INSTITUT FÜR INFORMATIK UND PRAKTISCHE MATHEMATIK

LEHRSTUHL FÜR SOFTWARETECHNOLOGIE

Verification of Hybrid Systems: Formalization and Proof Rules in PVS

Erika Ábrahám-Mumm
Ulrich Hannemann
Martin Steffen

Bericht Nr. TR-ST-01-1
29. January 2001

CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL

Verification of Hybrid Systems: Formalization and Proof Rules in PVS*

29. January 2001

Erika Ábrahám-Mumm[1], Ulrich Hannemann[2], and Martin Steffen[1]

[1] Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel
Preußerstraße 1–9, D-24105 Kiel, Deutschland
{eab|ms}@informatik.uni-kiel.de
[2] Computing Science Department, University of Nijmegen
P.O. Box 9010, NL - 6500 GL Nijmegen, The Netherlands
ulrichh@cs.kun.nl

Abstract. Combining discrete state-machines with continuous behavior, hybrid systems are a well-established mathematical model for discrete systems acting in a continuous environment. As a priori *infinite* state systems, their computational properties are undecidable in the general model and the main line of research concentrates on model checking of finite abstractions of restricted subclasses of the general model. In our work, we use *deductive methods*, falling back upon the general-purpose theorem prover PVS.
To do so we extend the classical approach for the verification of state-based programs by developing an inductive proof method to deal with the parallel composition of hybrid systems. It covers shared variable communication, label-synchronization, and especially the common continuous activities in the parallel composition of hybrid automata. Besides hybrid systems and their parallel composition, we formalized their operational step semantics and a number of proof-rules within PVS, for one of which we give also a rigorous completeness proof. Moreover, the theory is applied to the verification of a number of examples.

Keywords: hybrid systems, deductive methods, machine-assisted verification.

1 Introduction

Combining discrete state-machines with continuous behavior, hybrid systems [1] have been successfully used to model a large number of applications in areas such as real-time software, embedded systems, and others. Basically, it is a state-based formalism augmented by real-valued variables which may continuously evolve over time. The discrete behavior is given as a labeled transition system, typically in guarded-command notation, allowing shared-variable communication and synchronization over transition

* The work was supported by the technology foundation STW, project EIF 3959, "Formal Design of Industrial Safety-Critical Systems" and further by the German Research Council (DFG) within the special program KONDISK under grant LA 1012/5-1.

labels. The continuous behavior — the activities, as it is called — is typically specified per control-state by differential (in-)equations.

By its continuous part, hybrid automata are a priori *infinite* state systems. Moreover, their computational properties are undecidable in the general model (this is already true for timed-automata, an important subclass). Depending on various restrictions on the form of the invariants, the guards, the activities, etc., a score of variants and simplifications of the general model have been investigated, especially to obtain decidable and automatically checkable subclasses of the general definition (cf. for instance [1] [2] [18] [12] [24]). The main line of research concentrated on model checking of finite abstractions of restricted subclasses of the general model. Besides the drawback of limited expressive power, fully-automatic approaches suffer from the usual state-space explosion problem, when dealing with the parallel composition of subsystems.

Hence in our work, we pursue an alternative route, using *deductive methods* and falling back upon a general-purpose theorem prover. To assure rigorous formal reasoning, we employ the interactive theorem prover PVS [23], based on higher-order logic, extensive libraries of data-structures and theories, powerful strategies to assist in routine verification tasks, and modularization facilities.

A classical approach for the verification of state-based programs is that of *inductive assertions:* to prove invariance of a property for all reachable states, it suffices to give an inductive proof, i.e., to prove initial satisfaction and preservation under computational steps. To cope with the verification of parallel systems, it is advantageous to exploit the system's parallel structure (cf. for instance [7] for an extensive monograph on the topic). In the present paper we develop an inductive proof method to deal with the parallel composition of hybrid systems, which we prove to be complete. The method covers the shared variable communication, label-synchronization, and especially the common continuous activities in the parallel composition of hybrid automata. Besides hybrid systems and their parallel composition, we formalize the operational step semantics and a number of proof-rules within PVS, and apply the theory to the verification of a number of examples.

The rest of the paper is organized as follows. We start in Section 2, defining hybrid systems and their transition semantics. Section 3 develops a proof method for verifying safety properties of hybrid systems, based on assertion networks. After defining the parallel composition of hybrid systems (Section 4), we generalize the proof rules in Sections 5. Finally, in Section 6 we conclude with related and future work. The library of PVS-theories formalizing the hybrid system model, together with the proof methods and the examples is available via `http://www.informatik.uni-kiel.de/~eab`.

2 Hybrid systems

Hybrid systems [1] are a well-known formal model for discrete systems acting in a continuous environment. The system's discrete part is represented as a finite set of locations or modes Loc, connected by discrete transitions or edges. The continuous part is given by a finite set Var of variables ranging over the real numbers $\mathbb{R}$. To be able to reason about the parallel composition of hybrid systems, the variable set Var of a hybrid system is a finite subset of a common countably-infinite variable set Var_g. A mapping $\nu : Var \to \mathbb{R}$ of variables to real values is called a *valuation*; the set of all valuations is denoted by V. A location-valuation pair $\sigma = (l, \nu) \in Loc \times V$ constitutes a *state* of a hybrid system. Let $\Sigma = Loc \times V$ denote the set of all states. A state set $Ini \subseteq \Sigma$ characterizes the initial states of the system.

As states consist of a discrete and a continuous part, so do the transitions of a hybrid system. Discrete change of state is captured by the edges of the graph: leading from one location to another, a transition changes the discrete part of the state; besides that, in going from one location to the next, it may alter non-deterministically the values of the variables.

To cater for synchronization between parallel components, the edges come decorated with a synchronization label from a finite label set Lab. The set of labels contains a specific stutter label τ denoting internal moves, not eligible for synchronization. Each location l is assumed to be able to perform a *stutter* transition labelled by τ. Such a transition stands, as usual, for a "do-nothing" step and denotes that other hybrid systems involved in the parallel composition take some discrete transitions. To distinguish between variables the component has under its control in a stutter transition and those it cannot actively influence, the variable set is split into *control* and *non-control* variables. The distinction is drawn per location by a function $Con : Loc \to 2^{Var}$. Stutter transitions leave the valuations for control variables of the given location unchanged, while putting no restriction on the effect concerning the non-control variables, as they are considered as being influenced solely by the outside.

For the continuous part, the values of the variables may evolve over time, where the corresponding behavior is described, per location, by a set of *activities.* An activity is a continuous function, describing the variables' change starting from the moment the location is entered. Since the specific entrance point in time should not influence the behavior relative to that moment, the set of activities for a location is required to be insensitive against shift in time, or time-invariant. Let $\mathcal{F}$ denote the set of all continuous functions in $\mathbb{R}^{\geq 0} \to V$. A set $F \subseteq \mathcal{F}$ of activities is called *time-invariant*, if for all $f \in F$ and $t \in \mathbb{R}^{\geq 0}$, also $f + t \in F$, where $f + t$ denotes the function which assigns to each $t' \in \mathbb{R}^{\geq 0}$ the value $f(t + t')$. An *invariant* finally is attributed to each location, i.e., a predicate over the set of valuations V, where the system is allowed to enter or stay in a location only as long as the invariant evaluates to true.

Before giving the formal definition of a hybrid system, let us fix some notations. We write $f|_{A'} : A' \to B$ for the restriction of a function $f : A \to B$ to a sub-domain $A' \subseteq A$; the same notation is used for the extension of the restiction operator to sets of functions, as well. For $f \in \mathbb{R}^{\geq 0} \to V$ and $x \in Var$, we denote by f^x the function in $\mathbb{R}^{\geq 0} \to \mathbb{R}$ such that $f^x(t) = f(t)(x)$ for all $t \in \mathbb{R}^{\geq 0}$. We call a function $f \in \mathbb{R}^{\geq 0} \to V$ continuous, if for all $x \in Var$, f^x is continuous. The following definition corresponds to the one encoded in PVS; to avoid overly baroque notation, we elide type declarations present in PVS within the definitions in the paper, whenever the type can unambigously be inferred from the context. This convention applies to all the following definitions.

Definition 1 (Hybrid system). *A* hybrid system *H is a tuple $(Loc, Var, Con, Ini, Lab, Edg, Act, Inv)$, where Loc is a finite, non-empty set of* locations *and Var a finite, non-empty set of* variables. *The function $Con \in Loc \to 2^{Var}$ defines the* control variables *in each state, the set* $Ini \subseteq \Sigma = Loc \times V$ *the* initial states. *The* transitions *are given by* $Edg \subseteq Loc \times Lab \times (2^V, V \to 2^V) \times Loc$, *where* Lab *denotes a finite set of* labels *containing the stutter label* τ. *For all* $l \in Loc$ *there is a stutter transition* $(l, \tau, (\phi, f), l) \in Edg$ *such that* $\phi = V$ *and* $f(\nu) = \{\nu' \mid \nu|_{Con(l)} = \nu'|_{Con(l)}\}$. *The* activities *are given by* $Act : Loc \to 2^{\mathcal{F}}$ *such that* $Act(l)$ *is time-invariant for each location* $l \in Loc$. *The function* $Inv : Loc \to 2^V$ *specifies the* invariants.

For a discrete transition $(l_1, \alpha, (\phi, f), l_2) \in Edg$, $\phi \subseteq V$ is called the *guard* and $f : V \to 2^V$ its *effect*. Depending on various restrictions on the form of the invariants, the guards, the activities etc., a score of variants and simplifications of this model have been investigated, especially to obtain decidable and automatically checkable subclasses of the general definition (cf. for instance [1] [2] [18] [12] [24]). As in this paper we are concerned with formulating a proof method within a deductive framework, we will stick to the general definition.

Let's illustrate the definitions so far on a simple, well-tried example, the thermostat.

Example 2 (Thermostat). The temperature x of a room is controlled by a thermostat, which continuously senses the temperature and turns a heater on and off if the threshold values x^{min} and x^{max} are reached, where $x^{min} < x^{max}$ and $x^{min}, x^{max} \in \mathbb{R}^{>0}$. When the heater is off, the temperature decreases according to the function $x(t) = x_0 e^{-Kt}$, where x_0 is the initial temperature, t the time, and $K \in \mathbb{R}^{>0}$ a room constant. With the heater turned on, the temperature follows the function $x(t) = (x_0 - h)e^{-Kt} + h$, where $h > x^{min} + x^{max}$ is a real-valued constant which depends on the power of the heater. The initial temperature is x^{max} degrees and the heater is off initially. The two modes of the thermostat are represented by two control locations, l_{off} and l_{on}. In both locations, the time-invariant activity sets are specified as differential equations. The invariants represent the conditions concerning the extremal temperatures for each location. Together with the guards on the edges between the two locations, these invariants force the system two switch between the two modes iff. the extremal temperatures are reached. Two variables y and z serve to record the duration of the time spent in the heating and the non-heating mode. The resulting hybrid system is shown in Figure 1. By convention, trivial components of an edge $(l, \alpha, (\phi, f), l')$, i.e., $\alpha = \tau$, $\phi = true$, or $f = Id$ are not shown, and neither are stutter transitions. The same simplification is done for trivial invariants in locations.

As mentioned before, a system's state can change in two ways: either by time delay or by discrete transitions. Hence there are two kinds of transitions between states: *time steps,* written $\to^{f,t}$, describe the evolution of the values of the variables in a given location and according to an activity in that location. An instantaneous, discrete step, written $\to^{\alpha}$, follows an edge $(l_1, \alpha, (\phi, f), l_2)$ of the system, thereby moving from location l_1 to l_2 and possibly changing the values of the variables according to (ϕ, f). For both relations, control may stay in a location (i.e., time can progress in a location), resp. enter a location through a discrete state change, only if the invariant is not violated. The resulting transition semantics is summarized in the following definition.

Definition 3 (Step semantics). *Let* $H = (Loc, Var, Con, Ini, Lab, Edg, Act, Inv)$ *be a hybrid system with set of states* $\Sigma = Loc \times V$. *For an activity* f, *a non-negative*

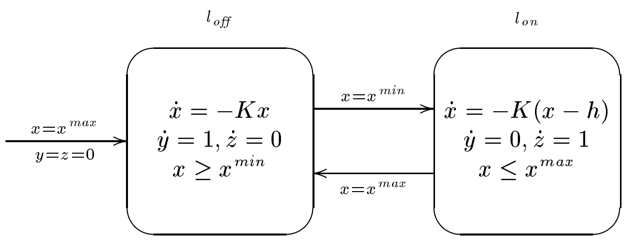


Fig. 1. Thermostat

time delay t, and a discrete transition $(l_1, \alpha, (\phi, f), l_2)$, the discrete step relation $\rightarrow^{\alpha} \subseteq \Sigma \times \Sigma$ and the time step relation $\rightarrow^{f,t} \subseteq \Sigma \times \Sigma$ are given by the following two rules:

$$\frac{\nu_2 \in f(\nu_1) \quad \nu_1 \in \mathit{Inv}(l_1) \quad \nu_2 \in \mathit{Inv}(l_2) \qquad (l_1, \alpha, (\phi, f), l_2) \in \mathit{Edg} \quad \phi(\nu_1) = \mathit{true}}{(l_1, \nu_1) \rightarrow^{\alpha} (l_2, \nu_2)}$$

$$\frac{f(0) = \nu_1 \quad f(t) = \nu_2 \quad \forall 0 \leq t' \leq t.\ f(t') \in \mathit{Inv}(l) \qquad t \in \mathbb{R}^{\geq 0} \quad f \in \mathit{Act}(l)}{(l, \nu_1) \rightarrow^{f,t} (l, \nu_2)}$$

The one-step *relation $\rightarrow$ is defined by $\rightarrow^{\alpha} \cup \rightarrow^{f,t}$. A* run *of the hybrid system H is a (finite or infinite) sequence $\rho = \sigma_0 \rightarrow \sigma_1 \rightarrow \sigma_2 \rightarrow \cdots$, with $\sigma_0 = (l_0, \nu_0) \in \mathit{Ini}$ and $\nu_0 \in \mathit{Inv}(l_0)$. We denote the set of runs of H by $[\![H]\!]$. A state $\sigma \in \Sigma$ is* reachable *in H, if there exists a run $\rho = \sigma_0 \rightarrow \sigma_1 \rightarrow \sigma_2 \rightarrow \cdots \rightarrow \sigma_n$ of H with $\sigma_n = \sigma$. We write $R(H)$ for the set of all reachable states of H.*

We use $\rightarrow^{n}$, $\rightarrow^{*}$, and $\rightarrow^{+}$ to denote respectively the n-step relation, the reflexive-transitive closure, and the transitive closure of the one step relation.

The semantics allows a system to evolve by arbitrarily mixing discrete and continuous steps. It will sometimes be convenient later to assume a more constrained form of trajectories, disallowing time steps of duration zero and disallowing two consecutive time steps. We will call such trajectories to be in *normal form.* Since discrete stutter steps are always possible, each trajectory can be transformed into one in normal form. This is expressed in the following lemma.

Lemma 4. $\rightarrow^{*} = (\rightarrow^{\alpha}\rightarrow^{f,t>0} \cup \rightarrow^{\alpha})^{*}$.

Proof. The direction $\rightarrow^{*} \supseteq (\rightarrow^{\alpha}\rightarrow^{f,t>0} \cup \rightarrow^{\alpha})^{*}$ is immediate. For the reverse direction, assume $\sigma_1 \rightarrow^{*} \sigma_2$ and proceed by induction on the number n of $\rightarrow$-steps. The base case for $n = 0$ is immediate. For the induction step, assume $\sigma_1 \rightarrow^{*} \sigma_3 \rightarrow \sigma_2$ for some state σ_3. Independant of whether the last step $\sigma_3 \rightarrow \sigma_2$ is a time or a discrete step, the invariant holds at σ_3, therefore a (discrete) stutter step $\sigma_3 \rightarrow^{\tau} \sigma_3$ is possible.

Using the induction hypothesis thus gives $\sigma_1(\rightarrow^{\alpha}\rightarrow^{f,t>0} \cup \rightarrow^{\alpha})^*\sigma_3 \rightarrow^{\tau} \sigma_3 \rightarrow \sigma_2$, from which the result is immediate. $\square$

Before giving an example, let us fix some conventions to specify the components of the hybrid system. The standard way to describe the activities is as solutions of differential equations and differential inclusions. A differential equation is written $\dot{\boldsymbol{x}} = g(\boldsymbol{x})$, where $\boldsymbol{x} = (x_1, \ldots, x_n)$ are variables from the variable set *Var* of a given hybrid system, and with g a function from V to V. The solution set of a differential equation $\dot{\boldsymbol{x}} = g(\boldsymbol{x})$ is the set of all functions $f \in \mathcal{F}$ such that for all $i \in \{1, \ldots, n\}$, f^{x_i} is differentiable and $\dot{f}^{x_i}(t) = g(f(t))(x_i)$ for all $t \in \mathbb{R}^{>0}$. Similarly, a differential inclusion is an expression of the form $\dot{\boldsymbol{x}} \in g(\boldsymbol{x})$, where g is a function from V to 2^V. The solution set of a differential inclusion $\dot{\boldsymbol{x}} \in g(\boldsymbol{x})$ is the set of all functions $f \in \mathcal{F}$ such that for all $i \in \{1, \ldots, n\}$, f^{x_i} is differentiable and $\dot{f}^{x_i}(t) \in g(f(t))(x_i)$ for all $t \in \mathbb{R}^{>0}$. We will write subsets of valuations V, like the invariants of the locations, in form of boolean predicates $\varphi : V \rightarrow Bool$. Such a predicate φ defines the set of all valuations ν, such that $\varphi(\nu) = true$. In such formulas we write short x for the evaluation $\nu(x)$, where $x \in Var$. In a transition relation (ϕ, f), the non-deterministic change of valuation associated with an edge of the system expressed by f will be written in the form of a simultaneous, non-deterministic assignment $x_1, \ldots, x_n := g_1, \ldots, g_n$, where $x_1, \ldots, x_n \in Var$, and $g_1, \ldots, g_n$ are set-valued functions from V to $2^{\mathbb{R}}$. The relation f is then defined as the set of all valuation pairs $(\nu, \nu') \in V^2$ such that $\nu'(x_i) \in g_i(\nu)$ for all $i = 1, \ldots, n$, and $\nu(y) = \nu'(y)$ for all $y \in Var \setminus \{x_1, \ldots, x_n\}$.

3 Inductive assertional proofs

Our approch and formalization to analyze the behaviour of hybrid systems is based on Floyd's *inductive assertion method* [8]. In this classical state-based verification method one associates an assertion, i.e., a predicate over the current values of variables, with each control location of the underlying program. This gives a finite number of verification conditions to check for proving the given correctness criteria of that program. While originally developed in the context of sequential programs, the inductive assertion method serves also as fundamental technique in the analysis of concurrent programs [7]. We extend the inductive assertion method to hybrid systems.

Let $(Loc, Var, Con, Ini, Lab, Edg, Act, Inv)$ be a hybrid system. An *assertion* on a location l is a boolean predicate over V, or equivalently a subset of V, and an *assertion network* is a subset of the global set $\Sigma = Loc \times V$ of states. For a given assertion network Q of H and a location l, let the assertion $Q_l \subseteq V$ be defined by $Q_l = \{\nu \mid (l, \nu) \in Q\}$, i.e., $\nu \in Q_l$ iff. $(l, \nu) \in Q$. Considering subsets of states as predicates on or properties of the states, we say Q_l holds for a valuation ν, in case $\nu \in Q_l$, and correspondingly for states and assertion networks. By the same token, we will speak of an assertion network implying a property etc. In connection with the system's transition semantics, an assertion network is *invariant,* if it holds for all reachable states, it is called *inductive,* if it holds for the initial states and is preserved under the transition relation.

Definition 5. *An assertion network Q of H is called* inductive, *if*

1. $\{(l,\nu) \in \mathit{Ini} \mid \nu \in \mathit{Inv}(l)\} \subseteq Q$,
2. *if* $\sigma \in Q$ *and* $\sigma \to^{\alpha} \sigma'$, *then* $\sigma' \in Q$, *and*
3. *if* $\sigma \in Q$ *and* $\sigma \to^{f,t} \sigma'$, *then* $\sigma' \in Q$,

for arbitrary states σ *and* σ' *from* Σ. *We call* Q *an* invariant *of* H, *if* $R(H) \subseteq Q$.

Obviously, each inductive network is invariant, while the converse will, in general, not hold. Therefore, when interested in verifying a property φ to hold for all reachable states, one can do so by finding a stronger invariant, i.e., an inductive assertion network Q which implies φ. This proof principle, known as *inductive assertion method,* is summarized in the following rule:

$$\frac{Q \to \varphi \qquad Q \text{ inductive for } H}{R(H) \to \varphi}\ \textsc{Ind}$$

Lemma 6. *The inductive assertion method given by rule* IND *is sound and complete.*

Proof. For soundness, let Q be an inductive assertion network of H which implies φ and $\sigma \in R(H)$ a reachable state of H. Then there exists a run $\rho = \sigma_0 \to \sigma_1 \to \ldots \to \sigma_n$ of H such that $\sigma_n = \sigma$. By induction on the length of ρ follows immediately by the definition of the inductiveness that $\sigma \in Q$, and thus also $\varphi(\sigma)$.

For completeness in the reverse direction, let φ be an invariant of H, i.e., $R(H) \to \varphi$. Set $Q = R(H)$. Since φ is an invariant, $Q \subseteq \varphi$, as required by the first premise of rule IND. □

We illustrate the approach, continuing with the thermostat example.

Example 7 (Thermostat). The thermostat of Example 2 is expected to keep the temperature between x^{min} and x^{max} degrees. Additionally, we show that at any point in time the sum of duration of the heating periods is less or equal than the overall non-heating time, provided that $h > x^{min} + x^{max}$. The duration of the two periods is recorded in the two variables y and z. Both requirements are expressed by the following predicate:

$$\varphi = x^{min} \leq x \leq x^{max} \wedge y - z \geq 0.$$

We define the network Q as follows:

$$\begin{aligned} Q_{l_{\mathit{off}}} &= x^{min} \leq x \leq x^{max} \\ &\qquad \wedge y - z \geq -\frac{1}{K} \ln \frac{x}{x^{max}}, \\ Q_{l_{on}} &= x^{min} \leq x \leq x^{max} \\ &\qquad \wedge y - z \geq -\frac{1}{K} \ln \frac{x^{min}(x^{min} - h)}{x^{max}(x - h)}. \end{aligned}$$

Since logarithms and exponentials are not predefined in PVS, we introduced axiomatically a number of algebraic and mathematical facts about these.

4 The parallel composition of hybrid systems

Complex systems are often built from smaller components working in parallel. Before we can start to reason about the composition of systems, we need to introduce their parallel composition [1]. The parallel composition of two hybrid systems H_1 and H_2 is given by a standard product construction and written as $H_1 \times H_2$. Locations are paired and the set of variables combined. The two partners can take a common discrete step, either by synchronizing on the same label, or in that one of the contributors performs a discrete non-synchronizing transition while its partner stutters. Besides synchronizing on the label in a common discrete step, the conjunction of the actions on the variables is taken, i.e., a common step is possible only if both guards are true and if the outcome on the variables coincides. On variables it does not control, a component cannot block non-synchronizing transitions of its partner, since stutter transitions, available at each location, don't restrict the behavior of non-controlled variables. On control variables, on the other hand, stuttering is allowed only without changing the variables' values. Time transitions of the composed systems are time transitions in both systems, i.e., the activities of the composed system, restricted to the local variables, are activities of the component systems. Invariants of the composition finally are conjunctions of the component invariants.

Definition 8 (Parallel composition). *Let H_1 and H_2 be two hybrid systems of the forms $(Loc_i, Var_i, Con_i, Ini_i, Lab_i, Edg_i, Act_i, Inv_i)$. The* product *$H_1 \times H_2$ is the hybrid system $H = (Loc_1 \times Loc_2, Var_1 \cup Var_2, Con, Ini, Lab_1 \cup Lab_2, Edg, Act, Inv)$ such that for all $l_1, l_1' \in Loc_1$, $l_2, l_2' \in Loc_2$, $\alpha \in Lab$, $\phi \subseteq V$, $f : V \to 2^V$ and $g \in \mathcal{F}$:*

1. *$((l_1, l_2), \nu) \in Ini$ iff. $(l_i, \nu|_{Var_i}) \in Ini_i$, for $i = 1, 2$;*
2. *$Con((l_1, l_2)) = Con_1(l_1) \cup Con_2(l_2)$;*
3. *$(l_1, l_2) \longrightarrow^{\alpha}_{(\phi,f)} (l_1', l_2') \in Edg$, iff. there exist $l_i \longrightarrow^{\alpha_i}_{(\phi_i,f_i)} l_i' \in Edg_i$, such that*
   (a) $\alpha = \alpha_1 = \alpha_2$, or $\alpha_1 = \alpha \notin Lab_2$ and $\alpha_2 = \tau$, or $\alpha_1 = \tau$ and $\alpha = \alpha_2 \notin Lab_1$,
   (b) $\phi(\nu) = \phi_1(\nu|_{Var_1}) \wedge \phi_2(\nu|_{Var_2})$, and
   (c) $\nu' \in f(\nu)$, iff. $\nu'|_{Var_1} \in f_1(\nu|_{Var_1})$ and $\nu'|_{Var_2} \in f_2(\nu|_{Var_2})$;
4. *$g \in Act((l_1, l_2))$, iff. for both $i = 1$ and $i = 2$, there exist $g_i \in Act_i(l_i)$, such that $\lambda t.g(t)|_{Var_i} = g_i$;*
5. *$Inv((l_1, l_2))|_{Var_i} = Inv_i(l_i)$ for $i = 1, 2$.*

Note that by construction the set of activities $Act((l_1, l_2))$ for a composed location is time invariant, since $Act_1(l_1)$ and $Act_2(l_2)$ are. It is routine albeit tedious to show that parallel composition is associative and commutative. For a parallel composition $H_1 \times \ldots \times H_n$ with $n > 0$ and $j \in \{1, \ldots, n\}$, we call the composition system without H_j the *context* of H_j. Let Σ_H denote the state space of H. For the product system $H = H_1 \times H_2$, and a state $\sigma = ((l_1, l_2), \nu)$ of H, we write $\sigma\downarrow_{H_1} = (l_1, \nu|_{Var_1})$ and $\sigma\downarrow_{H_2} = (l_2, \nu|_{Var_2})$ for the projections on the respective components; we will use the same notation for sets of states. For a run $\rho = \sigma_0 \to_0 \sigma_1 \to_1 \sigma_2 \to_2 \cdots$ of the product system H, the projection $\rho\downarrow_{H_1}$ is the sequence $\sigma_0' \to_0' \sigma_1' \to_1' \sigma_2' \to_2' \cdots$, where for all $i = 0, 1, 2, \ldots$, $\sigma_i' = \sigma_i\downarrow_{H_1}$, and if $\to_i$ is a time-step of the form $\to^{f_i,t_i}$, then $\to_i'$ is a time-step $\to^{f_i',t_i}$ where f_i' is defined by $f_i'(t) = f_i(t)|_{Var_1}$ for all $t \in \mathbb{R}^{\geq 0}$, and if $\rightarrow_i$ is a discrete-step $\rightarrow^{\alpha_i}$ then $\rightarrow'_i$ is also a discrete-step $\rightarrow^{\alpha'_i}$ where $\alpha'_i = \alpha_i$ if $\alpha_i \in Lab_1$, and $\alpha'_i = \tau$ otherwise. The projection $\rho \downarrow_{H_2}$ is defined analogous.

A basic property of the product system is that all runs of the product projected to one of the component systems are runs of that component system:

Lemma 9. *Let $H = H_1 \times H_2$ be the parallel composition of two hybrid systems H_1 and H_2. Then $[\![H]\!]\downarrow_{H_i} \subseteq [\![H_i]\!]$ and $R(H)\downarrow_{H_i} \subseteq R(H_i)$, for $i = 1, 2$.*

Proof. Straightforward, albeit tedious. □

Example 10 (Thermostat revisited). We extend Example 2, considering a finite number of heating systems $H_1, \ldots, H_n$ running in parallel and sharing a common fuel tank. An additional variable u_i for each thermostat records its fuel consumption; it is a control variable in both the on- and the off-locations, with values $u_i = 0$ in location l_{off} and $u_i = c_i$ in l_{on}, and where the constant c_i represents the consumption per time unit. We abbreviate the system's overall fuel consumption $\sum_{i=1}^{n} u_i$ by the variable u. Heating is possible only as long as the tank is not empty and a heater can be turned on only if there's a certain amount of fuel in the tank, represented by the threshold value $\epsilon \in \mathbb{R}^{>0}$. The discrete transitions and the invariants from Example 2 are modified accordingly.

The fuel tank H_{fuel} is emptied by the heaters in their on-mode, and can be refilled with a constant rate d to keep the filling level v between Vol_1 and Vol_2 with $\mathrm{Vol}_1, \mathrm{Vol}_2 \in \mathbb{R}^{>0}$ and $\epsilon < \mathrm{Vol}_1 < \mathrm{Vol}_2$. Initially, the tank is full at level Vol_2. We assume that the filling rate d exceeds the sum $\sum_{i=1}^{n} c_i$ of the outflows in case of maximal heating, i.e., when all heaters are on. The resulting composed system is shown in Figure 2.

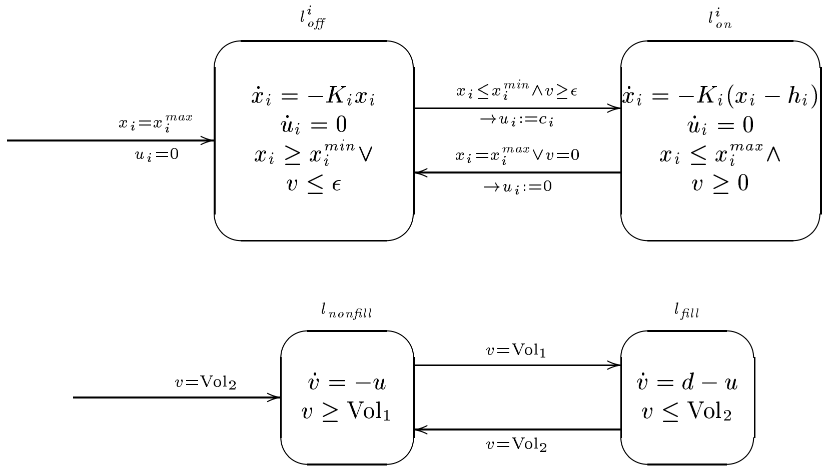


Fig. 2. Thermostats with common fuel tank: $H_1 \times \ldots H_n \times H_{fuel}$

5 Inductive assertional proofs for parallel composition

A first approach to the verification of the parallel composition of hybrid systems is immediately given by the observation that the parallel composition of two hybrid systems as defined in Definition 8 results in a hybrid system for which one can construct an inductive assertion network again. The number of resulting verification obligations turns out to reflect the state-explosion problem.

The basic idea for an improvement over the plain product of assertions for the classical programming concepts (cf. to [7] for an exhaustive treatment) is a two-level approach, where first *local* assertion networks are checked for local consistency, and then some global consistency test relates these local networks, reducing the amount of verification conditions. In the sequel, we improve the inductive assertion method for hybrid systems following those very ideas. An important technique in this extension of the inductive method is the introduction of new, otherwise unused variables, which allow to speak about the peer processes of a component. These variable are commonly called *auxiliary variables*. The extension of a system by introducing (finitely many) auxiliary variables is called *augmentation.* Auxiliary variables allow the formulation of a sound and complete proof rule for the parallel composition of hybrid systems which we apply to the above example.

5.1 Augmentation

As auxiliary variables are added for the sole purpose of verification, their addition must not influence the system's behaviour in any way, i.e., the unaugmented part of the system must be left untouched. The additional variables may be control variables (per location) but the augmentation must not change the "control"-status of the old variables. Furthermore, projected onto the domain belonging to the original set of variables, the state space and the set of initial states remains unchanged (the set of locations is not altered, anyway). As for the discrete transition, each one in the original system is mirrored in the augmented one, and vice versa. Especially, the transition relation for each edge must result in the same change on the old variables and conversely as well: each transition from the original system must be reflected by some of the augmented one. Analogously, augmenting the system with auxiliary variables must not constrain its continuous behavior: each activity on the original variables must have a counterpart in the extended system and vice versa. To be compliant with the definition of a hybrid system, the new set of activities in each location must be closed under time-shift, i.e., be time-invariant. The invariants, finally, must not restrain the new variables at all.

Definition 11 (Augmented hybrid system). *Given a set of global variables Var_g, a hybrid system $H = (Loc, Var, Con, Ini, Lab, Edg, Act, Inv)$, and a set of fresh auxiliary variables $Var_{aux} \subseteq Var_g$, i.e., $Var_{aux} \cap Var = \emptyset$. We call H' an* augmentation *of H with Var_{aux}, if $H' = (Loc, Var', Con', Ini', Lab, Edg', Act', Inv')$ with $Var' = Var \dot\cup Var_{aux}$ and if the following conditions hold:*

1. *$Con(l) \subseteq Con'(l)$ and $Con'(l) \setminus Con(l) \subseteq Var_{aux}$, for all locations $l \in Loc$;*
2. *$Ini = \{(l, \nu) \mid \exists (l, \nu') \in Ini'.\ \nu = \nu'|_{Var}\}$;*

3. *$Inv'(l) = \{\nu' \mid \nu'|_{Var} \in Inv(l)\}$, for all locations $l \in Loc$;*
4. (a) *for all $(l_1, a, (\phi, f), l_2) \in Edg$ there exists a $(l_1, a, (\phi', f'), l_2) \in Edg'$ such that for all $\nu' \in V'$: $\phi(\nu'|_{Var})$ implies $\phi'(\nu')$ as well as $f(\nu'|_{Var}) \subseteq f'(\nu')|_{Var}$;*
   (b) *for all $(l_1, a, (\phi', f'), l_2) \in Edg'$ exists $(l_1, a, (\phi, f), l_2) \in Edg$ such that $\phi'|_{Var} \subseteq \phi$, and for all valuations $\nu' \in V'$, $f'(\nu')|_{Var} \subseteq f(\nu'|_{Var})$;*
   (c) *for all locations $l \in Loc$ there is a stutter transition $(l, \tau, (\phi', f'), l) \in Edg'$ such that $\phi' = V'$ and $f = \lambda\nu' : V'.\lambda\hat{\nu}' : V'.\hat{\nu}'|_{Con'(l)} = \nu'|_{Con'(l)}$;*
5. *for all locations $l \in Loc$*
   (a) *$Act'(l)$ is time-invariant;*
   (b) *for all $f \in Act(l)$ and $\nu' \in V'$ with $\nu'|_{Var} = f(0)$, there exists $f' \in Act'(l)$ such that $f = \lambda t.f'(t)|_{Var}$;*
   (c) *for all $f' \in Act'(l)$, there exists $f \in Act(l)$ such that $f = \lambda t.f'(t)|_{Var}$.*

In the following, we will write $H' \geq H$, when H' is an augmentation of H. As the control flow and the activities for variables of H are not influenced by the auxiliary variables, the set of reachable states of H' restricted to Var in the valuation component equals the reachable states of the original system.

Lemma 12. *Let H be a hybrid system with variable set Var and H' an augmentation of H by auxiliary variables Var_{aux}. Then $R(H')\downarrow_{Var} = R(H)$.*

Proof. There are two directions to show. For $R(H) \subseteq R(H')\downarrow_{Var}$, induce over the assertion $Q(\sigma) = \exists\sigma' \in R(H').\ \sigma'\downarrow_H = \sigma$. For the initial states, assume some arbitrary $\sigma_0 = (l_0, \nu_0) \in Ini$ with $\nu_0 \in Inv(l_0)$. By the requirement on Ini', there exists a state $\sigma'_0 = (l_0, \nu'_0) \in Ini'$, such that $\nu'_0|_{Var} = \nu_0$. By the condition on the invariants, $\nu'_0 \in Inv'(l_0)$, hence $(l_0, \nu'_0) \in R(H')$, and therefore $(l_0, \nu'_0|_{Var}) \in R(H')\downarrow_H$, as required. For preservation under discrete steps, assume $\sigma = (l, \nu) \in Q$, and $\sigma \rightarrow^a \hat{\sigma} = (\hat{l}, \hat{\nu})$ for some label $a \in Lab$, some locations l and $\hat{l}$ from Loc, and two valuations ν and $\hat{\nu}$ from $Var \rightarrow \mathbb{R}$. By definition of the step semantics (Definition 3), there exists an edge $(l, a, (\phi, f), \hat{l}) \in Edg$ such that $\phi(\nu)$ holds, that $\hat{\nu} \in f(\nu)$, and furthermore $\nu \in Inv(l)$ as well as $\hat{\nu} \in Inv(\hat{l})$. Since $\sigma \in Q$, there exists some $\sigma' \in R(H')$ with $\sigma'\downarrow_H = \sigma$, i.e., $\sigma' = (l, \nu')$ with $\nu'|_{Var} = \nu$. By the definition of augmentation, $\nu' \in Inv'(l)$ and furthermore, there exists an edge $(l, a, (\phi', f'), \hat{l}) \in Edg'$ such that $\phi'(\nu')$ holds and additionally, since $f'(\nu')|_{Var} \supseteq f(\nu'|_{Var}) = f(\nu)$, there exists a $\hat{\nu}' \in f'(\nu')$ with $\hat{\nu}'|_{Var} = \hat{\nu}$. By Definition 11 once more, this implies $\hat{\nu}' \in Inv'(\hat{l})$, whence by Definition 3 $\sigma = (l, \nu') \rightarrow^a (\hat{l}, \hat{\nu}') = \hat{\sigma}'$ with $\hat{\sigma}'\downarrow_H = \hat{\sigma} \in R(H)$, i.e., $\hat{\sigma} \in Q$, as required.

For preservation under continuous steps, assume some state $\sigma = (l, \nu) \in Q$ with $\sigma \rightarrow^{f,t} \hat{\sigma} = (l, \hat{\nu})$ for some location $l \in Loc$ and valuations ν and $\hat{\nu}$ from $Var \rightarrow \mathbb{R}$. By definition of the step semantics (Definition 3), $t \geq 0$, $f \in Act(l)$. Additionally $f(0) = \nu$, $f(t) = \hat{\nu}$, and for all time-points $\tilde{t}$ from 0 until t, $f(\tilde{t}) \in Inv(l)$. Since $\sigma \in Q$, there exists some $\sigma' \in R(H')$ with $\sigma'\downarrow_H = \sigma$, i.e., $\sigma' = (l, \nu')$ with $\nu'|_{Var} = \nu$. By definition of augmentation, $f \in Act(l)$ implies that there exists an activity $f' \in Act'(l)$ such that $\lambda t.f'(t)|_{Var} = f$ and furthermore $f'(\tilde{t}) \in Inv'(l)$, whenever $0 \leq \tilde{t} \leq t$ by the condition for the invariants. Also by Definition 11(5b), we get for the startpoint at $t = 0$, that $f'(0) = \nu'$ and the endpoint $f'(t) = \hat{\nu}'$ with $\hat{\nu}'|_{Var} = \hat{\nu} = f(t)$. Hence by

Definition 3, $(l, \nu') \rightarrow^{f',t} (l, \hat{\nu}') = \hat{\sigma}'$, i.e., $\hat{\sigma}' \in R(H')$ and $\hat{\sigma}' \downarrow_H = (l, \hat{\nu}' \downarrow_{Var}) = (l, \hat{\nu}) = \hat{\sigma}$. Thus $\hat{\sigma} \in Q$ which finishes the case.

The reverse direction is shown similarly. □

We obtain as immediate corollary for the invariants of a system and its augmentation:

Corollary 13. *Assume $H' \geq H$. If φ' is an invariant of the augmentation H', then the projection $\varphi' \downarrow_H$ is an invariant of H.*

Proof. By assumption $R(H') \subseteq \varphi'$, therefore $R(H) = R(H') \downarrow_H \subseteq \varphi' \downarrow_H$, where we use the simple fact, that projection are monotone. □

Thus, a property whose satisfaction does not depend on the values for the auxiliary variables, holds for all reachable states of H', iff. it holds for all reachable states of H.

5.2 A complete proof method

Two hybrid systems running parallel are related by the time. We allow to refer to the global time in the assertions as a variable t. Referring to the global time is equivalent to having a common (auxiliary) variable in all hybrid systems running in parallel, whose initial value is 0, whose derivative is always 1, and whose is not affected by any transition. Allowing to refer to t as a pre-defined common auxiliary variable simplifies the verification.

Definition 14. *Let H_1 and H_2 be hybrid systems, $H = H_1 \times H_2$ their parallel composition, and Q_1 and Q_2 assertion networks for H_1 and H_2, respectively. We define the composition of the local assertion networks as $Q_1 \times Q_2 = \{\sigma \in \Sigma_H | \sigma \downarrow_{H_1} \in Q_1 \wedge \sigma \downarrow_{H_2} \in Q_2\}$.*

Note that $Q_1 \times Q_2$ is an assertion network of H. We also apply conjunction and implication to assertion networks as predicates if they are of the same type. So let H_1 and H_2 be two hybrid systems, $H = H_1 \times H_2$ with variable set Var their parallel composition, and $\varphi \subseteq \Sigma_H$ a predicate on the set of H's states. Then φ is an invariant of H if and only if there exists an auxiliary variable set Var_{aux}, hybrid systems H_1' and H_2', such that $H' = H_1' \times H_2'$ is an augmentation of H with Var_{aux}, and inductive assertion networks Q_1' and Q_2' of H_1' and H_2', respectively, such that $Q = \{(l, \nu) \in \Sigma_H \mid \exists (l, \nu') \in Q_1' \times Q_2'.\, \nu = \nu'|_{Var}\} \subseteq \varphi$. With these conventions, we can formulate the proof rule to deal with the parallel composition of systems.

$$\frac{H_1' \times H_2' \geq H_1 \times H_2 \qquad Q_1' \text{ inductive for } H_1' \quad Q_2' \text{ inductive for } H_2' \qquad \forall (l, \nu') : \Sigma_{H'}.\, Q_1' \times Q_2'(l, \nu') \rightarrow \varphi(l, \nu'|_{Var})}{R(H) \rightarrow \varphi} \textsc{Comp}$$

Proposition 15. *The proof rule* (COMP) *is sound and complete.*

Proof. There are two directions to show.

Case 1: Soundness
Let $\sigma = (l, \nu) \in R(H)$ be some reachable state of H. Since $H' \geq H$, there exists by Lemma 12 some state $\sigma' = (l, \nu') \in R(H')$ with $\nu = \nu'|_{Var}$. By Lemma 9 and the soundness of the inductive assertional method, $\sigma' \downarrow_{H_1} \in R(H_1') \subseteq Q_1'$ and $\sigma' \downarrow_{H_2} \in R(H_2') \subseteq Q_2'$. Therefore by construction, $\sigma \in Q_1' \times Q_2'$, and so $\sigma \in \varphi$, as required.

Case 2: Completeness
To prove completeness, we introduce three auxiliary variables shared by H_1' and H_2': one to measure the time, one to record the history of the discrete steps, and one to enforce the existence of a common activity for continuous steps. So define H_1' as the augmentation $(Loc_1, Var_1', Con_1', Ini_1', Lab_1, Edg_1', Act_1', Inv_1')$ (H_2' is given analogously) with $Var_1' = Var_1 \dot\cup \{h_t, h_\rho, h_f\}$ $Var_1'' = Var_1 \dot\cup \{h_t\}$, similarly $Var_2' = Var_2 \dot\cup \{h_t, h_\rho, h_f\}$ and $Var_2'' = Var_1 \dot\cup \{h_t\}$, and furthermore $Var' = Var_1' \cup Var_2'$ and $Var'' = Var_1'' \cup Var_2''$. By the same convention, we write V_1' for the valuations $Var_1' \to \mathbb{R}$ with typical elements $\nu_1', \hat\nu_1', \ldots$, Σ'' for the set of states $(Loc_1 \times Loc_2) \times (Var'' \to \mathbb{R})$ with typical elements σ'', $\hat\sigma''$, ..., etc. The auxiliary variable h_ρ is used to record *runs* of the system (including the history of t); it is of type of finite sequences over $\Sigma'' \times Lab_1 \times Lab_2 \times \Sigma''$.[1]

Initially, the recorded history is empty, the elapsed time is 0, and also h_f starts at 0, i.e.,

$$Ini_1' = \{(l_1, \nu_1') \mid (l_1, \nu_1'|_{Var_1}) \in Ini_1, h_\rho = \epsilon, h_t = h_f = 0\},$$

where ϵ denotes the empty sequence. For the definition of the edges Edg_1', we will use the following abbreviation: given two label sets Lab_1 and Lab_2, the predicate $\bowtie$ on $Lab_1 \times Lab_2$ —let's call it "compatible", i.e., eligible for a common step— is defined as follows: $a_1 \bowtie a_2 = \mathit{true}$ if $a_1 = a_2$, or $a_1 = \tau$ and $a_2 \in Lab_2 \setminus Lab_1$, or $a_2 = \tau$ and $a_1 \in Lab_1 \setminus Lab_2$, and false otherwise (the predicate thus corresponds to the label synchronization condition in clause (3a) of Definition 8) The edges of H_1' are defined such that they correspond to the ones of H_1 as far as locations and labels are concerned. The guard and the effect of the new edge are "augmented" to deal with the new auxiliary variables. The history-recording variable h_ρ especially is defined to extend the stored sequence by a new step, where in the post-state the variable h_t is reset to zero. In this way, h_t measures the time the control stays in the current location and these durations are kept in the history variable h_ρ. In addition to current edge's label of H_1, the step records the potential (i.e., compatible) label α_2 of the communication partner. Extending the sequence by a new element is denoted by $\circ$. The definition of Edg_1' is summarized by the following rule:

$$\frac{\begin{array}{c} l_1 \longrightarrow^{\alpha_1}_{\phi_1, f_1} \hat{l}_1 \in Edg_1 \qquad (\phi_1', f_1') \geq (\phi_1, f_1) \qquad \alpha_1 \bowtie \alpha_2 \\ \hat\nu_1' \in f_1'(\nu_1') \qquad \sigma'' \in \Sigma'' \qquad \sigma'' \downarrow_{H_1} = (l_1, \nu_1'|_{Var_1}) \qquad \hat\sigma'' \downarrow_{H_1} = (\hat{l}_1, \hat\nu_1'|_{Var_1}) \\ \hat\nu_1'(h_t) = 0 \qquad \hat\nu_1'(h_\rho) = \nu_1'(h_\rho) \circ (\sigma'', (\alpha_1, \alpha_2), \hat\sigma'') \end{array}}{l_1 \longrightarrow^{\alpha_1}_{\phi_1', f_1'} \hat{l}_1 \in Edg_1'}$$

[1] Since we assume the set of locations, the set of labels, and the set of used variables to be finite, auxiliary variables storing the history can be encoded in the model.

For the activities, we set $f'_1 \in Act'_1(l_1)$, iff there exists an $f_1 \in Act_1(l_1)$ such that $\lambda t. f'_1(t)|_{Var_1} = f_1$ and

$$\frac{\partial f'^{t}_{1}}{\partial t} = 1$$
$$\frac{\partial f'^{h_f}_{1}}{\partial t}(t) = \begin{cases} 0 & \text{if } \exists f_2 \in Act_2(l_2(h_\rho)). \forall 0 \le t' \le t.\, f_2(t')|_{Var_2} = f_1(t')|_{Var_2} \wedge \\ & \qquad Inv_2(l_2(h_\rho))(f_2(t')) \\ 1 & \text{else} \end{cases}$$

where $l_1 \in Loc_1$ is an arbitrary location from H_1 and $l_2(h_\rho)$ is the location component for H_2 stored at the last state in the recored history, i.e., if $h_\rho = \epsilon$, the value is undefined, and if $h_\rho = h'_\rho \circ (\_, \_, \_, (\sigma_1, (\alpha_1, \alpha_2), ((\hat{l}_1, \hat{l}_2), \hat{\nu})))$, then $l_2(h_\rho) = \hat{l}_2$. The auxiliary variable h_t, shared by H'_1 and H'_2, is increased by a constant rate of 1 and thus measures the sojourn at the current location. The activity set $Act'_2(l_2)$ is defined symmetrically, except setting $\partial f'^{h_f}_2 / \partial t\,(t) = -1$ in the "else"-case of f'^{h_f}. Note that $Act'_1(l_1)$ is time invariant, because $Act_2(l_2)$ is. For the invariants $Inv'_1(l_1) : V'_1 \rightarrow Bool$, the new variables are not restricted at all, i.e., we set $Inv'_1(l_1)(\nu'_1) = true$ iff. $Inv_1(l_1)(\nu'_1|_{Var_1}) = true$. It is then straightforward, albeit tedious, to check that $H'_1 \times H'_2 \geq H_1 \times H_2$.

Now let H'_1 and H'_2 be defined as above, then the key step for completeness is to show that the history variables record enough information to ensure that each state reachable separately in H'_1 and H'_2 is commonly reachable, as well, i.e.,

$$\text{if } \sigma' \downarrow_{H'_1} \in R(H'_1) \text{ and } \sigma' \downarrow_{H'_2} \in R(H'_2) \text{ then } \sigma' \in R(H'_1 \times H'_2). \tag{1}$$

To prove the implication, assume some state $\sigma' = ((l_1, l_2), \nu')$ with $\sigma'_i = (l_i, \nu'|_{Var'_i}) \in R(H'_i)$. By Lemma 4, the two projections of σ' are reachable by runs in normal form, i.e, $\sigma'^{i}_{0} \twoheadrightarrow^* \sigma' \downarrow_{H'_i}$, where we use $\twoheadrightarrow$ as abbreviation for $\rightarrow^{\alpha} \rightarrow^{f,t>0} \cup \rightarrow^{\alpha}$. More specifically there exist two runs ρ'_1 and ρ'_2 of H'_1 and H'_2 such that

$$\rho'_i = \sigma'^{i}_{0} \twoheadrightarrow_i \sigma'^{i}_{1} \twoheadrightarrow_i \sigma'^{i}_{2} \twoheadrightarrow_i \ldots \twoheadrightarrow_i \sigma'^{i}_{n^i}, \tag{2}$$

where $\sigma'^{i}_{j} = (l^i_j, \nu'^{i}_{j})$, for $i \in \{1, 2\}$ and all $0 \le j \le n^i$. Furthermore, for the two start states $\sigma'^{i}_{0} = (l^i_0, \nu'^{i}_{0}) \in Ini'_i$ and $\nu'^{i}_{0} \in Inv'_i(l^i_0)$, and the runs' last states $\sigma'^{i}_{n^i} = (l^i_{n^i}, \nu'^{i}_{n^i})$ equal $(l^i, \nu'|_{Var'_i})$.

Since H'_1 and H'_2 share the auxiliary variable $h_\rho \in Var_{aux} = Var'_1 \cap Var'_2$, the final states $\sigma'^{1}_{n^1}$ and $\sigma'^{2}_{n^2}$ assign the same value to h_ρ in their valuation component, and since by construction of Edg'_i, each discrete step is recorded in h_ρ, $n^1 = n^2 \geq 0$ $(= n$, say), i.e.,

$$\rho'_i = \sigma'^{i}_{0} \twoheadrightarrow_i \sigma'^{i}_{1} \twoheadrightarrow_i \sigma'^{i}_{2} \twoheadrightarrow_i \ldots \twoheadrightarrow_i \sigma'^{i}_{n}. \tag{3}$$

With the number of steps being the same for the two projected states, we can proceed by induction on the definition of the reachability predicate.

Base step: $n = 0$
In this case we are given $\sigma'^{i}_{0} = \sigma'^{i}_{n} \in Ini'_i$, for both $i \in \{1, 2\}$, therefore by definition of parallel composition immediately $\sigma' \in Ini'$, and thus $\sigma' \in R(H'_1 \times H'_2)$.

Induction step: $n > 0$
In this case the situation is characterized by the following diagram:

$$\begin{tikzcd}
\tilde{\sigma}_1' \arrow[r, two heads] & \sigma_1' \\
\tilde{\sigma}' \arrow[u, dotted, no head, "\dashv_{H_1'}"'] \arrow[d, dotted, no head, "\dashv_{H_2'}"'] \arrow[r, dotted, two heads] & \sigma' \arrow[u, no head, "\dashv_{H_1'}"'] \arrow[d, no head, "\dashv_{H_2'}"'] \\
\tilde{\sigma}_2' \arrow[r, two heads] & \sigma_2'
\end{tikzcd}$$

So assume $\tilde{\sigma}_i' \twoheadrightarrow \sigma_i'$, i.e., $\tilde{\sigma}_i' \rightarrow^{\alpha}\rightarrow^{f,t>0} \cup \rightarrow^{\alpha} \sigma_i'$ for both H_1' and H_2'. Since by definition of Edg_i', the clock h_t is reset to zero by both components on each discrete entering of a location, $\nu_1'(h_t) = \nu_2'(h_t) = 0$. Since additionally the common variable h_t increases at a constant rate of 1 during the stay at a location, since the value of h_t is being recorded in the history variable of each component, and since finally a time-step cannot last time zero, the processes can do either both a discrete step or else a discrete step followed by a continuous activity of equal non-zero length. This means, we can distinguish the following two subcases.

Subcase: $\tilde{\sigma}_i' \rightarrow^{\alpha} \sigma_i'$
More precisely we are given

$$\begin{tikzcd}
\tilde{\sigma}_1' \arrow[r, "\alpha_1"] & \sigma_1' \\
 & \sigma' \arrow[u, no head, "\dashv_{H_1'}"'] \arrow[d, no head, "\dashv_{H_2'}"'] \\
\tilde{\sigma}_2' \arrow[r, "\alpha_2"] & \sigma_2'
\end{tikzcd}$$

where $\alpha_1 \bowtie \alpha_2$ by the definition of Edg_i' and the fact that H_1' and H_2' share the variable h_ρ. By the definition of the step semantics (Definition 3), $\tilde{\sigma}_i' = (\tilde{l}_i, \tilde{\nu}_i')$ and $\sigma_i' = (l_i, \nu_i')$, with $(\tilde{l}_i, \alpha_i, (\phi_i', f_i'), l_i) \in Edg_i'$, and furthermore $\phi_i'(\tilde{\nu}_i') = true$, $\nu_i' \in f_i'(\tilde{\nu}_i')$, and for the invariants $\tilde{\nu}_i' \in Inv_i'(\tilde{l}_i)$ and $\nu_i' \in Inv_i'(l_i)$. By definition of parallel composition, $(\tilde{l}_1, \tilde{l}_2) \longrightarrow^{\alpha'}_{(\phi', f')} (l_1, l_2)$, with ϕ' and f' given as in Definition 8. Moreover, if we define $\tilde{\nu}' = \tilde{\nu}_1' \cup \tilde{\nu}_2'$ —the two valuations $\tilde{\nu}_1'$ and $\tilde{\nu}_2'$ coincide on the common variables— and $\tilde{\sigma}' = ((\tilde{l}_1, \tilde{l}_2), \tilde{\nu}') \in \Sigma'$, then $\phi'(\tilde{\nu}') = true$ and $\nu' \in f'(\tilde{\nu}')$. Furthermore $\tilde{\nu}' \in Inv'(\tilde{l}_1, \tilde{l}_2)$, and $\nu' \in Inv'(l_1, l_2)$, where the invariants Inv' of $H_1' \times H_2'$ are given according to Definition 8. Thus by the step semantics of Definition 3, $\tilde{\sigma}' \rightarrow^{\alpha} \sigma'$, as required.

Subcase: $\tilde{\sigma'}_i \rightarrow^{\alpha}\rightarrow^{f,t>0} \sigma'_i$
More precisely we are given

$$\begin{tikzcd}
\tilde{\sigma}'_1 \arrow[r, "\alpha_1"] & \breve{\sigma}'_1 \arrow[r, "f_1{,}t_1>0"] & \sigma'_1 \arrow[d, dash, "\text{-}|_{H'_1}"] \\
 & & \sigma' \arrow[d, dash, "\text{-}|_{H'_2}"] \\
\tilde{\sigma}'_2 \arrow[r, "\alpha_2"] & \breve{\sigma}'_2 \arrow[r, "f_2{,}t_2>0"] & \sigma'_2
\end{tikzcd}$$

for some states $\breve{\sigma}'_1$ and $\breve{\sigma}'_2$ and where $\sigma'_1 = (l_1, \nu'_1)$ and $\sigma'_2 = (l_2, \nu'_2)$. By the definition of the edges of H'_1 and H'_2 and the fact that H'_1 and H'_2 share the variable h_t, the two durations t_1 and t_2 are equal, say $t = t_1 = t_2$. Since time transitions don't change the location, $\breve{\sigma}'_1 = (l_1, \breve{\nu}'_1)$ and likewise $\breve{\sigma}'_2 = (l_2, \breve{\nu}'_2)$. Now the states $\breve{\sigma}'_1$ and $\breve{\sigma}'_2$ agree on the auxiliary history variable h_ρ, especially the state $\breve{\sigma}'' \in \Sigma''$ recorded last in h_ρ upon entering $\breve{\sigma}'_1$ resp. $\breve{\sigma}'_2$ is the same for H'_1 and H'_2. This assures that setting $\breve{\sigma}' \in \Sigma'$ as $((l_1, l_2), \breve{\nu})$ with $\breve{\nu}' = \breve{\nu}'_1 \cup \breve{\nu}'_2$ is well defined, and furthermore $\breve{\sigma}' \downarrow_{H'_i} = \breve{\sigma}'_i$. By the definition of the activities of the common auxiliary variable h_f, mutually connecting the two activities of H_1 and H_2, guarantees that there *exists* a common activity f. Thus, $\breve{\sigma}' \rightarrow^{f,t>0} \sigma'$. From there, the existence of some state $\tilde{\sigma}' \in \Sigma'$ such that $\tilde{\sigma}' \rightarrow^{\alpha} \breve{\sigma}'$ and $\tilde{\sigma}'|_{H'_i} = \tilde{\sigma}'_i$ follows by the same arguments as in the previous subcase.

The above induction shows, that the existence of two runs for H'_1 and H'_2 according to equation (3) implies that there exists a common run ρ' of H', which gives equation (1). With this property at hand, the proof of completeness is straightforward. Let the predicate φ be an invariant of H, i.e., $R(H) \rightarrow \varphi$. Now define the assertion networks Q'_i as the sets of reachable states $R(H'_i)$ for $i = 1$ and 2, and let furthermore $Q' = Q'_1 \times Q'_2 = \{\sigma' \in \Sigma_{H'} \mid \sigma' \downarrow_{H'_1} \in Q'_1 \wedge \sigma' \downarrow_{H'_2} \in Q'_2\}$. Assume further some state $(l, \nu') \in Q'_1 \times Q'_2$, i.e., $(l, \nu') = ((l_1, l_2), \nu')$ with $(l_1, \nu'|_{Var'_1}) \in R(H'_1)$ and $(l_2, \nu'|_{Var'_2}) \in R(H'_2)$. By equation (1) from above, $(l, \nu') \in R(H')$, whence with Lemma 12 $(l, \nu'|_{Var}) \in R(H)$. $\square$

5.3 Examples

Besides formalizing the proof rules in PVS, we applied the method to a number of examples, e.g., non-linear variations of the water level monitor [1], or a modified clock synchronization of the MPEG4 standard.

Example 16 (Thermostat). For the extended thermostat system of Example 10, we want to verify that the temperature of all rooms stays between the specified limits, and that the tank never gets below the minimum level needed for turning a heater on, i.e.,

$$\varphi = \epsilon \leq v \wedge \bigwedge_i x_i^{min} \leq x_i \leq x_i^{max}$$

is establied invariant of $H_1 \times \ldots \times H_n \times H_{fuel}$ (cf. Figure 2). Using the proof method from above, $Q_i^{cons} = u_i \leq c_i$ can be established as invariant of H_i, and thus the parallel composition H satisfies the conjunction $\bigwedge_i Q_i^{cons}$. Furthermore, under the assumption that the context of H_{fuel} satisfies the invariant $u \leq \sum_i c_i$, the fuel system H_{fuel} enjoys $Q_{fuel} = \mathrm{Vol}_1 \leq v$ as invariant. Finally, assuming the context of H_i satisfies the invariant Q_{fuel}, the property $Q_i^{temp} = x_i^{min} \leq x_i \leq x_i^{max}$ is an invariant of H_i. Since $Q_{fuel} \wedge \bigwedge_i Q_i^{temp}$ implies φ, the desired property φ holds for the composed system $H_1 \times \ldots \times H_n \times H_{fuel}$.

These examples and the proven properties are listed in Appendix A

6 Conclusion

In this paper we present an assertional, deductive proof method for the verification of hybrid systems. For the verification of composed systems, we give a complete proof rule to reduce the complexity introduced by the parallel composition. To facilitate the tedious verification of hybrid systems without restricting the model artificially, we embedded the proof system into the PVS theorem prover.

Beside offering the full power of higher-order logic, a further advantage of such a deductive verification environment is that it allows a straightforward rigorous formalization of the mathematical definitions, without the need to resort to any specific logic. Furthermore, PVS comes equipped with a wide range of automated proof-strategies and heuristics. Also its modularization facilities are helpful in structuring the work.

Of course, to perform the verification requires some expertise in hybrid systems, PVS, and the nature of inductive proofs, but the goal-directed manner of applying the proof-methods and rules gives valuable guidance. Sometimes, the theorem prover surprises the user by taking a long time for seemingly routing task (for instance, type-checking the collected theorems). In those cases, rephrasing the formalization into a mathematically equivalent one but using different data representations (e.g. using lists instead of sets) can cut down the times noticeably. The examples we analyzed in PVS demonstrate how to apply our mathematical framework in praxis. Despite their relative small size, they point out the relevance of our method for the verification of *arbitrary* hybrid systems (for instance hybrid systems with non-linear properties or parameterized hybrid systems).

As the main line of research on hybrid systems focuses on model checking techniques for appropriately *restricted* subclasses, there are less investigations on deductive methods for their verification. Closest in spirit to our work is [4], which embed timed automata into PVS and apply their approach on the steam boiler example. The same example is treated in [26], with the goal of deriving an implementation of a real-time program in a number of refinement steps [16]. The PVS theorem prover is also used in [14] in combination with model checking using HyTech [3] for the reachability analysis for various classes of linear hybrid automata. For the verification of safety properties of hybrid systems, [17] employ hybrid temporal logic HTL, an extension of interval temporal logic. They give a number of proof-rules which they prove sound. Likewise building upon temporal logic, [21] use the Stanford theorem prover STeP as proof environment. See [19] for an overview over deductive and algorithmic approaches for verification of hybrid systems.

For future work, we intend to apply our method to larger case studies, especially to extend the control example based on MPEG4 of [6], and further a laser steering system for mass spectroscopy. For the verification of complex case studies the addition of a tool which allows for a graphical specification of hybrid systems is essential, as is a parser that automatically generates the PVS verification conditions for a given specification. This also allows for a clean separation between type checking (i.e., checking that the specified system is indeed a hybrid system) and the actual verification task. Moreover, the representation of the verification conditions for each control mode and each discrete transition as separate lemmas improves the structure of the proofs and increases the level of automation. To improve the specification structure of hybrid systems, the interface information can be extended, for instance separating the variable set into input and output variables as in [20]. Such a cleaner separation is a necessary prerequisite for the development of an assume-guarantee reasoning scheme (cf. [25] or in the context of hybrid systems [13, 9]). Especially we expect that the verification will benefit from an alternative semantics allowing for compositional proofs [11].

References

1. R. Alur, C. Courcoubetis, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995. A preliminary version appeared in the proceedings of 11th. International Conference on Analysis and Optimization of Systems: Discrete Event Systems (LNCI 199).
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:252–235, 1994.
3. R. Alur, T. A. Henzinger, and P. Ho. Automatic symbolic verification of embedded systems. In *Proc. 14th Annual Real-Time Systems Symposium*, pages 2–11. IEEE Computer Society Press, 1993.
4. M. Archer and C. Heitmeyer. Verifying hybrid systems modeled as timed automata: A case sudy. In O. Maler, editor, *Hybrid and Real-Time Systems (HART'97)*, number 1201 in Lecture Notes in Computer Science, pages 171–185. Springer-Verlag, 1997.
5. M. D. D. Benedetto and A. Sangiovanni-Vincentelli, editors. *Proceedings of the 4th International Wokrshop on Hybrid Systems: Computation and Control (HSCC 2001), Rome*, volume 2034 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
6. J. B. de Meer and E. Ábrahám-Mumm. Formal methods for reflective system specification. In J. Grabowski and S. Heymer, editors, *Formale Beschreibungstechniken für verteilte Systeme*, pages 51–57. Universität Lübeck/Shaker Verlag, Aachen, Juni 2000 2000.
7. W.-P. de Roever, F. de Boer, U. Hannemann, J. Hooman, Y. Lakhnech, M. Poel, and J. Zwiers. *Concurrency Verification: Introduction to Compositional and Noncompositional Proof Methods*. Cambridge University Press, 2001.
8. R. W. Floyd. Assigning meanings to programs. In J. T. Schwartz, editor, *Proc. Symp. in Applied Mathematics*, volume 19, pages 19–32, 1967.
9. G. F. Frehse, O. Stursberg, S. Engel, R. Huuck, and B. Lukoschus. Modular analysis of discrete controllers for distributed hybrid systems. 2001. Technical Report, submitted for publication.
10. R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors. *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.
11. U. Hannemann. *Semantic Analysis of Compositional Proof Methods for Concurrency*. PhD thesis, Utrecht Universiteit, Oct. 2000.

12. T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata. In *27th Annual ACM Symposium on Theory of Computing*. ACM Press, 1995.
13. T. A. Henzinger, M. Minea, and V. Prabhu. Assume-guarantee reasoning for hierarchical hybrid systems. In Benedetto and Sangiovanni-Vincentelli [5], pages 275–290.
14. T. A. Henzinger and V. Rusu. Reachability verification for hybrid automata. In Henzinger and Sastry [15], pages 190–204.
15. T. A. Henzinger and S. Sastry, editors. *Proceedings of the First International Workshop on Hybrid Systems: Computation and Control (HSCC'98)*, volume 1386 of *Lecture Notes in Computer Science*. Springer, 1998.
16. J. Hooman. A compositional approach to the design of hybrid systems. In Grossman et al. [10], pages 121–148.
17. A. Kapur, T. A. Henzinger, Z. Manna, and A. Pnueli. Proving safety properties of hybrid systems. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems 1994*, volume 863 of *Lecture Notes in Computer Science*, Kiel, Germany, September 1994. Springer-Verlag. 3rd International School and Symposium.
18. Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In Grossman et al. [10], pages 179–208.
19. S. Kowalewski, P. Herrmann, S. Engell, H. Krumm, H. Treseler, Y. Lakhnech, R. Huuck, and B. Lukoschus. Approaches to the formal verification of hybrid systems. *at-Automatisierungstechnik. Special Issue: Hybrid Systems II: Analysis, Modeling, and Verification*, 49(2):66–74, Feb. 2001.
20. N. A. Lynch, R. Segala, and F. W. Vaandrager. Hybrid I/O automata revisited. In Benedetto and Sangiovanni-Vincentelli [5].
21. Z. Manna and H. B. Sipma. Deductive verification of hybrid systems using STeP. In Henzinger and Sastry [15].
22. MPEG-4 (moving picture experts group) ISO/IEC-standard, ISO number 14496, February 1999.
23. S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In D. Kapur, editor, *Automated Deduction (CADE-11)*, volume 607 of *Lecture Notes in Computer Science*, pages 748–752. Springer-Verlag, 1992.
24. O. Roux and V. Rusu. Uniformity for the decidability of hybrid automata. In R. Cousot and D. A. Schmidt, editors, *Proceedings of SAS '96*, volume 1145 of *Lecture Notes in Computer Science*, pages 301–316. Springer-Verlag, 1996.
25. N. Shankar. Lazy compositional verification. In W.-P. de Roever, H. Langmaack, and A. Pnueli, editors, *Compositionality: The Significant Difference (Compos '97)*, volume 1536 of *Lecture Notes in Computer Science*, pages 541–564. Springer, 1998.
26. J. Vitt and J. Hooman. Assertional specification and verification using PVS of the steam boiler system. In *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control System*, volume 1165 of *Lecture Notes in Computer Science*, pages 453–472. Springer, 1996.

A Examples

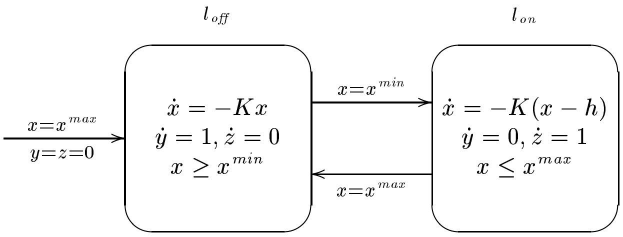
The appendix collects short descriptions and figures to document the examples dealt with in PVS.

A.1 Thermostat (File `thermostat.pvs`)

This is a modified version of the thermostat example in [1].

The temperature of a room is controlled by a *thermostat,* which continuously senses the temperature and turns a heater on and off. When the heater is off, the temperature, denoted by x, decreases according to the function $x(t) = x_0 e^{-Kt}$, where x_0 is the initial temperature, t the time, and $K > 0$ a room constant. With the heater off, the temperature follows the function $x(t) = (x_0 - h)e^{-Kt} + h$, where h is a real-valued constant which depends on the power of the heater. The initial temperature is x^{max} degrees ($x^{max} > 0$) and the heater is off initially. The heater turns on, as soon as the temperature reaches a threshold value of x^{min} degrees ($0 < x^{min} < x^{max}$, $h > x^{min} + x^{max}$). Conversely the heater turns off if the temperature reaches x^{max} degrees.

The temperature is to be kept between x^{min} and x^{max} degrees. Additionally, we verify that, at any point in time, the sum of the durations of the heating periods (denoted by z) is less or equal then the sum of the durations of the non-heating periods (denoted by y).



Inductive assertion network:

$$Q(l_{off}) = x^{min} \leq x \leq x^{max} \wedge -\frac{1}{K} \ln \frac{x}{x^{max}} \leq y - z$$
$$Q(l_{on}) \;= x^{min} \leq x \leq x^{max} \wedge -\frac{1}{K} \ln \frac{x^{min}(x^{min}-h)}{x^{max}(x-h)} \leq y - z$$

Invariance property:

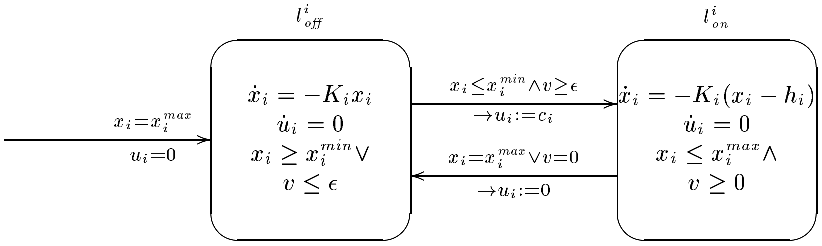
$$x^{min} \leq x \leq x^{max} \wedge 0 \leq y - z$$

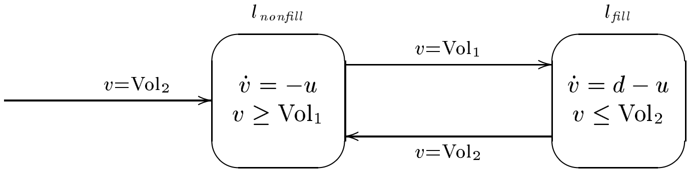
A.2 Heater system (file `thermostat_comp.pvs`)

We extend the thermostat example by introducing a new variable u denoting the *fuel consumption* of the heater per time unit. The variable u is control variable in both locations and has the value 0 in location l_{off} and c ($c > 0$) in l_{on}. Further, the heater can be turned on only if the amount of fuel volume in the tank is equal to or above a threshold value ϵ; heating is possible only as long as the tank is not empty. The discrete transitions and the invariants of the example from Section A.1 are modified accordingly.

We assume n such extended thermostat systems $heater_i$, $i = 0, \ldots, n-1$ with $n > 0$. The variable v denotes the actual fuel volume of the tank. Furthermore, we define a hybrid system *fuel* that represents a system filling a tank of volume Vol_2 each time the fuel volume of the tank reaches a threshold value Vol_1 with $0 < \epsilon < \mathrm{Vol}_1 < \mathrm{Vol}_2$. We assume that the inflow rate d for filling the tank is greater then the sum of the outflow rates $c = \sum_{i=0}^{n-1} c_i$ for heating. The sum of the actual outflow rates will be denoted by $u = \sum_{i=0}^{n-1} u_i$.

We *verify* that for all heater systems $heater_i$ the corresponding temperature x_i is always between the given boundaries x_i^{min} and x_i^{max} and that the fuel level of the tank is always above the threshold value ϵ, i.e., it can be heated all the time.





Inductive assertion network of $heater_i$:

$$Q^i_{cons}(l^i_{on}) = Q^i_{cons}(l^i_{off}) = u_i \leq c_i$$

Inductive assertion network of $fuel$ under the assumption $u \leq c$:

$$Q_{fuel}(l_{nonfill}) = Q_{fuel}(l_{fill}) = \mathrm{Vol}_1 \leq v$$

Inductive assertion network of $heater_i$ under the assumption $\mathrm{Vol}_1 \leq v$:

$$Q^i_{temp}(l^i_{on}) = Q^i_{temp}(l^i_{off}) = x_i^{min} \leq x_i \leq x_i^{max}$$

Invariance property of the heater system $H = fuel \times heater_0 \times \ldots \times heater_{n-1}$:

$$\epsilon \leq v \wedge \forall i : 0 \leq i \leq n-1 \to x_i^{min} \leq x_i \leq x_i^{max}$$

A.3 Clock synchronization (File `mpeg.pvs`)

The example deals with a modified clock synchronization of the MPEG4 [22] standard (cf. also [6]).

Given are a sender process with a local clock y and a receiver process with clock x. The sender transmits the value of its clock to the receiver each time it is about to send relevant data.

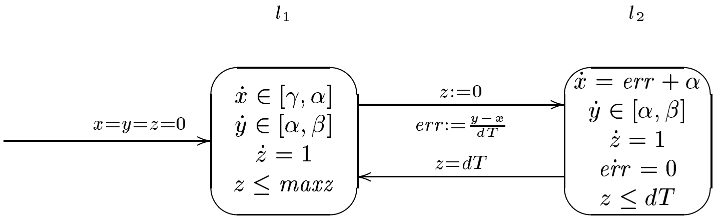
If the receiver clock would be set to the sended clock value in a discrete manner, transmitted information could get lost. Therefore the receiver clock is adapted in a continuous manner in dT time.

The sender clock y has a continuous and piecewise differentiable activity with a derivation in the interval $[\alpha, \beta]$, where $0 < \alpha < \beta$. The activity of the receiver clock x is likewise a continuous and piecewise differentiable function such that in the non-adapting state l_1 the derivation (if it exists) lies within $[\gamma, \alpha]$ with $0 < \gamma < \alpha$. and in the adapting state l_2 the activity of x is defined by

$$\dot{x} = \frac{y_0 - x_0}{dT} + \alpha,$$

where y_0 and x_0 represent the values of y and x at the last synchronization, i.e., at the last time point the value of the sender clock was sended.

We show that, if the length of the time periods between two consecutive synchronizations are always equal to or smaller than a threshold value $maxz$ ($maxz >= dT$), then the difference $y - x$ is always positive and smaller or equal to $(\beta - \gamma)maxz$.



Inductive assertion network:

$$\begin{aligned} Q(l_1) &= 0 \leq y - x \leq (\beta - \gamma)z \wedge 0 \leq z \leq maxz \\ Q(l_2) &= err(dT - z) \leq y - x \leq (\beta - \gamma)z + err(dT - z) \wedge \\ &\qquad 0 \leq z \leq dT \wedge 0 \leq err \leq (\beta - \gamma)\tfrac{maxz}{dT} \end{aligned}$$

Invariance property:

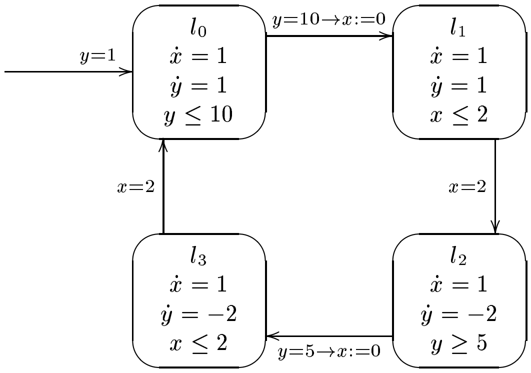
$$x \leq y \wedge y - x \leq (\beta - \gamma)maxz$$

A.4 Water level monitor (file `water_level_monitor.pvs`)

This is the *water-level monitor* example from [1]. The water level in a tank is controlled by a monitor, which continuously senses the water level and turns a pump on and off. There is a *delay* of 2 seconds from the time the monitor signals a status change to the time that the change becomes effective. The water level changes as a piecewise-linear function over time. When the pump is off, the water level, denoted by the variable y, falls by 2 inches per second; when the pump is on, the water level rises by 1 inch per second. Suppose that initially the water level is 1 inch and the pump is turned on.

The monitor signals whenever the water level passes 5 and 10 inches, resp. The system has 4 locations: in locations l_0 and l_1, the pump is turned on; in locations l_2 and l_3, the pump is off. The clock x is used to specify the delays.

We show that the water level is always between 1 and 12 inches.



Inductive assertion network:

$$\begin{aligned}
Q(l_0) &= 1 \leq y \leq 10 \\
Q(l_1) &= 1 \leq y \leq 10 + x \ \wedge \ x \leq 2 \\
Q(l_2) &= 5 \leq y \leq 12 \\
Q(l_3) &= 5 - 2x \leq y \leq 12 \ \wedge \ x \leq 2
\end{aligned}$$

Invariance property:

$$1 \leq y \leq 12$$

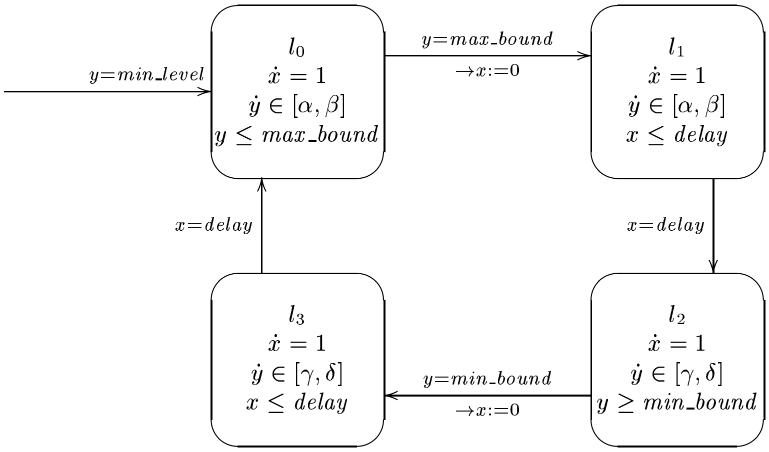
A.5 Modified water level monitor (file `water_level_monitor_non_linear.pvs`)

A modified version of the water level monitor example.

The water level monitor example is modified in the activities: if the pump is on, the water level increases according to a continuous and piecewise differentiable activity with derivation (if it exists) in $[\alpha, \beta]$ ($0 < \alpha < \beta$); if the pump is off, the water level decreases according to a continuous and piecewise differentiable activity with derivation (if it exists) in $[\gamma, \delta]$ ($\gamma < \delta < 0$).

The system needs $delay$ time unit from signaling to change the status by the monitor till the change becomes effective. We wish to keep the water level between min_level ($0 < min\_level$) and max_level ($max\_level > min\_level + delay \cdot \beta$ and $max\_level > min\_level - delay \cdot \gamma$). The monitor signals to turn the pump on by the water level $min\_bound = min\_level - delay\gamma$ and signals to turn the pump off by the water level $max\_bound = max\_level - delay \cdot \beta$.

We verify that the water level is always between min_level and max_level.



$$
\begin{aligned}
Q(l_0) &= min\_level \leq y \leq max\_bound \\
Q(l_1) &= min\_level \leq y \leq max\_bound + x\beta \,\wedge\, x \leq delay \\
Q(l_2) &= min\_bound \leq y \leq max\_level \\
Q(l_3) &= min\_bound + x\gamma \leq y \leq max\_level \,\wedge\, x \leq delay
\end{aligned}
$$

Invariance property:

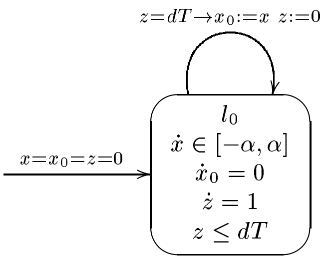
$$min\_level \leq y \leq max\_level$$

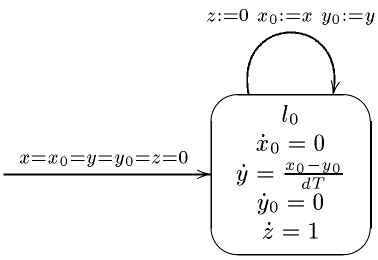
A.6 Linear approximator (file `linear_approximation.pvs`)

We define two hybrid systems $input\_generator$ and $linear\_approximator$. The first one changes the value of a variable x according to a continuous and piecewise differentiable activity with derivation (if it exists) in $[-\alpha, \alpha]$ $(0 < \alpha)$. The second one reads the value of x periodically after each dT time interval and approximates the value of x linear (basing on the last two received values). The approximated value is represented by the variable y. For measuring the time we introduce the clock variable z.

Note that if f and g denote the activities for x and y, resp., then $g(t)$ is an approximation of $f(t - dT)$ and not of $f(t)$.

We verify that in each reachable state of the composition system, $|x - y| <= \alpha \cdot dT$.





Inductive assertion network for $input\_generator$:

$$Q_{input}(l_0) = x_0 - \alpha z \leq x \leq x_0 + \alpha z$$

Inductive assertion network for $linear\_approximator$:

$$\begin{aligned} Q_{approx}(l_0) = y = y_0 + \tfrac{x_0 - y_0}{dT} z \wedge \\ |x_0 - y_0| \leq \alpha dT \wedge \\ |x - y| \leq \tfrac{|x_0 - y_0|}{dT}(dT - z) + \alpha z \end{aligned}$$

Invariance property:

$$|x - y| \leq \alpha dT$$