rönkkö, A. Ravn. *Differential Equations as Actions.* TUCS Techinal Report 109, Computer Science Department, Åbo Akademi, 1997. Also, submitted to 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**