Assume-Guarantee Reasoning for Hybrid I/O-Automata by Over-Approximation of Continuous Interaction

Goran Frehse, Zhi Han, Bruce Krogh
Department of Electrical and Computer Engineering, Carnegie Mellon University,
Pittsburgh, PA 15213, USA, {gfrehse,zhih,krogh}@ece.cmu.edu

***Abstract*— Assume-guarantee reasoning (AGR) is recognized as a means to counter the state explosion problem in the verification of safety properties. We propose a novel assume-guarantee rule for hybrid systems based on simulation relations. This makes it possible to perform compositional reasoning that is conservative in the sense of over-approximating the composed behaviors. The framework is formally based on hybrid input/output automata and their labeled transition system semantics. In contrast to previous approaches that require global receptivity conditions, the circularity is broken in our approach by a state-based nonblocking condition that can be checked in the course of computing the AGR simulation relations. The proposed procedures for AGR are implemented in a computational tool, called PHAVer, for the class of linear hybrid I/O automata, and the approach is illustrated with a simple example.**

I. Introduction

Hybrid automata are widely used to model and analyze continuous-discrete behavior occurring, e.g., in digitally controlled systems. Applications in verification are so far limited in their scope because the computational complexity increases exponentially with the number of variables and components in the system, which is further worsened by complex continuous dynamics. These problems can be addressed by two approaches: abstraction and compositional reasoning. Abstraction refers to the use of conservative approximations with simpler dynamics, less variables, fewer discrete states etc. Compositional reasoning exploits the modular structure of a system and infers knowledge about the composed system. To retain as much knowledge about the interaction between systems as possible, we use an input/output-structure for hybrid systems. This allows us to model phenomena like open inputs.

On a formal basis, abstraction can be incorporated into proofs and verified by establishing *simulation relations* that identify matching behavior between states of systems. Since the continuous trajectories of two hybrid systems are impossible to compare without a finitary representation, the behavior of a hybrid system is formally defined with a hybrid labeled transition system, i.e., by transitions between states, with an associated action or time. Using this state-based approach, compositional reasoning becomes implementable if the behavior has a finite representation, e.g., based on polyhedra. However, the continuous path information is lost, which leads to an over-approximation in the behavior when the labeled transition systems are composed rather than hybrid automata.

Assume-guarantee reasoning is a form of compositional proof, that analyzes a subsystem using assumptions about the rest of the system. Provided a simplified model of each module, one original model is composed with the rest of the simplified system and its behavior verified. If the simplified model is a conservative abstraction, the proof is sound. Otherwise additional conditions must ensure that no undetected violations can occur. Applications in literature rely on non-blocking between shared actions for soundness. We present a sufficient condition based on simulation relations that makes no such restriction. While the resulting over-approximation can be prohibitively extensive, e.g., for general feed-back systems or systems with inputs in differential equations, it can present a valuable solution for sufficiently restricted classes of systems, e.g., feed-back systems with bounded or sampled inputs.

The hybrid I/O automaton (HIOA) model was first introduced by Lynch et al. [1] to model the input/output behavior of hybrid systems. Our model is an extension of the hybrid automata in [2], which in its simplicity is more apt to our proofs. Compositional reasoning with simulation relations has been first employed by Grumberg and Long [3] for discrete systems. Assume-guarantee reasoning for hybrid systems has been studied by Alur and Henzinger et al. [4], [5], in which the condition for assume/guarantee rules to hold is that every module must be *receptive*, which requires that the module is not blocked by *any* possible input. The receptiveness condition requires more effort for modeling physical systems in the HIOA framework. An assume-guarantee rule using simulation relations has been provided in [6], but it also requires receptiveness.

The following section defines hybrid I/O-automata and their labeled transition system semantics. Section III introduces the notion of simulation relations, and Sect IV presents our proof rule for assume-guarantee reasoning. Section V illustrates the usefulness of the approach with experimental data. The proofs have been omitted for lack of space. They can be found in [7].

II. Hybrid Automata and Hybrid Labeled Transition Systems

Hybrid automata are a compact modeling paradigm for continuous-discrete behavior based on state-transition systems [8], [9], [2].The I/O-automata model [1] imposes additional structure on the model by declaring certain variables as inputs and outputs. In the following variation of this

model, an automaton has its own set of state variables, to which other automata have access if they are declared as output variables. Input variables can change their value arbitrarily, in the sense of an "open" input. Since we are interested in safety properties of the system, we use a hybrid I/O-extension of the labeled transition system semantics from [2].

Definition 2.1: Given a set *Var* of variables, a valuation $v : Var \to \mathbb{R}$ maps a real number to each variable. Let $V(Var)$ denote the set of valuations over *Var*. An *activity* is a function $f : \mathbb{R}^{\geq 0} \to V$ in C^∞ and describes the change of valuations over time. Let $act(Var)$ denote the set of activities over *Var*. Let $f + t$ be defined for $t \geq 0$ by $(f + t)(d) = f(d + t), d \in \mathbb{R}^{\geq 0}$. A set S of activities is *time-invariant* if for all $f \in S, t \in \mathbb{R}^{\geq 0} : f + t \in S$.

Definition 2.2: A *hybrid input/output-automaton* (HIOA) $H = (Loc, Var_S, Var_I, Var_O, Lab, \to, Act, Inv, Init)$ consists of the following:

- A finite set *Loc* of locations.
- Finite and disjoint sets of state and input variables, Var_S and Var_I, and of output variables $Var_O \subseteq Var_S$. Let $Var = Var_S \cup Var_I$. The state space is $S_H = Loc \times V(Var)$, and $(l, v) \in S_H$ a *state*.
- A finite set *Lab* of labels,
- A finite set of discrete transitions $\to \subseteq Loc \times Lab \times 2^{V(Var)\times V(Var)} \times Loc$. A transition $(l, a, \mu, l') \in \to$ is also written as $l \xrightarrow{a,\mu}_H l'$.
- A mapping $Act : Loc \to 2^{act(Var)}$ from locations to time-invariant sets of activities.
- A mapping $Inv : Loc \to 2^{V(Var)}$ from locations to sets of valuations.
- A set $Init \subseteq Loc \times V(Var)$ of initial states.

Hybrid automata can be composed with a parallel composition operator, which enables the modular modeling of complex systems. For HIOA, a notion of compatability is needed:

Definition 2.3: HIOA $H_i = (Loc_i, Var_{Si}, Var_{Ii}, Var_{Oi}, Lab_i, \to_i, Act_i, Inv_i, Init_i)$, $i = 1, 2$, are *compatible* if $Var_{S1} \cap Var_{S2} = \emptyset$, and $Var_{Ii} \cap Var_{Sj} \subseteq Var_{Oj}$ for $(i, j) \in \{(1, 2), (2, 1)\}$.

The parallel composition operator determines how two automata interact. Changes in the continuous variables must be matched in both, and a discrete transition can only change a variable if the automaton that has it as a state variable can match the change. An input variable disappears in the composition with an automaton that has it as an output variable.

Definition 2.4: Given compatible HIOA $H_i = (Loc_i, Var_{Si}, Var_{Ii}, Var_{Oi}, Lab_i, \to_i, Act_i, Inv_i, Init_i)$, $i = 1, 2$, their *parallel composition* $H_1 \,||\, H_2$ is the HIOA $H = (Loc_1 \times Loc_2, Var_{S1} \cup Var_{S2}, (Var_{I1} \cup Var_{I2}) \setminus (Var_{S1} \cup Var_{S2}), Var_{O1} \cup Var_{O2}, Lab_1 \cup Lab_2, \to, Act, Inv, Init)$ with

- $f \in Act(l_1, l_2)$ iff $f{\downarrow_{Var_i}} \in Act_i(l_i)$, $i = 1, 2$,
- $v \in Inv(l_1, l_2)$ iff $v{\downarrow_{Var_i}} \in Inv_i(l_i)$, $i = 1, 2$, and
- $(l_1, l_2) \xrightarrow{a,\mu} (l_1', l_2')$ with $\mu = \{(v, v') | (v{\downarrow_{Var_i}}, v'{\downarrow_{Var_i}}) \in \mu_i, i = 1, 2\}$ iff for $i = 1, 2$: $a \in Lab_i \wedge l_i \xrightarrow{a,\mu_i}_i l_i'$, or $a \notin Lab_i \wedge l_i = l_i' \wedge \mu_i = \{(v, v') | v{\downarrow_{Var_{Si}}} = v'{\downarrow_{Var_{Si}}}\}$.
- $((l_1, l_2), v) \in Init$ iff $(l_i, v{\downarrow_{Var_i}}) \in Init_i$, $i = 1, 2$.

We use *hybrid labeled transition systems* to provide the semantic basis for hybrid automata. They preserve most of the structure of the hybrid automaton but abstract from the continuous activities and invariants:

Definition 2.5: A *hybrid labeled transition system* (HLTS) $L = (Loc, Var_S, Var_I, Var_O, \Sigma, \to_L, Init)$ consists of a finite set *Loc* of locations, finite disjoint sets Var_I and Var_S of variables, a set $Var_O \subseteq Var_S$ of output variables, a set Σ of labels, a transition relation $\to \subseteq Loc \times V(Var) \times \Sigma \times V(Var) \times Loc$ and a set of initial states $Init \subseteq Loc \times V(Var)$, where $Var = Var_I \cup Var_S$.

HLTSs in composition interact, similarly to HIOA, by synchronizing on common labels:

Definition 2.6: Given HLTSs $L_i = (Loc_i, Var_{Si}, Var_{Ii}, Var_{Oi}, \Sigma_i, \to_i, Init_i)$, $i = 1, 2$, $Var_{S1} \cap Var_{S2} = \emptyset$, their *parallel composition* $L_1 \,||\, L_2$ is the hybrid labeled transition systems $L = (Loc_1 \times Loc_2, Var_{S1} \cup Var_{S2}, (Var_{I1} \cup Var_{I2}) \setminus (Var_{S1} \cup Var_{S2}), Var_{O1} \cup Var_{O2}, \Sigma_1 \cup \Sigma_2, \to_L, Init)$ with $((l_1, l_2), v) \xrightarrow{\alpha}_L ((l_1', l_2'), v')$ iff $\alpha \in \Sigma_i$ and $(l_i, v{\downarrow_{Var_i}}) \xrightarrow{\alpha}_{L_i} (l_i', v'{\downarrow_{Var_i}})$ or $\alpha \notin \Sigma_i$ and $l_i = l_i'$, $v{\downarrow_{Var_{Si}}} = v'{\downarrow_{Var_{Si}}}$, for $i = 1$ and $i = 2$, and $Init = \{((l_1, l_2), v) | (l_i, v{\downarrow_{Var_i}}) \in Init_i, i = 1, 2\}$.

The behavior of a HIOA is defined by an associated HLTS, called its *timed transition system*:

Definition 2.7: The *timed transition system* (TTS) of a HIOA H is the HLTS $[\![H]\!] = (Loc, Var_S, Var_I, Var_O, \Sigma, \to_{LH}, Init)$ where $\Sigma = Lab \cup \mathbb{R}^{\geq 0} \cup \varepsilon$ and

- $(l, v) \xrightarrow{a}_{LH} (l', v')$ iff $l \xrightarrow{a,\mu}_H l', (v, v') \in \mu, v \in Inv(l), v' \in Inv(l')$ (discrete transitions),
- $(l, v) \xrightarrow{t}_{LH} (l', v')$ iff $l = l'$ and there exists $f \in Act(l), f(0) = v, f(t) = v'$, and $\forall t', 0 \leq t' \leq t : f(t') \in Inv(l)$ (timed transitions),
- $(l, v) \xrightarrow{\varepsilon}_{LH} (l', v')$ iff $l = l', v{\downarrow_{Var_S}} = v'{\downarrow_{Var_S}}, v, v' \in Inv(l)$ (environment transitions).

The loss of information about the activities of the hybrid automaton entails that the composition operator $||$ and the timed transition sytem operator $[\![\,]\!]$ are not commutative. This will be discussed in more detail in Sect. III, when the comparison of systems is formalized by the notion of simulation relations. The following result is important to the discussion:

Proposition 2.1: For any hybrid automata H_1, H_2, and $\alpha \in Lab_1 \cup Lab_2 \cup \mathbb{R}^{\geq 0} \cup \varepsilon$ a transition $((l_1, l_2), v) \xrightarrow{\alpha}_{H_1 \,||\, H_2} ((l_1', l_2'), v')$ in $[\![H_1 \,||\, H_2]\!]$ implies a transition $((l_1, l_2), v) \xrightarrow{\alpha}_{[\![H_1]\!] \,||\, [\![H_2]\!]} ((l_1', l_2'), v')$ in $[\![H_1]\!] \,||\, [\![H_2]\!]$.

A simple, but interesting, class of hybrid automata are *linear hybrid automata* [9], which can be described using linear formulas. Their behavior can be computed exactly using polyhedra:

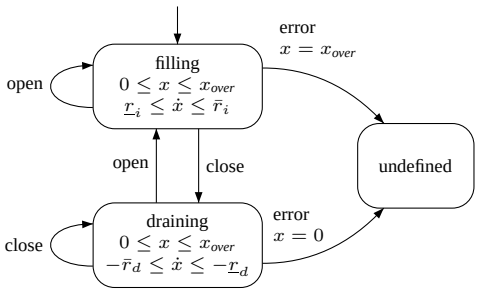
Definition 2.8: A *linear HIOA* is a HIOA in which

- for all locations $l \in Loc$, $Act(l)$ is given by a linear

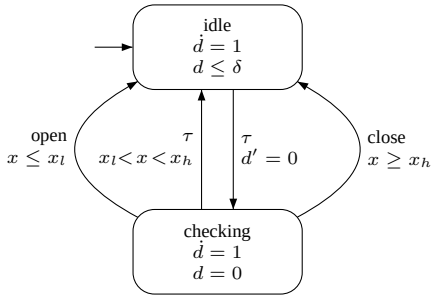
formula over the time derivatives of the state variables, i.e., over $\dot{Var}_S = \{dx/dt | x \in Var_S\}$.
- for all locations $l \in$ *Loc*, *Inv*(l) is given by a linear formula over *Var*,
- for all transitions $(l, a, \mu, l') \in \rightarrow$, μ is given by a linear formula over *Var* $\cup$ *Var*$'$, where *Var*$'$ denotes the variables of the second element of a pair $(v, v') \in \mu$.

We will illustrate our approach with the following simple example:

Example 2.1: Consider a tank level monitoring system consisting of a tank with continuous outflow, a discrete inlet valve (modeled as part of the tank), and a controller. The tank is modeled as a linear HIOA P_1, shown in Fig. 1(a). Its inlet valve is operated by the controller via the labels *open* and *close*. The level x of the tank changes at a rate $\underline{r}_i \leq \dot{x} \leq \bar{r}_i$ if the valve is open, and at a rate of $-\bar{r}_d \leq \dot{x} \leq -\underline{r}_d$ if it is closed. The location "undefined" represents states that were excluded from the model, such that unmodeled behavior can be detected with the transitions with label "error". P_1 has the state and output variable x, and no input variable. The controller, modeled by the linear HIOA P_2 in Fig. 1(b), is triggered by the timer d every δ seconds to check the level of the tank, and instantly decides whether to open the valve, close it or do nothing, after which it returns to the idle state. P_2 has the input variable x, the state variable d and no output variable.



(a) Tank P_1



(b) Controller P_2

Fig. 1. Tank level monitoring system

III. Simulation of Hybrid Automata

A simulation relation between automata P and Q relates states in P to all the states of Q that show the same, or more, behavior. The automaton Q is considered to contain the behavior of P, in the sense of an over-approximation, if every initial state of P finds a corresponding initial state of Q in the relation. This is denoted as $P \preceq Q$. Since the behavior of a HIOA is given by its TTS, we define first simulation for HLTSs. In the following, let P and Q be HLTS $P = (Loc_P, Var_{SP}, Var_{IP}, Var_{OP}, \Sigma_P, \rightarrow_P, Init_P)$ and $Q = (Loc_Q, Var_{SQ}, Var_{IQ}, Var_{OQ}, \Sigma_Q, \rightarrow_Q, Init_Q)$. In order for P to be comparable with Q, it must have a subset of the input, and the same output variables, and the same labels, formally:

Definition 3.1: P is *comparable* with Q if $\Sigma_P = \Sigma_Q$, $Var_{IQ} \subseteq Var_{IP}$ and $Var_{OQ} = Var_{OP}$. Hybrid automata H_1 and H_2 are comparable if their TTSs are comparable.

Definition 3.2: Given HTLS P, Q, P comparable with Q, a relation $R \subseteq S_P \times S_Q$ is a *simulation relation* if and only if $R \subseteq \{(k, u, l, v) | u{\downarrow}_{Var_{OQ}} = v{\downarrow}_{Var_{OQ}} \wedge u{\downarrow}_{Var_{IQ}} = v{\downarrow}_{Var_{IQ}}\}$ and for all $(p, q) \in R, \alpha \in \Sigma_P, p' \in S_P$ holds:

$$p \xrightarrow{\alpha} p' \;\Rightarrow\; \exists q' \in S_Q : (q \xrightarrow{\alpha} q' \wedge (p', q') \in R). \quad (1)$$

A state q *simulates* a state p if there exists a simulation relation R with $(p, q) \in R$, written as $p \preceq q$. Q *simulates* P, written as $P \preceq Q$, if and only if for all $(k, u) \in Init_P$ there exists a $(l, v) \in Init_Q$ such that $(k, u) \preceq (l, v)$. For any HIOA H_1, H_2, let $H_1 \preceq H_2$ if $[\![H_1]\!] \preceq [\![H_2]\!]$.

Example 3.1: Consider the tank level monitoring system from Ex. 2.1. The goal of the verification is to show that the tank level stays within the limits $x_m \leq x \leq x_M$ and that the model remains within the modeling bounds, i.e., produces no "error"-transitions. This invariant can be specified with the linear HIOA Q shown in Fig. 2. Self-loops allow the labels τ, "open" and "close" at any time, while the label "error" never occurs. Q has the state and output variable x, and no input variables.

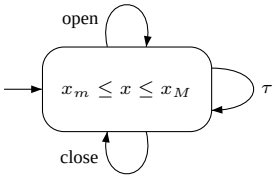


Fig. 2. Specification Q of the composed system

In order for the simulation concept to be applied in a compositional analysis, it must allow inference about the behavior of comparable automata, and hold under different contexts, i.e., when composed with other automata. For HLTS, this is always the case, which is formally expressed as follows:

Proposition 3.1: Simulation of HLTSs is a precongruence with respect to parallel composition, i.e., reflexive, transitive and invariant under parallel composition: $P \preceq Q \Rightarrow P \,||\, S \preceq Q \,||\, S$.

Simulation for HIOA, on the other hand, is not invariant under composition. It retains the remaining properties:

Corollary 3.1: Simulation for HIOA is a preorder.

procedure *GetSimRel*
Input: Hybrid Labeled Transition Systems P, Q,
optionally initial relation $R^0 \subseteq S_P \times S_Q$,
Output: a simulation relation R
if R^0 **undefined,** $R^0 := S_P \times S_Q$
$R := R^0 \cap \{(k,u,l,v) | u{\downarrow}_{Var_{OQ}} = v{\downarrow}_{Var_{OQ}} \wedge u{\downarrow}_{Var_{IQ}} = v{\downarrow}_{Var_{IQ}}\}$
while there exist (k,l)
with $R(k,l) \cap B(k,l) \neq \emptyset$ **do**
$R(k,l) := R(k,l) \cap \neg B(k,l)$
end while

Fig. 3. Semi-algorithm for computing a simulation relation

Intuitively, a HLTS only carries information about which states can transition to which, and whether by time elapse or by a discrete transition, but not about the trajectories taken to get there. Consequently, the composition of two timed transition systems of hybrid automata with shared variables is an over-approximation of the behavior of the composed hybrid automata because non-matching trajectories in the systems can be paired. The compositional reasoning in the following section will make use of this over-approximation, formally expressed by the following proposition, which follows directly from Prop. 2.1, and the definition of TTSs:

Proposition 3.2: $[\![H_1 \,||\, H_2]\!] \preceq [\![H_1]\!] \,||\, [\![H_2]\!]$.

A simulation relation R between HLTSs P and Q can be obtained with a fixpoint computation, in which the states that violate (1) are successively removed. The set of violating states in a pair of locations $(k,l) \in Loc_P \times Loc_Q$ is given by a set of valuations:

$$B(k,l) = \{(u,v) | (k,u) \xrightarrow{\alpha} (k',u') \wedge \nexists (l',v') : [(l,v) \xrightarrow{\alpha} (l',v') \wedge (k',u',l',v') \in R]\}. \quad (2)$$

Then R is the largest fixpoint of the operator

$$R(k,l) := R(k,l) \cap \neg B(k,l). \quad (3)$$

The relation can be initialized with the product of the set of reachable states. Depending on the system, this can tremendously speed up or slow down the convergence [10]. A simple semi-algorithm to compute simulation relations is shown in Fig. 3. For a more detailed discussion of simulation relations for hybrid automata, see [10], and more advanced algorithms for computing simulation relations can be found, e.g., in [11].

IV. Assume-Guarantee Reasoning

Assume-guarantee reasoning aims at deducing the behavior of a composed system from an analysis of parts of the system under assumptions about the rest of the system. Consider a system consisting of HIOA $P = P_1 \,||\, P_2$ with a specification $Q = Q_1 \,||\, Q_2$. The goal is to show that $P_1 \,||\, P_2 \preceq Q_1 \,||\, Q_2$, which by definition is equal to showing that $[\![P_1 \,||\, P_2]\!] \preceq [\![Q_1 \,||\, Q_2]\!]$ for their TTSs.

Non-circular assume-guarantee reasoning occurs if the abstraction of one automaton serves as the guarantee to another, yielding a triangular structure:

$$\frac{\begin{array}{cc} [\![P_1]\!] & \preceq \; [\![Q_1]\!] \\ [\![Q_1]\!] \,||\, [\![P_2]\!] & \preceq \; [\![Q_1 \,||\, Q_2]\!] \end{array}}{P_1 \,||\, P_2 \preceq Q_1 \,||\, Q_2}. \quad (4)$$

The proof is straightforward using the precongruence properties of simulation: $[\![P_1]\!] \preceq [\![Q_1]\!]$ implies $[\![P_1]\!] \,||\, [\![P_2]\!] \preceq [\![Q_1]\!] \,||\, [\![P_2]\!]$ due to the invariance of the simulation of HLTS under composition. With transitivity it follows from $[\![Q_1]\!] \,||\, [\![P_2]\!] \preceq [\![Q_1 \,||\, Q_2]\!]$ that $[\![P_1]\!] \,||\, [\![P_2]\!] \preceq [\![Q_1 \,||\, Q_2]\!]$, and with Prop. 3.2 follows the conclusion. While in cases with continuous input the application of Prop. 3.2 can lead to gross over-approximation, it does enable the proof for some interesting examples and applications.

Circular assume-guarantee reasoning additionally uses Q_2 to restrict the behavior of P_1 in the analysis. It is only sound if additional conditions, in the following called A/G conditions, ensure that Q_1 and Q_2 don't block transitions in their composition that are enabled for the composition of P_1 and P_2. The basic structure is:

$$\frac{\begin{array}{cc} [\![P_1]\!] \,||\, [\![Q_2]\!] & \preceq \; [\![Q_1 \,||\, Q_2]\!] \\ [\![Q_1]\!] \,||\, [\![P_2]\!] & \preceq \; [\![Q_1 \,||\, Q_2]\!] \\ \multicolumn{2}{c}{\textit{A/G conditions}} \end{array}}{P_1 \,||\, P_2 \quad \preceq \quad Q_1 \,||\, Q_2}. \quad (5)$$

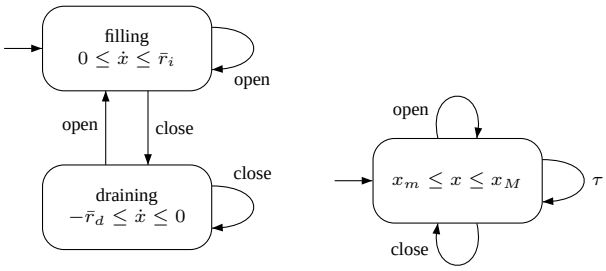
Example 4.1: Consider the tank level monitoring system from Ex. 2.1. To verify the global specification Q from Fig. 2 using assume-guarantee reasoning, specifications Q_i are created manually for each subsystem. It is then checked that their composition guarantees Q, i.e., that $Q_1 || Q_2 \preceq Q$. The specification Q_1 for the tank, see Fig. 4(a), is a simplified version of P_1. The inflow and outflow rate are over-approximated and the invariants as well as the location "undefined" are omitted. The essential information that guarantees the functioning of the controller within the A/G-reasoning is that the level rises after opening of the valve, and falls after closing. The label "error" is in Lab_{Q_1}, but is never allowed. Q_1 has the state and output variable x, and no input variables. The specification Q_2 for the controller, shown in Fig. 4(b), is simply that it somehow guarantees the invariant. It differs from Q only in that its alphabet does not contain "error". Q_2 has no state variable, and the input variable x. Note that Q_2 represents the function of the controller, and has virtually nothing to do with the controller implementation P_2. This allows to abstract from implementation details such as the timer d. It also means that the specification doesn't have to be reinvented if the implementation changes. Note that neither Q_1 nor Q_2 are conservative over-approximations of P_1 and P_2.

The A/G-condition is given by the following theorem:

Theorem 4.1 (A/G-simulation): Consider HIOA P_1, P_2, Q_1, Q_2, P_i comparable to Q_i, for which

$$[\![P_1]\!] \,||\, [\![Q_2]\!] \preceq [\![Q_1 \,||\, Q_2]\!] \quad \text{and} \quad (6)$$

$$[\![Q_1]\!] \,||\, [\![P_2]\!] \preceq [\![Q_1 \,||\, Q_2]\!]. \quad (7)$$



(a) Tank spec. Q_1 (b) Controller spec. Q_2

Fig. 4. Modular specifications for A/G-reasoning

If there exist simulation relations R_1 for (6) and R_2 for (7) such that for all $((k_1, k_2, x), (l_1, l_2, z))$ for which there exist $(\hat{l}_1, \hat{z}_1), (\hat{l}_2, \hat{z}_2)$ with $((k_i, l_j, y_i), (l_i, \hat{l}_j, \hat{z}_i)) \in R_i$, and all $\alpha \in \Sigma_{P_1} \cap \Sigma_{P_2}$ holds

$$(k_1, k_2, x) \xrightarrow{\alpha}_{P_1||P_2} (k_1', k_2', x') \Rightarrow [\exists i, l_i', z' : (l_i, z{\downarrow_{Q_i}}) \xrightarrow{\alpha}_{Q_i} (l_i', z'{\downarrow_{Q_i}}) \wedge z'{\downarrow_{P_j \cap Q_i}} = x'{\downarrow_{P_j \cap Q_i}}] \quad (8)$$

a simulation relation for $P_1 \,||\, P_2 \preceq Q_1 \,||\, Q_2$ is given by

$$R = \{((k_1, k_2, x), (l_1, l_2, z)) | \exists y_i, \hat{l}_j, \hat{z}_j : ((k_i, l_j, y_i), (l_i, \hat{l}_j, \hat{z}_i)) \in R_i, y_i{\downarrow_{P_i}} = x{\downarrow_{P_i}}, y_i{\downarrow_{Q_j}} = z{\downarrow_{Q_j}}, \hat{z}_i{\downarrow_{Q_i}} = z{\downarrow_{Q_i}}\}.$$

Note that R not necessarily contains the initial states. The A/G-condition is trivially fulfilled if for every label in $Lab_{Q1} \cap Lab_{Q2}$ either Q_1 or Q_2 is non-blocking, and that in each location of $Q_1||Q_2$ either always allows an arbitrary time elapse.

The A/G-condition (8) looks similar to the requirement of simulation (1) for the composed automata, but differs in two important points: Firstly, the target states are not required to lie within the relation, so there is no fixpoint computation necessary. Secondly, it is only required that either one of Q_1 or Q_2 has a corresponding transition.

Checking for A/G-simulation consists of the construction of simulation relations R_1 and R_2 for which the A/G-condition holds. A simple procedure is to remove potentially violating states from candidate relations R_1^0 and R_2^0. The resulting trimmed relations R_1' and R_2' must (again) be turned into simulation relations R_1 and R_2 by a fixpoint computation. As the trimming is done as on over-approximation, the candidate relations should be as small as possible, and are therefore initialized as simulation relations.

The sets of critical labels and states in R_i that could violate the A/G-conditions are given for $(i, j) \in \{(1, 2), (2, 1)\}$ by:

$$D_{R_i} = \{(q_i, q_j, \alpha) | \alpha \in \Sigma_{P_1} \cap \Sigma_{P_2} \wedge \exists p_i, p_i' : ((p_i, q_j), (q_1, q_2)) \in R_i \wedge p_i \xrightarrow{\alpha} p_i' \wedge \nexists q_1' : q_1 \xrightarrow{\alpha} q_1' \wedge \nexists q_2' : q_2 \xrightarrow{\alpha} q_2'\} \quad (9)$$

It is a sufficient condition for A/G-simulation that $D_{R_1} \cap D_{R_2} = \emptyset$. If there are such violating states, an element $((p_i, q_j), (q_1, q_2))$ of R_i is removed if $p_i \xrightarrow{\alpha} p_i' \wedge (q_1, q_2, \alpha) \in D_{R_j}$. An algorithm is shown in Fig. 5.

procedure *CheckAGSimulation*
Input: hybrid automata P_1, P_2, Q_1, Q_2
Output: A/G-simulation relations R_1, R_2

$R_1^0 :=$ *GetSimRel*$([\![P_1]\!] \,||\, [\![Q_2]\!], [\![Q_1 \,||\, Q_2]\!])$
$R_2^0 :=$ *GetSimRel*$([\![P_2]\!] \,||\, [\![Q_1]\!], [\![Q_1 \,||\, Q_2]\!])$
for $(i, j) \in \{(1, 2), (2, 1)\}$**:**
  $D_{R_i} = \{(q_i, q_j, \alpha) | \alpha \in \Sigma_{P_1} \cap \Sigma_{P_2} \wedge \exists p_i, p_i' : ((p_i, q_j), (q_1, q_2)) \in R_i^0 \wedge p_i \xrightarrow{\alpha} p_i' \wedge \nexists q_1' : q_1 \xrightarrow{\alpha} q_1' \wedge \nexists q_2' : q_2 \xrightarrow{\alpha} q_2'\}$.
if $D_{R_1} \cap D_{R_2} \neq \emptyset$
  for $(i, j) \in \{(1, 2), (2, 1)\}$**:**
    $R_i' := R_i^0 \backslash \{((p_i, q_j), (q_1, q_2)) | \exists p_i' : p_i \xrightarrow{\alpha} p_i' \wedge (q_1, q_2, \alpha) \in D_{R_j}\}$
  $R_1 :=$ *GetSimRel*$([\![P_1]\!] \,||\, [\![Q_2]\!], [\![Q_1 \,||\, Q_2]\!], R_1')$
  $R_2 :=$ *GetSimRel*$([\![P_2]\!] \,||\, [\![Q_1]\!], [\![Q_1 \,||\, Q_2]\!], R_2')$
else
  for $i = 1, 2$**:** $R_i := R_i^0$
end if

Fig. 5. Algorithm for checking assume-guarantee simulation of labeled transition systems with composite checking of R_1 and R_2.

Note that there is a choice whether to remove a state from R_1 or R_2. Therefore the order of determining D_{R_i} and trimming matters, and other solutions than the one in Fig. 5 are possible. To finalize the A/G-proof, it must be shown that all the initial states of $P_1||P_2$ have a matching initial state of $Q_1||Q_2$ in R. Let R_i' be defined for $(i, j) \in \{(1, 2), (2, 1)\}$ as

$$R_i' = \{((k_i, u_j), (l_1, l_2, z)) | \exists \hat{l}_j, \hat{z}_i : ((k_i, l_j, y_i), (l_i, \hat{l}_j, \hat{z}_i)) \in R_1, z{\downarrow_{Q_1}} = \hat{z}_1{\downarrow_{Q_1}}, z{\downarrow_{Q_2}} = y_1{\downarrow_{Q_2}}\}.$$

It must be shown that for all (k_1, k_2, x), $(k_1, x{\downarrow_{P_1}}) \in \mathit{Init}_{P_1}$, $(k_2, x{\downarrow_{P_2}}) \in \mathit{Init}_{P_2}$ there exists $l_j, y_i, \hat{l}_j, \hat{z}_j$ such that:

- $((k_i, l_j, y_i), (l_i, \hat{l}_j, \hat{z}_i)) \in R_i$,
- $y_i{\downarrow_{P_i}} = x{\downarrow_{P_i}}$,
- $y_i{\downarrow_{Q_j}} = \hat{z}_j{\downarrow_{Q_j}}$, $y_i{\downarrow_{Q_j}} \in \mathit{Init}_{Q_j}$,
- $\hat{z}_i{\downarrow_{Q_i}} \in \mathit{Init}_{Q_i}$.

In a formulation using R_1' and R_2' this means that for all (k_1, k_2, x), $(k_1, x{\downarrow_{P_1}}) \in \mathit{Init}_{P_1}$, $(k_2, x{\downarrow_{P_2}}) \in \mathit{Init}_{P_2}$ there exists (l_1, l_2, z), $(l_1, z{\downarrow_{Q_1}}) \in \mathit{Init}_{Q_1}$, $(l_2, z{\downarrow_{Q_2}}) \in \mathit{Init}_{Q_2}$ such that $((k_i, x{\downarrow_{P_i}}), (l_1, l_2, z)) \in R_i'$ for $i = 1, 2$.

Circular A/G-reasoning is only a sufficient condition with respect to the containment of the initial states. There are cases in which R_1 and R_2 exist, but no simulation relation can be constructed from R_1 and R_2 that contains the initial states appropriately, even though a global R' exists and $P_1||P_2 \preceq Q_1||Q_2$ holds.

A sufficient condition for the containment is that for all (k_1, l_2, y_1) with $(k_1, y_1{\downarrow_{P_1}}) \in \mathit{Init}_{P_1}$, $(l_2, y_1{\downarrow_{Q_2}}) \in \mathit{Init}_{Q_2}$ and $(l_1, v_1) \in \mathit{Init}_{Q_1}$ there exists $(\hat{l}_2, \hat{z}_1)$ with $\hat{z}_1 \downarrow_{Q_1} = v_1$ such that $((k_1, l_2, y_1), (l_1, \hat{l}_2, \hat{z}_1)) \in R_1$. A symmetric argument is also valid for R_2.

Example 4.2: Consider the level monitoring system and its specifications given in Ex. 4.1. Let the parameters be $x_{over} = 200$, $x_m = 20$, $x_M = 180$, $x_l = 30$, $x_h = 176$, $\underline{r}_i = 2$, $\bar{r}_i = 5$, $\underline{r}_d = 1$, $\bar{r}_d = 3$, $\delta = 1$. For an initial set of states $40 \leq x \leq 160$, $d = 0$, the verification is successful. The sets of critical states D_{R_1} and D_{R_2} are empty, and every initial state in $P_1||P_2$ finds a match in the initial states of $Q_1||Q_2$.

V. Experimental Results

The semi-algorithms for checking simulation and A/G-simulation were implemented in C++ as part of a tool for verifying linear hybrid automata called PHAVer, of which an earlier version was presented in [10]. For operations on convex polyhedra it uses the *Parma Polyhedra Library* (PPL) by Roberto Bagnara et al. [12], which employs exact arithmetic with unlimited digits.

An extended version of the level monitoring system was used as a benchmark. The tank model was parameterized by introducing n_T intermediate locations for filling and draining with varying dynamics. The controller was extended by n_C intermediate idle locations, and a min. and max. sampling time. Table I show the results for an Intel Pentium 4M with 1.9GHz, 768MB RAM. With increasing $n = n_T = n_C$ (note that the other parameters change also) the A/G-reasoning (A/G-Sim.) shows a clear advantage over simulation checking of the composed system (Sim.), and even over a convex-hull reachability analysis (Reach.). This correlates with the size of the simulation relations, $|R|$ for the composed analysis and $\sum |R_i| = |R_1| + |R_2|$ for A/G-reasoning, each measured in the number of locations. Note that most of the time in the non-compositional analysis is spent in the composition of the system. However, for $n = 80$, the entire A/G-analysis takes less time than even the net analysis time of the reachability algorithm.

TABLE I
Analysis of an extended Tank Level Monitor

n	Sim.	Reach.	A/G-Sim.	$\|R\|$	$\sum \|R_i\|$
1	0.46 s	0.30 s	1.21 s	4	6
10	10.67 s	3.67 s	5.28 s	183	42
20	33.70 s	14.49 s	9.76 s	490	82
40	197.53 s	109.07 s	19.44 s	2030	162
80	1826.59 s	1217.35 s	43.13 s	9312	318

VI. Conclusion

This paper presents a method for performing compositional reasoning to verify properties of hybrid systems with continuous interactions and proposes a novel, constructive, assume/guarantee rule for hybrid I/O-automata based on simulation relations. The use of labeled transition system semantics leads to semi-algorithms that can be implemented for interesting classes of systems. The importance of assume/guarantee reasoning for hybrid systems is underlined by the fact that shared variables make it often impossible to deduct properties or validate an abstraction of a subsystem on its own, without any assumptions about its inputs. The proposed approach provides a way to include such assumptions at the cost of neglecting the interacting continuous dynamics. While it usually results in an over-approximation, it is in many applications sufficient for showing relevant properties. Experimental results from an implementation of A/G-reasoning for linear hybrid automata show a clear advantage over non-compositional methods when large subsystems are combined. Compositional simulation checking was integrated in the verification tool PHAVer, which is available for download at `http://www.cs.ru.nl/~goranf/`. We are currently evaluating the effectiveness of this approach for examples of increasing complexity. On the theoretical side, we are investigating classes of I/O-automata for which timed transition systems are an exact compositional representation with respect to simulation.

References

[1] N. A. Lynch, R. Segala, and F. Vaandrager, "Hybrid I/O automata," *Information and Computation*, vol. 185, no. 1, pp. 105–157, 2003.

[2] T. Henzinger, "The theory of hybrid automata," in *Proc. of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96)*, New Brunswick, New Jersey, 1996, pp. 278–292.

[3] O. Grumberg and D. E. Long, "Model checking and modular verification," in *CONCUR*, ser. Lecture Notes in Computer Science, J. C. M. Baeten and J. F. Groote, Eds., vol. 527. Springer, 1991.

[4] R. Alur and T. A. Henzinger, "Modularity for timed and hybrid systems," in *Proceedings of the Eighth International Conference on Concurrency Theory (CONCUR)*, ser. LNCS, vol. 1243. Springer, 1997, pp. 74–88.

[5] T. A. Henzinger, M. Minea, and V. Prabhu, "Assume-guarantee reasoning for hierarchical hybrid systems," in *HSCC '01: 4th International Workshop on Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, vol. 2034. Springer, 2001, pp. 275–290.

[6] T. A. Henzinger, S. Qadeer, S. K. Rajamani, and S. Tasiran, "An assume-guarantee rule for checking simulation," *ACM Trans. Program. Lang. Syst.*, vol. 24, no. 1, pp. 51–64, 2002.

[7] G. Frehse, "Compositional verification of hybrid systems using simulation relations," Ph.D. dissertation, Radbout Universiteit Nijmegen, Dec. 2004.

[8] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," in *Workshop on Theory of Hybrid Systems*, ser. LNCS, R. L. Grossman, A. Nerose, A. Ravn, and H. Rischel, Eds., vol. 736. Springer-Verlag, 1993, pp. 209–229.

[9] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995.

[10] G. Frehse, "Compositional verification of hybrid systems with discrete interaction using simulation relations," in *Proc. CACSD'04, Taipei*, 2004.

[11] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke, "Computing simulations on finite and infinite graphs," in *IEEE Symposium on Foundations of Computer Science*, 1995, pp. 453–462.

[12] R. Bagnara, E. Ricci, E. Zaffanella, and P. M. Hill, "Possibly not closed convex polyhedra and the Parma Polyhedra Library," in *Static Analysis: Proc. of the 9th Int. Symposium*, ser. LNCS, M. V. Hermenegildo and G. Puebla, Eds., vol. 2477. Madrid, Spain: Springer-Verlag, 2002, pp. 213–229.