Linear Hybrid Action Systems

Mauno Rönkkö and Xuandong Li

Turku Centre for Computer Science,
Department of Computer Science,
Åbo Akademi University,
Lemminkäisenkatu 14A,
FIN-20520 Turku, Finland,
{mronkko,xuandong.li}@abo.fi

Turku Centre for Computer Science
TUCS Technical Report No 245
2nd March 1999
ISBN 952-12-0397-8
ISSN 1239-1891

Abstract

Action Systems is a predicate transformer based formalism for the development of reactive and distributed systems. Recently, Action Systems were extended with a new action called the differential action. It models continuous behaviour. Action Systems with differential actions are used for modelling hybrid systems.

In this paper we investigate, what kind of hybrid systems are conveniently captured by Action Systems. For this purpose, we present a transition system semantics for Action Systems. We also study how a class of Action Systems called Linear Hybrid Action Systems is related to another formalism, linear hybrid automata. As the main result we show that Linear Hybrid Action Systems is a subclass of linear hybrid automata.

Keywords: action systems, linear hybrid systems, transition systems, linear hybrid automata

TUCS Research Group
Programming Methodology Research Group

1 Introduction

Action Systems [3, 4, 6] are predicate transformer based systems used especially in the development of reactive and concurrent systems. Recently, Rönkkö and Ravn [10] extended Action Systems with a new action called the *differential action*. It allows the use of differential equations in capturing continuous behaviour. The differential action has a predicate transformer semantics like the other actions, and therefore, it fits seamlessly into the Action Systems framework. Action Systems with differential actions model hybrid systems where a discrete controller interacts with some continuously evolving environment.

In the development of Action Systems the use of refinement plays a central role. The differential action makes no exception here. As shown by Rönkkö and Sere [11], there are simple refinement laws for the differential action allowing efficient development of provably correct hybrid systems.

However, it is not easy to see from the predicate transformer semantics, what kind of hybrid systems are conveniently captured by Action Systems. Also, the relation of Action Systems to other hybrid system formalisms is not clear. As an answer to these issues, the contribution of this paper is to present transition system semantics for Action Systems.

We then investigate a class of Action Systems called *Linear Hybrid Action Systems* that model linear hybrid systems. The reason why we study these systems is that in the literature there are more results for the linear hybrid systems than for the non-linear hybrid systems. Also, a non-linear hybrid system can be approximated with a linear hybrid system [9]. In this paper, we show that there exists a simple mapping that translates a linear hybrid action system to a linear hybrid automaton as defined, e.g., by Henzinger et al. [8]. As the main result we show that Linear Hybrid Action Systems is a subclass of linear hybrid automata.

The work presented in this paper builds a bridge between different formalisms. We show how to formally translate a linear hybrid action system to a linear hybrid automaton. The importance of this is that it allows the use of existing theoretical results of one formalism in the other. In particular, it gives the needed formal background to use tools like HyTech [9] for the analysis and verification of Linear Hybrid Action Systems.

Overview.

We start by defining actions in Section 2. In Section 3 we present Action Systems that model hybrid systems and give transition system semantics for them. In Section 4 we introduce linear hybrid automata and give the definition for Linear Hybrid Action Systems. In this section we also give a mapping that translates a linear hybrid action system to a linear hybrid automaton. We use this mapping to show that Linear Hybrid Action Systems is a subclass of linear hybrid automata. In Section 5 we illustrate how an action system specification of a water-level monitor is translated to a linear hybrid automaton. Finally, follows the conclusion in Section 6.

2 Actions

An action is any statement in Dijkstra's guarded command language [7]. Also, a pure guarded command can be used as an action. Actions operate on a fix set of program variables, and therefore, predicates over these variables specify states in a system.

When we consider a given action A, we may speak of a *postcondition* q. It is a predicate specifying desirable states after the execution of an action. The meaning of an action is defined with the *weakest precondition* predicate transformer $\mathrm{wp}(A,q)$. It is a predicate describing the largest set of states from which the action A terminates and reaches the postcondition q. The *termination*, i.e., reaching an arbitrary state, is given by $\mathrm{t}(A) \;\widehat{=}\; \mathrm{wp}(A, true)$. We may also speak of enabledness of an action, or *guard condition*, i.e., $\mathrm{g}(A) \;\widehat{=}\; \neg\mathrm{wp}(A, false)$. An action A is said to be enabled in all those states where $\mathrm{g}(A)$ holds.

Let $q[E/X]$ denote the textual substitution of free variables X with expressions E in a predicate q. Also, let e and p be predicates, A and B be actions, and τ and δ be real variables and not appearing in X, e, p, or q. The actions that we consider in this paper are defined as follows.

$$
\begin{array}{lll}
\mathrm{wp}(\mathbf{skip}, q) & \widehat{=}\; q & (\mathit{skip}) \\
\mathrm{wp}(X := E, q) & \widehat{=}\; q[E/X] & (\mathit{assignment}) \\
\mathrm{wp}(A; B, q) & \widehat{=}\; \mathrm{wp}(A, \mathrm{wp}(B,q)) & (\mathit{seq.\ composition}) \\
\mathrm{wp}(A \,[\!]\, B, q) & \widehat{=}\; \mathrm{wp}(A,q) \wedge \mathrm{wp}(B,q) & (\mathit{non-det.\ choice}) \\
\mathrm{wp}(p \rightarrow A, q) & \widehat{=}\; p \Rightarrow \mathrm{wp}(A,q) & (\mathit{guarded\ command}) \\
\mathrm{wp}(\mathbf{do}\, A\, \mathbf{od}, q) & \widehat{=}\; \exists n : n = \min\{i \mid i \geq 0 \wedge \neg\, \mathrm{g}(A^{i+1})\} & (\mathit{iteration}) \\
 & \qquad . \, \mathrm{wp}(A^n, q) & \\
\mathrm{wp}(e : \dot{X} = F(X), q) & \widehat{=}\; \exists!\, \Phi : (\Phi(0) = X \wedge \dot{\Phi} = F(\Phi)) & (\mathit{differential\ action}) \\
 & \qquad . \; \exists\, \delta : \delta = \inf\{\tau \mid \tau \geq 0 \wedge \neg e[\Phi(\tau)/X]\} & \\
 & \qquad\qquad . \; q[\Phi(\delta)/X] &
\end{array}
$$

where $A^0 \;\widehat{=}\; \mathbf{skip}$ and $A^{n+1} \;\widehat{=}\; A; A^n$. The differential action is used for modelling and reasoning about continuous behaviour. Let e be a predicate over the variables X, and $\dot{X} = F(X)$ be a system of differential equations, where $\dot{X}$ is a componentwise first derivative of X and $F(X)$ is a vector of functions with an equal number of components as in X. A differential action, $e : \dot{X} = F(X)$, evolves continuously the values of X provided that the evolution guard e holds. The evolution, which is given by a system of differential equations, stops when the values of X have reached the border of e. In the definition for the differential action, the first existential quantification requires the existence of a unique solution Φ for $\dot{X} = F(X)$, which is an *autonomous* system of differential equations, i.e., it may not refer to time explicitly. Time is implicit and it is modelled by the variable τ. The second existential quantification requires that the evolution eventually reaches the border of the evolution guard e and that the reached values satisfy the post-condition q. The variable δ is used for denoting the time, when the border of e is reached. In this paper, we assume that e describes a group of closed intervals, e.g., $0 \leq x \leq 3$. If e does not hold at the beginning of an evolution, δ becomes zero forcing the evolution not to start at all. Hence, the semantics resembles the semantics for an iteration given earlier. Clearly, the differential action is deterministic in its ending time. However, the general Action Systems have also a *non-deterministic assignment* [5], which, when composed sequentially with a differential action, could be used for modelling continuous behaviour with a non-deterministic ending time.

3 Action Systems

So far we have been describing individual actions. However, a hybrid system can very seldom be modelled by one action. For this purpose we use Action Systems. Here, we consider a special form of Action Systems where each action consists of both a discrete step and a continuous step. This kind of action system is particularly suitable for modelling hybrid systems. Formally, an action system $\mathcal{A}$ of the described form is an initialized block

$$\mathcal{A} \;\widehat{=}\; |[\, \mathbf{var}\ X : T \;\bullet\; X := E_0;\ \mathbf{do}\, A_1 \,[\!]\, .. \,[\!]\, A_n\, \mathbf{od}\,]| \;.$$

The expression $\mathbf{var}\ X : T$ declares a list of variables X with types T. These variables form the *state space* of $\mathcal{A}$. The action $X := E_0$ initializes all the variables X by expressions E_0. These expressions are constant expressions and may not refer to any variable, because the values of the variables are undefined before the initialization.

The number of actions A_i in $\mathcal{A}$ is countable, i.e., $i \in 1..n$. Also, each action A_i is of form $p_i \rightarrow X := E_i;\ e_i : \dot{X} = F_i(X)$, such that $p_i \Rightarrow e_i[E_i/X]$ holds. In other words, each action is a guarded command, where a differential action follows an assignment statement, and the evolution guard of the differential action holds always after the execution of the preceding assignment statement. An arbitrary action system is rewritten into this form by using refinement [1, 11]. An enabled action A_i is selected non-deterministically for execution after the initialization. There are no fairness assumptions about the selection of actions. The execution of an action A_i is always atomic. This means that also the continuous evolution of the differential action in A_i proceeds without interruption up to the border of the evolution guard. Only after that the other enabled actions have a chance for execution. If two enabled actions refer to disjoint variables, their execution can be in any order. Hence, this models parallelism by interleaving.

The execution of $\mathcal{A}$ terminates when none of the actions is enabled, i.e., $\neg\, \mathrm{g}(A_1 \,[\!]\, .. \,[\!]\, A_n)$. An action A_i is said to *abort*, if it does not terminate, i.e.,

$\neg \mathrm{t}(A_i)$ holds. Likewise, an action system aborts when an executed action aborts.

3.1 Action systems as transition systems

Due to the required form for actions an execution of an action A_i forms a sequence of a discrete step and a continuous step. The beginning of an action, $p_i \rightarrow X := E_i$, corresponds to the discrete step. The guard of the step is p_i, and the reset action is $X := E_i$. Since a state in an action system is solely determined by the values of the variables, a discrete step may occur at any state where the guard holds. After the discrete step comes the continuous step that corresponds to the execution of the latter part of the action, the differential action $e_i : \dot{X} = F_i(X)$. Due to the atomicity the values of the variables are continuously evolved up to the border of the evolution guard e_i without interruption during the continuous step. After one pair of a discrete and a continuous step has been executed, comes another pair of a discrete and a continuous step of another action A_j whose guard p_j is enabled. Based on this observation, we give a transition system semantics for Action Systems.

Definition 1. Let $\mathcal{A}$ be an action system as above. We give a label to each action in $\mathcal{A}$, such that, l_0 is the label for the initialization of $\mathcal{A}$ and $l_1..l_n$ are the labels for the actions $A_1..A_n$. With $\mathcal{A}$ we associate a labeled timed transition system $\mathcal{TA} = (\Sigma, Lab \cup I\!R_{\geq 0}, \rightarrow)$ in the following way. In $\mathcal{TA}$, the Σ is an infinite set of states. Every state (l_i, ν, t_0) is a tuple of an action label l_i, values ν of the variables X, and a time stamp t_0. In other words, the states in $\mathcal{TA}$ are associated with the actions in $\mathcal{A}$. Thus, the initial state of $\mathcal{TA}$ is the tuple $(l_0, E_0, 0)$. Lab is a finite set of discrete transition labels $a_{0..n \times 1..n}$ and $I\!R_{\geq 0}$ is an infinite set of continuous transition labels. The transition relation, $\rightarrow$, contains the discrete transitions

$$\frac{\begin{array}{cr} i \in 0..n \quad j \in 1..n & \text{(D1)} \\ X = \nu \;\Rightarrow\; e_i \wedge \mathrm{wp}(e_i : \dot{X} = F_i(X), X = \nu) \wedge p_j & \text{(D2)} \\ X = \nu \;\Rightarrow\; \mathrm{wp}(X := E_j, X = \nu') & \text{(D3)} \\ X = \nu' \;\Rightarrow\; e_j & \text{(D4)} \end{array}}{(l_i, \nu, t_0) \rightarrow^{a_{ij}} (l_j, \nu', t_0)}$$

and the continuous transitions

$$\frac{\begin{array}{cr} i \in 0..n \quad \delta \in I\!R_{\geq 0} & \text{(D5)} \\ X = \nu \;\Rightarrow\; e_i & \text{(D6)} \\ X = \nu \;\Rightarrow\; \mathrm{wp}(t := t_0;\; e_i \wedge t \leq t_0 + \delta : \dot{X}, \dot{t} = F_i(X), 1\;,\; X = \nu' \wedge t = t_0 + \delta) & \text{(D7)} \end{array}}{(l_i, \nu, t_0) \rightarrow^{\delta} (l_i, \nu', t_0 + \delta)}$$

where e_0 is the predicate $X = E_0$, $F_0(X)$ is an identity vector, and t is an auxiliary variable not appearing in X.
End of definition

In the definition above the discrete transitions correspond to the discrete steps of actions, and the continuous transitions correspond to the continuous steps of actions.

For the discrete transitions, the labels $a_{0..n \times 1..n}$ are associated with the execution order of the actions $A_1..A_n$. For instance, a label a_{ij} denotes the discrete step of the action A_j after the continuous step of the action A_i, and a label a_{0j} denotes the discrete step of the action A_j after the initialization. Because the initialization of the action system $\mathcal{A}$ is executed only once and it is associated with the subindex 0, the values of j in the condition (D1) of the discrete transition rule are limited to $1..n$. The initialization of the action system itself is encoded to the initial state of the transition system $\mathcal{T}A$. The condition (D2) states that the values ν of the variables before the discrete transition must be the termination values of the previously executed differential action $e_i : \dot{X} = F_i(X)$. This reflects the atomicity of the continuous step of the action A_i, which proceeds uninterrupted till termination before the discrete step of the action A_j occurs. Moreover, the condition (D2) states that the discrete step of the action A_j takes place only if its guard p_j holds. The condition (D3) states that the values ν' of the variables after the discrete step must be the values that are obtained with the reset action $X := E_j$ of A_j. The assumption $p_j \Rightarrow e_j[E_j/X]$ is written as the condition (D4). This condition ensures that the evolution guard of the differential action in A_j is always enabled right after its reset action.

The continuous transitions that take time occur during the continuous steps of actions. The condition (D5) of the continuous transition rule ensures that time does not decrease during a continuous step. According to the condition (D6), a continuous transition in a state labeled l_i may occur, only if the corresponding continuous step of the action A_i occurs, i.e., the evolution guard e_i of the differential action holds. In the condition (D7), the auxiliary variable t is used for measuring time during the continuous evolution. The postcondition in the weakest precondition expression in the condition (D7) requires that t advances δ time units. This ensures that from the values $X = \nu$ the continuous transition reaches in δ time units the values $X = \nu'$ only if the corresponding continuous step of A_i does so as well. As already noted, the atomicity of the continuous step is guaranteed by the condition (D2) of the discrete transition rule. By assumption, the evolution guard e_i describes a closed interval, and therefore, e_i still holds, when the continuous step terminates. Moreover, since the condition (D7) states that during a continuous transition time must progress the required δ time units, the evolution guard e_i must hold at all times during the transition. Otherwise, this requirement would not be satisfied.

The transition system semantics shows that the number of actions in an action system modelling a hybrid system depends only on the number of different kinds of continuous steps in the system. Therefore, a change in the discrete steps in a hybrid system does not affect at all the number of actions in the corresponding action system. This makes Action Systems an attractive formalism, whenever the specification is a subject to change, or when there are several possible discrete steps after a continuous step. However, during a continuous step there is no discrete transition taking place in the transition system semantics. Therefore, the kind of the action systems considered here cannot model hybrid systems where a discrete transition may interrupt a continuous step spontaneously.

4 Linear Hybrid Systems

An important class of hybrid systems is the class of linear hybrid systems. In these systems all the continuous behaviour progresses at a constant rate. Linear hybrid systems can be described by using only *linear expressions*, i.e., linear combination of variables with constant coefficients. Such a class of hybrid systems is covered by, e.g., *linear hybrid automata*.

In the literature, there are more results for the linear hybrid systems than for the non-linear hybrid systems. For instance, the verification of hybrid automata in general is difficult, but there are some results for the analysis and verification of linear hybrid automata, see Alur et al.[2] and Li et al. [12]. Moreover, a non-linear hybrid system can be approximated with a linear hybrid system [9].

In the following we investigate the relation of linear hybrid automata and their counterpart in Action Systems called Linear Hybrid Action Systems. We show that there exists a mapping from a linear hybrid action system to a linear hybrid automata, and that the class of linear hybrid action systems is, in fact, a subclass of linear hybrid automata.

4.1 Labeled linear hybrid automata

Based on the definition of linear hybrid automata [8, 9] we define a linear hybrid automaton $\mathcal{H}$ where also the discrete transitions are given labels:

$$\mathcal{H} \mathrel{\widehat{=}} (Var, Loc, Init, Inv, Trans, Rate) \quad .$$

Here, Var is a vector of variables, Loc is a set of locations, $Init$ is a predicate describing the initial condition, Inv is a set of predicates describing invariant conditions at locations, $Trans$ is a set of discrete transitions between the locations, and $Rate$ is a set of vectors denoting the rate of the change of the variables at locations. The discrete transitions are given labels, so that in $Trans$ a transition is denoted by a tuple (a, l, g, r, l'), where a is a label, l and l' refer to pre and post locations of the transition, g is a predicate describing the guard of the transition, and r is a vector describing the reset expression on the variables. In $\mathcal{H}$, all the expressions in $Init$, Inv, and $Trans$ are linear expressions.

We specialize the transition system semantics for hybrid systems given by Alur et al. [2] to that of linear hybrid systems as described by Henzinger et al. [9]. Thus, with a labeled linear hybrid automaton $\mathcal{H}$ we associate a labeled timed transition system $\mathcal{TH} = (\Sigma, Lab \cup \mathbb{R}_{\geq 0}, \rightarrow)$. In $\mathcal{TH}$, Σ is an infinite set of states, where a state (l, ν, t_0) is a tuple of a state label l, values ν of the variables Var, and a time stamp t_0. The initial state of $\mathcal{TH}$ is the tuple $(l, \nu, 0)$, where the location l and the values ν of the variables Var satisfy the initial condition $Init$. Lab is a finite set of labels associated with the transition labels in $Trans$, and $\mathbb{R}_{\geq 0}$ is an infinite set of labels used in continuous transitions. The transition relation, $\rightarrow$, contains the discrete transitions

$$\frac{\begin{array}{c} l, l' \in Loc \\ (a, l, g, r, l') \in Trans \\ Var = \nu \;\Rightarrow\; Inv(l) \wedge g \\ Var = \nu \;\Rightarrow\; r = \nu' \\ Var = \nu' \;\Rightarrow\; Inv(l') \end{array}}{(l, \nu, t_0) \rightarrow^{a} (l', \nu', t_0)}$$

and the continuous transitions

$$\frac{\begin{array}{c} l \in Loc \qquad \delta \in \mathbb{R}_{\geq 0} \\ \forall \tau : 0 \leq \tau \leq \delta \;.\; Var = \nu + \tau \cdot Rate(l) \;\Rightarrow\; Inv(l) \end{array}}{(l, \nu, t_0) \rightarrow^{\delta} (l, \nu + \delta \cdot Rate(l), t_0 + \delta)} \;.$$

In the discrete transition rule above, the first and the second condition ensure that the locations, the transition label, the guard, and the reset expression of the discrete transition belong to the labeled linear hybrid automaton in question. The third condition ensures that the values ν of the variables before the transition satisfy the invariant of the location l and the transition guard g. The fourth condition states that the values ν' of the variables after the transition are obtained with the reset expression, and the fifth condition ensures that those values also satisfy the invariant of the location l'.

In the continuous transition rule above, the first condition ensure that time does not decrease. The second condition states that the continuously evolved values not only satisfy the invariant of the location l during the evolution, but that they also evolve at the rate given by the rate vector of that location. Moreover, the second condition ensures that the new values $\nu + \delta \cdot Rate(l)$ for the variables are reached in δ time units.

4.2 Linear hybrid action systems

The subclass of Action Systems modelling linear hybrid systems is called Linear Hybrid Action Systems. It is defined as follows.

Definition 2. A linear hybrid action system $\mathcal{LA}$ is of form

$$\mathcal{LA} \;\widehat{=}\; |[\,\mathbf{var}\ X : T \;\bullet\; X := E_0;\ \mathbf{do}\ A_1 \,[\!]\, .. \,[\!]\, A_n\ \mathbf{od}\,]|\ ,$$

where each A_i is of form $p_i \rightarrow X := E_i;\ e_i : \dot{X} = C_i$, such that, all the expressions in p_i, E_i, and e_i are linear expressions, e_i is a predicate describing a group of closed intervals, $p_i \Rightarrow e_i[E_i/X]$ holds, and C_i is a vector of constants.

End of definition

Clearly, the class of linear hybrid action systems is a subclass of the Action Systems as defined in Section 3. Therefore, the labeled transition system semantics for Action Systems given in Definition 1 covers also the class of linear hybrid action systems.

As the following theorem shows, there exists a simple mapping from a linear hybrid action system to a labeled linear hybrid automata.

Theorem 3. Let $\mathcal{LA}$ be as in Definition 2. Also, let e_0 be the predicate $X = E_0$ and C_0 be an identity vector. Then, a labeled linear hybrid automaton

$$\mathcal{HA} \;\widehat{=}\; (X,\ \{l_0..l_n\},\ (l_0, e_0),\ \{e_0..e_n\},\ \{\sigma_{01}..,\sigma_{0n},\sigma_{11},..\sigma_{nn}\},\ \{C_0..C_n\})\ ,$$

where $\sigma_{ij} \;\widehat{=}\; (a_{ij},\ l_i,\ p_j \wedge e_i \wedge (\exists\,\Delta . X = \Delta \wedge \mathrm{wp}(e_i : \dot{X} = C_i, X = \Delta)),\ E_j,\ l_j)$ and Δ is a vector, has the same labeled transition system semantics as $\mathcal{LA}$.

Proof. By unfolding the transition system semantics for the labeled linear hybrid automaton we get $\mathcal{THA} = (\Sigma, \{a_{01}.., a_{0n}, a_{11}, ..a_{nn}\} \cup \mathbb{R}_{\geq 0}, \rightarrow)$. In $\mathcal{THA}$, Σ is an infinite set of states as before, and the initial state of $\mathcal{THA}$ is the tuple $(l_0, E_0, 0)$. The transition relation, $\rightarrow$, contains the discrete transitions

$$\frac{\begin{array}{lr} l_i, l_j \in \{l_0..l_n\} & \text{(T1)} \\ (a_{ij}, l_i, g, r, l_j) \in \{\sigma_{01}.., \sigma_{0n}, \sigma_{11}, ..\sigma_{nn}\} & \text{(T2)} \\ X = \nu \;\Rightarrow\; e_i \wedge p_j \wedge \mathrm{wp}(e_i : \dot{X} = C_i, X = \nu) & \text{(T3)} \\ X = \nu \;\Rightarrow\; E_j = \nu' & \text{(T4)} \\ X = \nu' \;\Rightarrow\; e_j & \text{(T5)} \end{array}}{(l_i, \nu, t_0) \rightarrow^{a_{ij}} (l_j, \nu', t_0)}$$

and continuous transitions

$$\frac{\begin{array}{lr} l_i \in \{l_0..l_n\} \quad \delta \in \mathbb{R}_{\geq 0} & \text{(T6)} \\ \forall\,\tau : 0 \leq \tau \leq \delta\ .\ X = \nu + \tau \cdot C_i \;\Rightarrow\; e_i & \text{(T7)} \end{array}}{(l_i, \nu, t_0) \rightarrow^{\delta} (l_i, \nu + \delta \cdot C_i, t_0 + \delta)}\ .$$

Now we argue, why the conditions (T1)..(T7) above are equal to the conditions (D1)..(D7) that are obtained by unfolding directly the transition system semantics for the linear action system $\mathcal{LA}$ by using Definition 1.

It is easy to prove that the equalities $(T1) \wedge (T2) \Leftrightarrow (D1)$, $(T3) \Leftrightarrow (D2)$,
$(T4) \Leftrightarrow (D3)$, and $(T5) \Leftrightarrow (D4)$ hold, and therefore, the conditions (T1)..(T5) for the discrete transition rule above equal to the corresponding conditions (D1)..(D4).

Similarly, it is easy to prove for the continuous transition rule that the equality $(T6) \Leftrightarrow (D5)$ holds, and therefore, the only equality that remains to be proven is $(T7) \Leftrightarrow (D6) \wedge (D7)$. We prove this in two parts, first to direction " $\Rightarrow$ ". Clearly, $(T7) \Rightarrow (D6)$ holds. Unfolding the condition (D7) yields

$$X=\nu \quad \Rightarrow \quad \exists\, \epsilon : \epsilon = \inf\{\tau \mid \tau \geq 0 \wedge \neg(e_i[\nu+\tau \cdot C_i/X] \wedge \tau \leq \delta)\}.\ \nu+\epsilon \cdot C_i = \nu' \wedge \epsilon = \delta \quad .$$

Under the condition $(T7)$ the infimum expression above simplifies to δ, whereby the whole expression above can be rewritten as $X \neq \nu \vee \nu+\delta \cdot C_i = \nu'$. Since $\nu+\delta C_i$ is the value ν' after the continuous transition, this disjunct holds trivially, and therefore, $(T7) \Rightarrow (D6) \wedge (D7)$ holds as well. Next we prove that the equality $(T7) \Leftrightarrow (D6) \wedge (D7)$ holds to direction " $\Leftarrow$ ". This can also be proven in two cases depending on whether $X=\nu$ holds. Proving that $(T7) \Leftarrow (D6) \wedge (D7)$ holds whenever $X \neq \nu$ holds is trivial. Now, consider the predicate $(T7) \Leftarrow (D6) \wedge (D7)$ when $X=\nu$ holds. We can simplify this predicate into

$$\begin{aligned} & e_i \wedge \exists\, \epsilon : \epsilon = \inf\{\tau \mid \tau \geq 0 \wedge \neg(e_i[\nu+\tau \cdot C_i/X] \wedge \tau \leq \delta)\}.\ \nu+\epsilon \cdot C_i = \nu' \wedge \epsilon = \delta \\ & \Rightarrow \forall \tau : 0 \leq \tau \leq \delta\ .\ X = \nu+\tau \cdot C_i \ \Rightarrow\ e_i \quad . \end{aligned}$$

Here, ϵ can only get values from the interval $[0, \delta]$ and the predicate e_i describes a closed interval. Therefore, it suffices to analyze the cases when $0 \leq \epsilon < \delta$ holds and when $\epsilon = \delta$ holds. In both of these cases the predicate above holds trivially. This concludes the proof of the predicate $(T7) \Leftarrow (D6) \wedge (D7)$. We have now shown that also the equality $(T7) \Leftrightarrow (D6) \wedge (D7)$ holds, and therefore the the conditions (T6) and (T7) for the continuous transition rule equal to the corresponding conditions (D5)..(D7). Since both the discrete and the continuous transition rules have equal conditions in $\mathcal{THA}$ and in the labeled transition system semantics of $\mathcal{LA}$, these semantics are the same. $\square$

Theorem 3 shows that any linear hybrid action system can be represented as a labeled linear hybrid automaton. However, the converse is not true, as argued in the following.

Corollary 4. The class of hybrid systems captured by Linear Hybrid Action Systems is a subset of the class of hybrid systems captured by linear hybrid automata.

Proof. Theorem 3 proves that all the linear hybrid action systems can be represented as labeled linear hybrid automata. Now, consider a labeled linear hybrid automaton

$$\mathcal{H} \ \widehat{=}\ (x,\ \{l_0,l_1\},\ (l_0,x=0),\ \{e_0,e_1\},\ \{\sigma_{01}\},\ \{C_0,C_1\})\ ,$$

where $e_0 \ \widehat{=}\ x \leq 2$, $e_1 \ \widehat{=}\ x \geq 1$, $\sigma_{01} \ \widehat{=}\ (a_{01},\ l_0,\ x \geq 1,\ x,\ l_1)$, $C_0 = 1$, and $C_1 = 1$. In the discrete transition σ_{01} the guard $x \geq 1$ allows the transition to occur before the value of x in location l_0 has reached the border of the invariant $x \leq 2$. According to the transition system semantics, in such a case the discrete transition may take place at any time, when $1 \leq x \leq 2$ holds. In other words, the continuous step has a non-deterministic ending time. However, it is not possible to model this with a linear hybrid action system, because the continuous step of an action has always a deterministic ending time. Therefore, the labeled linear hybrid automaton above cannot be expressed as a linear hybrid action system. Hence the class of hybrid systems captured by linear hybrid action systems is a subset of the class of hybrid systems captured by linear hybrid automata. $\square$

5 Example: Water-Level Monitor

As an example for illustrating the translation of a linear hybrid action system to a linear hybrid automaton we use a water-level monitor example that has been presented in several papers before, e.g., by Alur et al. [2] and Li et al. [12]. In this example, the task of the water-level monitor is to keep the water level of a tank within a certain interval. The monitor can either pump water into the tank or out of the tank. It takes two seconds for the monitor to switch the pumping direction. When the water is being pumped into the tank, the water level rises at the rate of 1 inch per second, and when the water is being pumped out of the tank, the water level falls at the rate of two inches per second. There are two sensors in the tank for sensing when the water level is at five or ten inches. When the water level reaches the sensor at five inches the monitor switches from pumping water out from the tank to pumping water into the tank. Likewise, when the water reaches the sensor at ten inches the monitor switches from pumping water into the tank to pumping water out from the tank. Figure 1 depicts the characteristics of the water tank.

The action system below is a specification of the water-level monitor.

$$
\begin{array}{lll}
\mathit{Monitor} & \widehat{=} & |[\ \mathbf{var}\ x,y : \mathbb{R},\mathbb{R}\ \bullet\ x,y := 2,1; \\
 & & \quad \mathbf{do}\ \ x=2 \wedge y \leq 5 \ \ \to x,y := x,y;\ \ y \leq 10 : \dot{x},\dot{y} = 1,1 \\
 & & \quad \ \ [\!]\ \ y=10 \qquad\qquad \to x,y := 0,y;\ \ x \leq 2 : \dot{x},\dot{y} = 1,1 \\
 & & \quad \ \ [\!]\ \ x=2 \wedge y \geq 10 \to x,y := x,y;\ \ y \geq 5 : \dot{x},\dot{y} = 1,-2 \\
 & & \quad \ \ [\!]\ \ y=5 \qquad\qquad\ \to x,y := 0,y;\ \ x \leq 2 : \dot{x},\dot{y} = 1,-2 \\
 & & \quad \mathbf{od} \\
 & &]|
\end{array}
$$

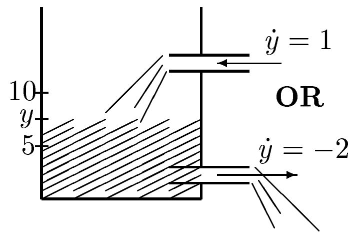

Figure 1: The water tank with threshold water levels and flow rate effects on the water level in inches per second.

There are two variables, x and y, in $\mathcal{M}onitor$. The actual water level is denoted by y, whereas x is a clock variable used for modelling the two second delay associated with the switching of the pumping direction. Initially, $x=2$ and the water level is at one inch, $y=1$.

There are four actions in $\mathcal{M}onitor$. The first action models the filling of the tank. $\mathcal{M}onitor$ may enter this phase, whenever the guard $x=2 \wedge y \leq 5$ holds. This happens, in particular, after the initialization. The reset action $x, y := x, y$ after the guard states that the values of the variables do not change when the monitor decides to start filling the tank. According to the differential action, $y \leq 10 : \dot{x}, \dot{y} = 1, 1$, the filling proceeds till the water level y in the tank reaches the sensor located at ten inches. Also, the clock variable x advances during the filling.

The second action models the two second delay when the monitor switches from pumping water into the tank to pumping water out of the tank. In order to measure the two second delay, the reset action sets the clock variable x to zero. Note that during the delay, the water level still rises at the rate of one inch per second.

Similarly to the first action, the third action models the phase when the water is being pumped out of tank. During this phase the water level falls at the rate of two inches per second. This phase ends when the water level y in the tank reaches the sensor located at five inches.

The fourth action models the two second delay when the monitor switches from pumping water out of the tank to pumping water into the tank. During this delay, the water level still falls at the rate of two inches per second.

Since all the actions in $\mathcal{M}onitor$ satisfy the requirements of Definition 2, $\mathcal{M}onitor$ describes a linear hybrid action system. Therefore, we can translate $\mathcal{M}onitor$ to a linear hybrid automaton by using Theorem 3. As a result we get the automaton

$$\mathcal{A}monitor \;\widehat{=}\; ((x,y),\ \{l_0..l_4\},\ (l_0,e_0),\ \{e_0..e_4\},\ \{\sigma_{01}..,\sigma_{04},\sigma_{11},..\sigma_{44}\},\ \{C_0..C_4\})\,.$$

In $\mathcal{A}monitor$, the invariants conditions at locations $l_0..l_4$ are

$$e_0 \;\widehat{=}\; x=2 \wedge y=1,\quad e_1 \;\widehat{=}\; y \leq 10,\quad e_2 \;\widehat{=}\; x \leq 2,\quad e_3 \;\widehat{=}\; y \geq 5,\quad \text{and } e_4 \;\widehat{=}\; x \leq 2.$$

The transitions, where the guard expressions have been simplified using predicate logic, are

$$\begin{array}{lcl}
\sigma_{01} & \widehat{=} & (a_{01},\ l_0,\ x=2 \wedge y=1,\ (x,y),\ l_1)\ , \\
\sigma_{12} & \widehat{=} & (a_{12},\ l_1,\ y=10,\ (0,y),\ l_2)\ , \\
\sigma_{13} & \widehat{=} & (a_{13},\ l_1,\ x=2 \wedge y=10,\ (x,y),\ l_3)\ , \\
\sigma_{21} & \widehat{=} & (a_{21},\ l_2,\ x=2 \wedge y \leq 5,\ (x,y),\ l_1)\ , \\
\sigma_{22} & \widehat{=} & (a_{22},\ l_2,\ x=2 \wedge y=10,\ (0,y),\ l_2)\ , \\
\sigma_{23} & \widehat{=} & (a_{23},\ l_2,\ x=2 \wedge y \geq 10,\ (x,y),\ l_3)\ , \\
\sigma_{24} & \widehat{=} & (a_{24},\ l_2,\ x=2 \wedge y=5,\ (0,y),\ l_4)\ , \\
\sigma_{31} & \widehat{=} & (a_{31},\ l_3,\ x=2 \wedge y=5,\ (x,y),\ l_1)\ , \\
\sigma_{34} & \widehat{=} & (a_{34},\ l_3,\ y=5,\ (0,y),\ l_4)\ , \\
\sigma_{41} & \widehat{=} & (a_{41},\ l_4,\ x=2 \wedge y \leq 5,\ (x,y),\ l_1)\ , \\
\sigma_{42} & \widehat{=} & (a_{42},\ l_4,\ x=2 \wedge y=10,\ (0,y),\ l_2)\ , \\
\sigma_{43} & \widehat{=} & (a_{43},\ l_4,\ x=2 \wedge y \geq 10,\ (x,y),\ l_3)\ , \text{ and} \\
\sigma_{44} & \widehat{=} & (a_{44},\ l_4,\ x=2 \wedge y=5,\ (0,y),\ l_4)\ .
\end{array}$$

The transitions $\sigma_{02}, \sigma_{03}, \sigma_{04}, \sigma_{11}, \sigma_{14}, \sigma_{32}$, and σ_{33} are not listed above, because their guards evaluate to $false$. Transitions with such guards are disabled at all states. Figure 2 shows graphically $\mathcal{A}monitor$ with all the transitions whose guard is not false. Although the guards of the transitions

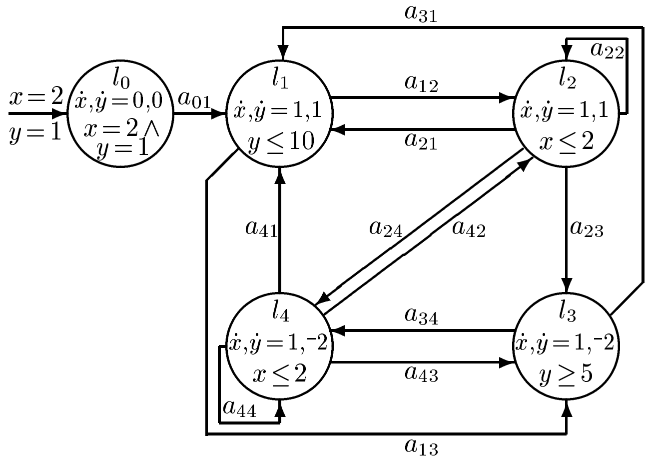

Figure 2: The linear hybrid automaton obtained as a translation from the action system describing the water-level monitor.

$\sigma_{13}, \sigma_{21}, \sigma_{22}, \sigma_{24}, \sigma_{31}, \sigma_{42}, \sigma_{43}$, and σ_{44} above are different from false, one can show by reachability analysis that these transitions are never enabled in a run starting from the initial condition. Therefore these transitions can also be removed from the automaton without changing the overall behaviour. Figure 3 depicts $\mathcal{A}monitor$ without the disabled transitions. This figure also reveals the underlying execution sequence of the actions in the action system $\mathcal{M}onitor$.
Lastly, the rates in $\mathcal{A}monitor$ are

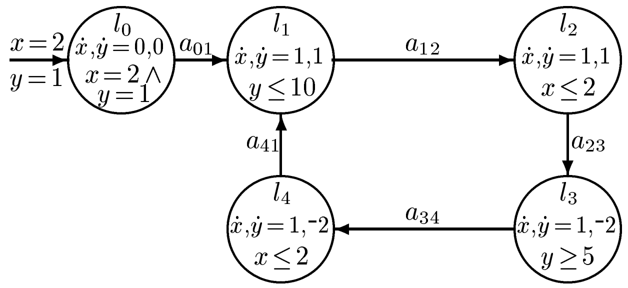

Figure 3: The linear hybrid automaton in Figure 2 without the disabled transitions.

$$C_0 \mathrel{\widehat{=}} (1,1),\ C_1 \mathrel{\widehat{=}} (1,1),\ C_2 \mathrel{\widehat{=}} (1,1),\ C_3 \mathrel{\widehat{=}} (1,-2),\ \text{and } C_4 \mathrel{\widehat{=}} (1,-2)\ .$$

6 Conclusion

In this paper we presented a transition system semantics for Action Systems modelling hybrid systems. This semantics shows us that the number of actions in an action system modelling a hybrid system depends only on the number of different kinds of continuous behaviours in the system. However, due to atomicity the kind of the Action Systems that were considered in this paper cannot model hybrid systems where continuous steps have non-deterministic ending time.

The transition system semantics provides us also a way to relate Action Systems to other formalisms. We illustrated this with a class of Action Systems called Linear Hybrid Action Systems. We showed that there exists a simple mapping translating a linear hybrid action system to a linear hybrid automaton. In addition, we showed that the class of hybrid systems captured by Linear Hybrid Action Systems is a subclass of hybrid systems captured by linear hybrid automata. However, this does not mean that Action Systems in general are not as expressive as linear hybrid automata. The major reason for Linear Hybrid Action Systems being a subclass of linear hybrid automata is the differential action which is deterministic in its ending time. The differential action cannot model by itself the "may" transitions present in the linear hybrid automata. The differential action insist upon "must" transitions. However, by adding an auxiliary variable and allowing non-deterministic assignments in Action Systems one can work around this difference. Such Action Systems have the capacity of expressing also the "may" transitions.

The translation of a linear hybrid action system to a linear hybrid automaton was illustrated with a water-level monitor example. This example also shows, how the translation can be used for extracting the underlying sequential behaviour of actions in a linear hybrid action system. The importance of being able to formally translate a linear hybrid action system to a linear hybrid automaton is that it allows the use of existing theoretical results of one formalism in the other. In particular, it gives the needed formal background to use tools like HyTech [9] for the analysis and verification of linear hybrid action systems. Therefore, an important topic for future research is to see how the existing theoretical results for hybrid automata can be applied for Action Systems.

Since Linear Hybrid Action Systems is a subclass of linear hybrid automata and the reachability problem in linear hybrid automata is generally undecidable, an interesting topic for future research is to see if the Linear Hybrid Action Systems have some distinct reachability properties.

Acknowledgements

We would like to thank Kaisa Sere for comments on the earlier version of this paper.

References

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

[2] R. Alur, C. Couroubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho et al. The Algorithmic Analysis of Hybrid Systems. In *Theoretical Computer Science*, 138, 1995, pp. 3-34.

[3] 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.

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

[5] R. J. R. Back and J. von Wright. *Trace Refinement of Action Systems.* In J. Parrow, editor, *CONCUR94*, LNCS 666, Springer-Verlag, 1994.

[6] 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.

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

[8] T. A. Henzinger. The Theory of Hybrid Automata. In Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996), pp. 278-292.

[9] T. A. Henzinger and P.-H. Ho. HyTech: The Cornell hybrid technology tool. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, LNCS 999, pp. 265-293, Springer-Verlag, 1995.

[10] M. Rönkkö and A. P. Ravn. *Action Systems with Continuous Behaviour.* To appear in the *Proceedings of Hybrid Systems V - Fifth International Hybrid Systems Workshop*, LNCS, Springer-Verlag, 1998.

[11] M. Rönkkö and K. Sere. *Refinement and Continuous Behaviour.* To appear in the Proceedings of Hybrid Systems: Computation and Control, Second International Workshop HSCC'99, Nijmegen, The Netherlands, LNCS 1569, Springer-Verlag, 1999.

[12] Li Xuandong, Zheng Tao, Hou Jianmin, Zhao Jianhua, and Zheng Guoliang. *Hybrid Regular Expressions.* In Hybrid Systems: Computation and Control LNCS 1386, pp. 384-399, Springer-Verlag, 1998.

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**