

Towards Component Based Design of Hybrid Systems: Safety and Stability ^{*}

Werner Damm¹, Henning Dierks², Jens Oehlerking¹, and Amir Pnueli^{3†}

¹ Department for Computer Science
University of Oldenburg, Germany

{damm,jens.oehlerking}@informatik.uni-oldenburg.de

² Department of Electrical and Information Engineering
Hamburg University of Applied Sciences, Germany

³ Computer Science Department
Courant Institute of Mathematical Sciences
New York University

Abstract. We propose a library based incremental design methodology for constructing hybrid controllers from a component library of models of hybrid controllers, such that global safety and stability properties are preserved. To this end, we propose hybrid interface specifications of components characterizing plant regions for which safety and stability properties are guaranteed, as well as exception mechanisms allowing safe and stability-preserving transfer of control whenever the plant evolves towards the boundary of controllable dynamics. We then propose a composition operator for constructing hybrid automata from a library of such pre-characterized components supported by compositional and automatable proofs of hybrid interface specifications.

1 Introduction

The use of component-based design approaches for embedded automotive applications has received strong momentum from the establishment of a de-facto standard for automotive component development by the Autosar consortium (see <http://www.autosar.org>). While the current notion of component interfaces in Autosar is rather weak, the overall strategic direction of achieving cost reductions by boosting re-use also of application components is expected to lead to an elaboration of the standard towards richer component interfaces (see e.g. [JMM08], [DMO⁺07]) providing information required for all phases of automotive embedded design flows, notably for establishing safety and real-time requirements. Related projects such as the Integrated Projects Speeds⁴ have provided formal contract based component interface specifications (see [JMM08]) for real-time

^{*} This paper reporting on joint research with Amir Pnueli is dedicated to the memory of Amir Pnueli. It has been partially supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Centre “Automatic Verification and Analysis of Complex Systems” (SFB/TR 14 AVACS).

⁴ IST Project 03347, see www.speeds.eu.com

and safety requirements (as in [DMO⁺07], [DPJ09]). Reflecting the significant share of control applications in embedded automotive development, this paper strives to extend this research by enabling component-based design for hybrid control applications. In particular, we propose a notion of *hybrid interface specifications* which is sufficiently expressive to cover the bridge between specification and implementation models of control, as elaborated below, and provide a framework for hierarchical construction of hybrid controllers supported by automatic verification tools enabling compositional verification of such interface specifications, which subsume both safety and stability properties.

It is in particular

- (1) the need to support not only specification but as well the design phase of such systems,
- (2) and the need to cater for both safety and stability requirements

which require extensions of previous work such as [Fre05, Fre06, Fre08] and [TPL04, HMP01] on compositional verification of hybrid systems. It follows from (1), that models which assume instantaneous reactions on mode-switching conditions, as is typically done in hybrid automata, are inadequate, since mode switching typically entails task-switching and thus comes with significant time penalties. This as well as delays in sensing and actuating the plant caused [JBS07] to propose lazy linear hybrid automata as a more realistic model of hybrid control; this model, however, is not supported by a compositional verification approach. Stauner [Sta01, Sta02] addresses the bridge between specification and design models by proposing a relaxed non-standard semantics of hybrid automata serving as specification models and proposes a systematic approach to derive discrete time design models from specification models such that this robustly refines the specification model. This approach assures that the implementation maintains safety properties, but lacks compositionality. A decompositional approach for verification of stability properties has been developed in [OT09] which constructs Lyapunov functions [Lya07] for individual modes so that their combination yields a Lyapunov function for the complete systems thus establishing stability. This approach is, however, not compositional, in that it assumes the complete system to be given as a basis for decomposition, and does not take implementation aspects into account.

The key achievement of this paper is thus the development of a compositional framework for component based design of hybrid controllers taking into account realistic assumptions about reaction times. This entails the need for what we call *alarms* in hybrid interface specifications, which alert the environment of the component about an encountered plant dynamics, for which stability or safety are endangered, such as through trajectories violating the components assumptions on plant dynamics. Such alarms come with an escape interval, during which safety and stability are still guaranteed by the component itself, thus providing sufficient margin for task-switching. It is this paradigm shift from centralized control of mode-switching to a decentralized setting, where modes take responsibility for creating awareness about the need to perform mode-switching,

which is fundamental for enabling a distributed component based design of control systems. We provide a simple language in the style of Massaccio [HMP01] for the hierarchical construction of hybrid controllers, focusing in this paper on sequential composition: this allows to connect (alarm raising) outputs of components through guarded transitions with inputs of components, whose interface specification explicates assumptions on plant states at entry time so that safety and stability can be guaranteed, or to propagate alarms upward in the hierarchy by connecting these to outputs of the enclosing system. Such port connections are annotated both with guards and jumps. We support distributed implementations of such composed components and give a distributed algorithm for resolving multiple helpful offers in alarm situations. This algorithm identifies a single helpful component, to which control is transferred, supporting in particular the delegation of control to yet unknown helpers in the environment of the composed system. Additional ingredients of interface specifications are plant safety and stability requirements, where we support both asymptotic stability as well as time bounded reachability of plant regions. We describe an approach of turning hybrid automata into a basic component supporting distributed helper identification, and employ fully automatic verification procedures [OT09] for verifying the compliance of such a component realization against its interface specification, for dynamics given as linear differential systems of equations, linear convex guards, and linear jumps, in particular relying on automatic synthesis of parameterized families of Lyapunov functions. For composed components, we give automatically verifiable verification conditions jointly ensuring, that local interface specifications augmented with auxiliary invariants such as local parameterized Lyapunov functions imply the interface specification of the composed system. These verification conditions can be seen as generating a constraint systems on parameters of local Lyapunov functions – if this constraint system is solvable, the combined system will meet its stability specification.

This paper is organized as follows. The subsequent section introduces our version of hybrid automata allowing super-dense discrete time switches as well as their parallel composition. Section 3 gives syntax and semantics of hybrid interface specifications, shows how to build basic components from hybrid automata, and provides syntax and semantics for sequential composition of components, incorporating the distributed protocol for helper election. Section 4 shows how to establish the correctness of interface specifications automatically, first for basic components, and then for composed systems. We use as running example a simple automatic cruise control (ACC) system to illustrate our approach. The conclusion summarizes the key achievements and explains directions of further research.

2 Basic Definitions

In the following, we will define a hybrid automaton formalism as required for the compositional design methodology, differentiating between input variables

that are assumed to be outside the control of the automaton, and controllable variables that are either local, or outputs variables of the automaton.

Definition 1 (Hybrid Automaton with Inputs). A hybrid automaton is a tuple

$$H = (\mathbb{M}, \text{Var}^{loc}, \text{Var}^{in}, \text{Var}^{out}, R^D, R^I, R^C, \Phi, \Theta)$$

where

1. \mathbb{M} is a finite set of modes,
2. $\text{Var}^{loc}, \text{Var}^{in}$ and Var^{out} are disjoint sets of local, input and output variables over \mathbb{R} , denote $\text{Var} = \text{Var}^{loc} \cup \text{Var}^{in} \cup \text{Var}^{out}$,
3. Φ is a first-order predicate over Var and a variable M , which takes values in \mathbb{M} , describing all combinations of initial states and modes,
4. Θ is a mapping that associates with each mode $m \in \mathbb{M}$ a local invariant $\Theta(m)$, which is a quantifier-free formula over Var ,
5. R^D is the discrete transition relation with elements $(m, \Phi, \mathcal{A}, m')$ called transitions, which are graphically represented as $m \xrightarrow{\phi, \mathcal{A}} m'$, where
 - $m, m' \in \mathbb{M}$,
 - ϕ is a first-order predicate over Var ,
 - \mathcal{A} is a first-order predicate over $\text{Var} \cup \text{Var}^{loc'} \cup \text{Var}^{out'}$ specifying discrete updates, where $\text{Var}^{loc'}$ and $\text{Var}^{out'}$ are the primed variants of the variables in Var^{loc} and Var^{out} , R^D consists of two disjoint sets R_U^D and R_L^D . The subset R_U^D contains the urgent transitions, and the subset R_L^D the lazy transitions.
6. R^I is a first-order predicate over Var^{in} and $(\text{Var}^{in})^\bullet$ of the form $\bigwedge v^\bullet \bowtie r$, where $v \in \text{Var}^{in}, \bowtie \in \{\leq, =, \geq\}, r \in \mathbb{R}$. It defines the differential inclusion modeling the evolution of the input signals.
7. R^C is a first-order predicate over Var and $(\text{Var}^{loc})^\bullet \cup (\text{Var}^{out})^\bullet$ of the form $\bigwedge v^\bullet \bowtie e$, where $v \in \text{Var}^{loc} \cup \text{Var}^{out}, \bowtie \in \{\leq, =, \geq\}$, and e is a real linear arithmetic expression over Var . It defines the differential inclusion modeling the continuous transition relation.

The discrete update predicate will also often be implicitly defined by a sequence of assignments of the form $v := e$, with $v \in \text{Var}^{loc} \cup \text{Var}^{out}$ and e an expression over Var . We identify such a sequence of assignments with the predicate relating the pre- and post-states of its sequential execution. For graphical representation, lazy transitions will be labeled in the form ϕ/\mathcal{A} , and urgent transitions $\uparrow \phi/\mathcal{A}$, with \mathcal{A} either in predicate or assignment notation. If a set of assignments is empty, it will be left out, meaning that all variables in Var remain unchanged upon switching. For a predicate ϕ , $\phi[\mathcal{A}]$ is the result of the substitution induced by \mathcal{A} on ϕ . Discrete variables may be included into hybrid automata according to our definition via an embedding of their value domain into the reals, and associating a derivative of constantly zero to them (locals and outputs). Timeouts are easily coded via explicit local timer variables with a derivative taken from $\{-1, 0, 1\}$.

2.1 Behavior

We now give the formal definition of runs of a hybrid automaton H capturing the evolution of modes and the real-valued variables over time. To this end, we consider the continuous time domain $Time = \mathbb{R}_{\geq 0}$ of non-negative reals, for the mode observable a function $M : Time \rightarrow \mathbb{M}$, and for the vector of variables in Var a corresponding function $X : Time \rightarrow \mathbb{R}^{|Var|}$, with $X(t) = [X^C(t)^T, X^I(t)^T]^T$, describing for each time point $t \in Time$ the current mode m and the current value of all variables in Var , respectively. Here, X^C covers all variables in $Var^{loc} \cup Var^{out}$ and X^I all variables in Var^{in} . Furthermore, the vector concatenation $\pi(t) = [M(t), X^C(t)^T, X^I(t)^T]^T$ describes the overall (hybrid) state of H . The order of variables in each of the two sub-vectors is fixed, but arbitrary. We identify the state vector $\pi(t)$ with the corresponding valuation of the variables in $M \cup Var$. For simplicity, for a predicate Ψ , we use the notation $\pi(t) \models \Psi$, if the valuation associated with $\pi(t)$ fulfills Ψ . We also define a substitution based on vectors, such that the predicate $\Psi[Var/X(t)]$ is the predicate Ψ with the variable values given by the valuation $X(t)$ substitute the corresponding variables in Var . The vectors $X^C(t)$ and $X^I(t)$ are handled in the same manner. The *time derivative* of $X(t)$ is denoted $dX/dt(t)$ or $X^\bullet(t)$.

Definition 2 (Runs of a Hybrid Automaton). A run of a hybrid automaton

$$H = (\mathbb{M}, Var^{loc}, Var^{in}, Var^{out}, R^D, R^I, R^C, \Phi, \Theta)$$

corresponding to a sequence of switching times $(\tau_i)_{i \in \mathbb{N}} \in Time^{\mathbb{N}}$ with

$$\tau_0 = 0 \wedge \forall i \in \mathbb{N} : \tau_i \leq \tau_{i+1}$$

is a sequence of tuples (π_i) ,

$$\pi_i = \begin{bmatrix} M_i \\ X_i \end{bmatrix}, \text{ with } X_i = \begin{bmatrix} X_i^C \\ X_i^I \end{bmatrix},$$

where $M_i : [\tau_i, \tau_{i+1}] \rightarrow \mathbb{M}$, $X_i^C : [\tau_i, \tau_{i+1}] \rightarrow \mathbb{R}^n$, and $X_i^I : [\tau_i, \tau_{i+1}] \rightarrow \mathbb{R}^k$ are continuously differentiable functions such that

- (1) *initial state:* $\pi_0(0) \models \Phi$
- (2) *non-Zeno:* $\forall t \in Time \exists i \in \mathbb{N} : t \leq \tau_i$
- (3) *mode switching times:* $\forall i \in \mathbb{N} \forall t \in [\tau_i, \tau_{i+1}) : M_i(t) = M(\tau_i)$
- (4) *continuous evolution:* $\forall i \in \mathbb{N} \forall t \in (\tau_i, \tau_{i+1}) :$
 $(dX_i^C/dt(t), X_i(t)) \models R^C(M_i(\tau_i))$
- (5) *input evolution:* $\forall i \in \mathbb{N} \forall t \in Time : (dX_i^I/dt(t), X_i^I(t)) \models R^I$
- (6) *invariants:* $\forall i \in \mathbb{N} \forall t \in Time : X_i(t) \models \Theta(M_i(t))$
- (7) *urgency:* $\forall i \in \mathbb{N} \forall t \in [\tau_i, \tau_{i+1}) \forall (M_i(t), \phi, \mathcal{A}, m') \in R_U^D$ we have that $X_i(t) \not\models \phi$.

(8) *discrete transition firing*: $\forall i \in \mathbb{N}$:

$$\begin{aligned}
& (M_i(\tau_{i+1}) = M_{i+1}(\tau_{i+1}) \wedge X_i(\tau_{i+1}) = X_{i+1}(\tau_{i+1})) \\
& \vee (\exists (m, \phi, \mathcal{A}, m') \in R^D : M_i(\tau_{i+1}) = m \wedge M_{i+1}(\tau_{i+1}) = m' \\
& \quad \wedge X_i(\tau_{i+1}) \models \phi \\
& \quad \wedge \models \mathcal{A}[Var^{loc'} \cup Var^{out'} / X_{i+1}^C(\tau_{i+1}), Var / X_i(\tau_{i+1})] \\
& \quad \wedge X_i^I(\tau_{i+1}) = X_{i+1}^I(\tau_{i+1})).
\end{aligned}$$

Define $\pi(t)$ as the $\pi_i(t)$, such that $\forall j > i : \tau_j > t$, i.e., $\pi(t)$ is the system state after all (possibly superdense) switches that occur at time t . Define $X(t), M(t), X^C(t)$ and $X^I(t)$ in the same manner.

Clause (1) stipulates that a run must start with an allowed initial state. The time sequence $(\tau_i)_{i \in \mathbb{N}}$ identifies the points in time, at which mode-switches may occur, which is expressed in Clause (3). Only at those points discrete transitions (having a noticeable effect on the state) may be taken. On the other hand, it is not required that any transition fires at some point τ_i , which permits to cover behaviors with a finite number of discrete switches within the framework above. Our simple plant models with only one mode provide examples. As usual, we exclude non-Zeno behavior (in Clause (2)). Clauses (4) and (5) force all variables to actually obey their respective differential inclusions. Clause (6) requires, for each mode, the valuation of continuous variables to meet both local and global invariants while staying in this mode. Clause (7) forces an urgent discrete transition to fire when its trigger condition becomes true. The effect of a discrete transition is described by Clause (8). Whenever a discrete transition is taken, local and output variables may be assigned new values, obtained by evaluating the right-hand side of the respective assignment using the previous value of locals and outputs and the current values of the input. If there is no such assignment, the variable maintains its previous value, which is determined by taking the limit of the trajectory of the variable as t converges to the switching time τ_{i+1} .

Definition 3 (Reach Set). For some $t \geq 0$, define a time bounded reach set $reach(H, \Phi, t)$ of hybrid automaton H from predicate Φ as the closure of

$$\{X \mid \exists \text{ trajectory } X(\cdot) \text{ of } H, t \geq t' \geq 0 : X(0) \models \Phi \wedge X(t') = X\}.$$

Analogously, define the unbounded reach set $reach(H, \Phi)$ of hybrid automaton H from predicate Φ as the closure of

$$\{X \mid \exists \text{ trajectory } X(\cdot) \text{ of } H, t' \geq 0 : X(0) \models \Phi \wedge X(t') = X\}.$$

2.2 Parallel Composition

Output variables of H_1 which are at the same time input variables of H_2 , and vice versa, establish communication channels with instantaneous communication. Those variables establishing communication channels remain output variables of $H_1 \parallel H_2$. Modes of $H_1 \parallel H_2$ are the pairs of modes of the component automata.

Definition 4 (Parallel Composition). *Let two hybrid automata*

$$H_i = (\mathbb{M}_i, \text{Var}_i^{\text{loc}}, \text{Var}_i^{\text{in}}, \text{Var}_i^{\text{out}}, R_i^D, R_i^I, R_i^C, \Phi_i, \Theta_i),$$

$i = 1, 2$ be given with $\text{Var}_1^{\text{loc}} \cap \text{Var}_2^{\text{loc}} = \emptyset$. Then the parallel composition

$$H_1 \parallel H_2 = (\mathbb{M}, \text{Var}^{\text{loc}}, \text{Var}^{\text{in}}, \text{Var}^{\text{out}}, R^D, R^I, R^C, \Phi, \Theta)$$

is given by:

- $\mathbb{M} = \mathbb{M}_1 \times \mathbb{M}_2$,
- $\text{Var}^{\text{loc}} = \text{Var}_1^{\text{loc}} \cup \text{Var}_2^{\text{loc}}$,
- $\text{Var}^{\text{out}} = \text{Var}_1^{\text{out}} \cup \text{Var}_2^{\text{out}}$,
- $\text{Var}^{\text{in}} = (\text{Var}_1^{\text{in}} \cup \text{Var}_2^{\text{in}}) - \text{Var}^{\text{out}}$,
- $R^C((m_1, m_2)) = R_1^C(m_1) \wedge R_2^C(m_2)$
- $R^I = R_1^I \wedge R_2^I$,
- R_U^D consists of the following transitions:
 - (1) $((m_1, m_2), \Phi_1, \mathcal{A}_1, (m'_1, m_2))$ for each $(m_1, \Phi_1, \mathcal{A}_1, m'_1) \in R_{U,1}^D$, and
 - (2) transitions of the form (1) with the role of H_1 and H_2 interchanged,
- R_L^D consists of the following transitions:
 - (1) $((m_1, m_2), \Phi_1, \mathcal{A}_1, (m'_1, m_2))$ for each $(m_1, \Phi_1, \mathcal{A}_1, m'_1) \in R_{L,1}^D$, and
 - (2) transitions of the form (1) with the role of H_1 and H_2 interchanged,
- $\Phi = \Phi_1 \wedge \Phi_2$,
- $\Theta((m_1, m_2)) = \Theta_1(m_1) \wedge \Theta_2(m_2)$.

3 Hierarchical Controller Design

In this chapter we elaborate our approach towards component based design of hybrid controllers. We use a simplified automatic cruise control application to illustrate our design methodology and the supporting formal definitions. To simplify the exposition, we restrict ourselves to controller design for a fixed, given plant – a generalization of the approach would simply take the reference plant specification as an additional parameter of hybrid interface specifications.

Formally, a plant specification is just a single-mode hybrid automaton extended with specifications of safety and stability conditions of the plant. Its input variables subsume the set of actuators, whose valuations are determined by the controller based on the current observable state of the plant as given by a valuation of its sensors. Note that we allow additional input variables to the plant – these correspond to what is typically called *disturbances*. The rate of change of these can be restricted by associated differential inclusions, while bounds on these can be expressed as invariance properties of the plant model. Stability requirements are expressed in terms of the interface variable of the plant and allow to specify a (convex) stability region subsuming the intended point of equilibrium.

Definition 5 (Plant). A plant is a hybrid automaton

$$P = (\mathbb{M}_P, \emptyset, \text{Var}_P^{\text{in}}, \text{Var}_P^{\text{out}}, R_P^D, R_P^C, R_P^I, \Phi_P, \Theta_P)$$

with

- an open first-order predicate φ_P^{safe} over $\text{Var}_P^{\text{in}} \cup \text{Var}_P^{\text{out}}$ describing the safety constraints of the plant,
- a convex, open first-order predicate $\varphi_P^{\text{stable}}$ over $\text{Var}_P^{\text{in}} \cup \text{Var}_P^{\text{out}}$ describing the system stability condition.

We also define a set of sensor variables $S \subseteq \text{Var}_P^{\text{out}}$ and a set of actuator variables $A \subseteq \text{Var}_P^{\text{in}}$.

Example For the simple ACC system, we restrict ourselves to a simple plant allowing to directly influence the acceleration a of the car, which is only perturbed through a disturbance s . The velocity v is observable by the controller. To simplify the discussion, we assume a fixed set point v_{desired} , and let v denote the difference between the actual velocity v_{actual} and the desired velocity v_{desired} . The stability requirement thus specifies, that v should be close to zero, while the plant is considered to be in an unsafe state if the deviation from desired actual velocity exceeds 35m/sec . Define

$$P = (\{m_P\}, \emptyset, \{a, s\}, \{v\}, \emptyset, R_P^C, R_P^I, \text{true}, \Theta_P)$$

with

- $R_P^C(m_P) = (v^\bullet = sa)$
- $R_P^I(m_P) = \text{true}$
- $\Theta_P(m_P) = 0.975 \leq s \leq 1.025$,

$$\varphi_P^{\text{safe}} = (-35 < v < 35), \text{ and } \varphi_P^{\text{stable}} = (-2 < v < 2).$$

Example (cont.) To motivate our approach to component based design of hybrid controllers, consider the design of an ACC controller C controlling the above plant, given by the following specification:

Comp.	Var^{in}	Var^{out}	φ^{assm}	φ^{prom}	Δ^{stable}
C	$\{v\}$	$\{a\}$	$-30 \leq v \leq 30$	$-2 \leq v^\bullet \leq 1.5$	300

The controller is employed in the design setting shown in Figure 1. It is required to drive the plant into its stability region within 300 seconds, for all plants states which deviate from the desired velocity v_{desired} by at most 30m/sec ., and as long as disturbances are bounded by the plant invariant. We are also looking for an implementation which meets comfort requirements, here simplified to a maximal deceleration of 2m/sec^2 , and a maximal acceleration of 1.5m/sec^2 . Entry conditions shown in Figure 1 will be discussed below.

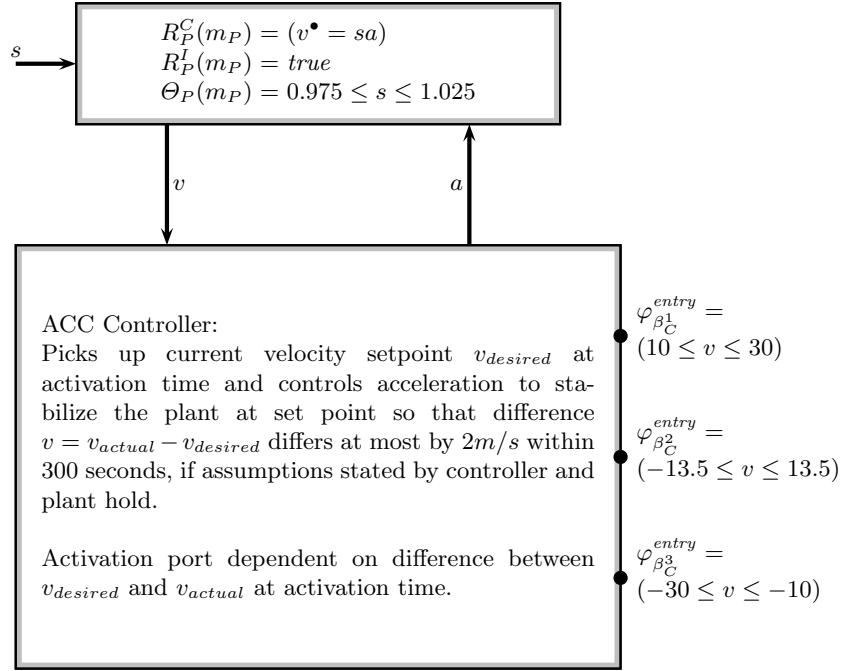


Fig. 1. Controller Design Setting

We envision a future design process for such embedded control applications which is supported by design libraries, whose components encapsulate “standard” control solutions. A designer would then check the hybrid interface specifications for a possible control component supporting the ACC requirement specification shown in Figure 1.

Let us assume that he finds the following component specification of a component named *PI*.

Comp.	Var^{in}	Var^{out}	φ^{assm}	φ^{prom}	Δ^{stable}
<i>PI</i>	$\{v\}$	$\{a\}$	$-15 \leq v \leq 15$	$-1.4 \leq v^{\bullet} \leq 1.4$	300

This component is offering a convergence time to the stability region which matches the specifications, although only for a restricted subspace of the plant. It employs an acceleration which matches those of the specification, and thus altogether looks like a good starting point for the controller design, if we can extend the scope of the controllable plant region with some “glue” control.

This takes us to a central point of our design methodology. Since we are addressing, as in AUTOSAR, distributed control applications, where different subcomponents of controllers are running on different electronic control units, we can no longer as in hybrid automata rely on a centralized control structure

ensuring mode switching. Instead, in this distributed setting, it becomes the duty of components to raise an alarm if the dynamics of the plant is evolving in an unforeseen way (such as through differences between the idealized plant model and the actual controlled plant, e.g., unmodeled disturbances). This must happen in time to allow a control component capable of addressing the critical dynamics to take control.

We thus extend our interface concept by allowing the declaration of possibly multiple outputs encapsulating possibly different types of dynamics raising *alarms*. Such output specifications signal the plant state causing the alarm, and provide time-guarantees for maintaining stability and safety for a rescue period, thus providing a time-window in which the switch of control can be initiated. To support a distributed agreement protocol for the selection of the as it were “helper component”, there is a persistency requirement on such alarm signals. They also exhibit the plant state at the latest allowable time for control-switch, as determined by the duration of the rescue period and the cost for control switching, which we assume to be given as a design parameter τ .

Example (cont.) For the PI control component, we have two sources of endangering the component’s promises in maintaining safety and stability, in that the vehicle could become either significantly slower or faster than the desired speed, as catered for in the following two output specifications:

Output	Comp.	$\varphi_\alpha^{alarmOn}$	μ_α	Δ_α	φ_α^{exit}
α_{PI}^1	PI	$v \geq 14$	0.006	0.01	$13.9 \leq v \leq 14.1$
α_{PI}^2	PI	$v \leq -14$	0.006	0.01	$-14.1 \leq v \leq -13.9$

By way of returning to our design scenario, let us assume that the design library offers a component which addresses traffic situations, where the actual velocity is significantly below the desired velocity, a simple acceleration component *ACCELERATE* with a still acceptable constant acceleration, which could recover from such plant states and force the plant into regions allowing a more fine-grained control, as exemplified below.

Comp.	Var^{in}	Var^{out}	φ^{assm}	φ^{prom}	Δ^{stable}
<i>ACCELERATE</i>	$\{v\}$	$\{a\}$	$-30 \leq v \leq -5$	$v^\bullet = 1.5$	300

The *ACCELERATE* component raises an alarm if its assumptions are endangered:

Output	Comp.	$\varphi_\alpha^{alarmOn}$	μ_α	Δ_α	φ_α^{exit}
$\alpha_{ACCELERATE}$	<i>ACCELERATE</i>	$v \geq -6$	0.005	0.01	$v \geq -6$

Intuitively, the combination of the *ACCELERATE* component with the *PI* component yields a more robust system, in that safety and stability are now guaranteed for a larger plant region.

We have now motivated most concepts occurring in a hybrid interface specification. The one remaining concept is that of *inports*: these serve to activate components resp. resume execution of components, under specified entry conditions, which jointly cover the assumptions on plant states for which this component guarantees safety and stability. Each inport specification defines as well a time-window in which it promises to respond to rescue requests arriving at this port.

Example (cont.) The *PI* component offers a single inport Alarm requests are answered within a time window of 0.0025 seconds.

Inport	Comp.	λ_β	φ_β^{entry}
β_{PI}	<i>PI</i>	0.0025	$-13.5 \leq v \leq 13.5$

To summarize, a hybrid interface specification for a given plant model P consists of

- a static interface definition of real-valued data interface variables and boolean control variables,
- specification of inports and outports, which in addition to the concepts elaborated in the example also define control signals used for distributed agreement in context-switching, as elaborated below,
- a specification of the plant states for which this component guarantees safety, stability, and promises,
- promises on the rate of change of out-variables,
- a maximal time to convergence to the plant stability region.

Definition 6 (Component Interface Specification). A component C associated with a plant P is described by an externally visible interface $SPEC_C$ consisting of:

- Var_C^{in} , a set of real valued input variables with $S \subseteq Var_C^{in}$,
- Var_C^{out} , a set of real valued output variables which is disjoint from Var_C^{in} and with $A \subseteq Var_C^{out}$,
- $C_C^{in} = \{suspend_C\} \cup \{c_\beta, start_\beta \mid \beta \in A_C^{in}\} \cup \{take_\alpha \mid \alpha \in A_C^{out}\}$, a set of binary control inputs,
- $C_C^{out} = \{active_C, fail_C\} \cup \{take_\beta \mid \beta \in A_C^{in}\} \cup \{b_\alpha, switch_\alpha \mid \alpha \in A_C^{out}\}$, a set of binary control outputs,
- a set A_C^{in} of incoming ports (“inports”) given as tuples

$$\beta = (c_\beta, \lambda_\beta, take_\beta, start_\beta, \varphi_\beta^{entry}),$$

where $c_\beta \in C_C^{in}$, $\lambda_\beta > 0$, $take_\beta \in C_C^{out}$, $start_\beta \in C_C^{in}$, and φ_β^{entry} is a first-order predicate over $Var_C^{in} \cup Var_C^{out}$,

- a set A_C^{out} of outgoing ports (“outports”) given as tuples

$$\alpha = (b_\alpha, \varphi_\alpha^{alarm}, \mu_\alpha, \Delta_\alpha, take_\alpha, switch_\alpha, \varphi_\alpha^{exit}),$$

where $b_\alpha \in C_C^{out}$, φ_α^{alarm} is a closed first-order predicate over $Var_C^{in} \cup Var_C^{out}$, $\mu_\alpha > 0$, $\Delta_\alpha > 0$, $take_\alpha \in C_C^{in}$, $switch_\alpha \in C_C^{out}$, and φ_α^{exit} is a first-order predicate over $Var_C^{in} \cup Var_C^{out}$,

- φ_C^{prom} , a first-order predicate over $Var_C^{in\bullet} \cup Var_C^{out\bullet} \cup Var_C^{in} \cup Var_C^{out}$,
- φ_C^{assm} , a first-order predicate over $Var_C^{in} \cup Var_C^{out}$,
- $\Delta_C^{stable} > 0$ is a time after which the system is required to converge to φ_P^{stable} .

In the rest of this paper we use φ_C^{entry} to abbreviate $\bigvee_{\beta \in A^{in}} \varphi_\beta^{entry}$.

Definition 7 (Switch Time). Define a global variable $0 < \tau \in \mathbb{R}$, representing the maximum time needed for a component switch.

We will now discuss how to build controllers hierarchically, and use the running ACC controller design to illustrate the key underlying concepts and issues.

Example (cont.)

Comp.	Var^{in}	Var^{out}	φ^{assm}	φ^{prom}	Δ^{stable}
<i>PI</i>	$\{v\}$	$\{a\}$	$-15 \leq v \leq 15$	$-1.4 \leq v^\bullet \leq 1.4$	300
<i>ACCELERATE</i>	$\{v\}$	$\{a\}$	$-30 \leq v \leq -5$	$v^\bullet = 1.5$	300

Output	Comp.	$\varphi_\alpha^{alarmOn}$	μ_α	Δ_α	φ_α^{exit}
α_{PI}^1	<i>PI</i>	$v \geq 14$	0.006	0.01	$13.9 \leq v \leq 14.1$
α_{PI}^2	<i>PI</i>	$v \leq -14$	0.006	0.01	$-14.1 \leq v \leq -13.9$
$\alpha_{ACCELERATE}$	<i>ACCELERATE</i>	$v \geq -6$	0.005	0.01	$v \geq -6$

Inport	Comp.	λ_β	φ_β^{entry}
β_{PI}	<i>PI</i>	0.0025	$-13.5 \leq v \leq 13.5$
$\beta_{ACCELERATE}$	<i>ACCELERATE</i>	0.0025	$-30 \leq v \leq -10$

We compose the *PI* and *ACCELERATE* components as follows:

- an alarm raised by *ACCELERATE* is forwarded to the inport of the *PI* component,
- an alarm raised by the *PI* component of type “actual speed is too slow to be handled by *PI* component” is forwarded to the inport of the *PI* component.

Note that such a composition leaves unspecified, how to handle plant dynamics, in which the current speed is much too fast in relation to the desired speed for the *PI* component to maintain stability. In general, the composition of two components then yields a composed component, whose outputs must cater for alarms which are not handled internally. Figure 2 gives an informal description of the composition of the *PI* component and the *ACCELERATE* component.

We can now perform a what in industrial design processes is typically referred to as a *virtual integration test*:

- Is the plant state at context switch time as given by the exit predicate of the output compatible with the plant state required by the connected inport?
- Does the inport have sufficient time to take a decision on whether it is willing to accept the alarm? To this end we compare the promised minimal stability period μ_α of output, as expressed by the λ_β which provides an upper bound for the inport to reply to incoming alarms.

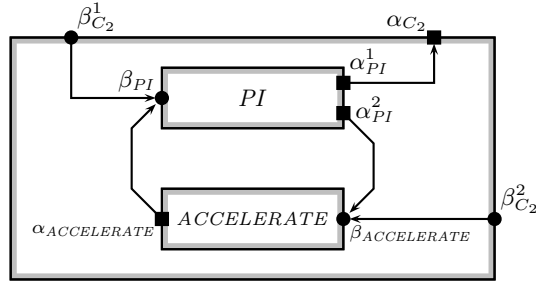


Fig. 2. Interconnection of *ACCELERATE* and *PI*

Having successfully carried out the virtual integration test, we can now either derive a component interface specification for the composed system C_2 , by propagating information derived from local specifications towards the boundary of the system, or check, whether an a priori given specification of the composed system C_2 is derivable from local specifications. This reasoning will be formalized in Section 4 on verification of hierarchical component based controller designs. The key property conspicuous already in our simple example is, that this reasoning is completely independent of the actual implementation of the subsystems. Indeed, the *PI* component might itself be composed of several subsystems, or be given as a what we call *basic* component: both the virtual integration test as well as compliance of the composed system to an interface specification of the composed system is purely based on component interface specifications. Industrial jargon uses the term *grey box view* to refer to schematics of a composed system as in Figure 2, where only the interface specifications of subsystem as well as their interconnection via ports are known. In contrast, a *white box view* of a composed system would also make visible the internal realization of the subsystems, across all levels of hierarchy. We will use a white box view on composed systems to define their semantics, and a grey box view in Section 4 for compositional verification of such systems.

As in the ACC example, components of a composed system offer different capabilities in establishing safety and stability requirements of one and the same plant. The formal definition of what we call the transition composition of components exhibits the following aspects.

- Alarms may be either be handled locally of forwarded to a yet unknown environment.
- We allow for multiple helpers, and offer guards on port connections to filter propagation of alarms.
- On the path to local helpers, interface variables of the entered component may be updated (thus motivating the usage of term *transition composition*).
- Statically it must be possible to have at least one feasible path to ask for help: the disjunction of all guard conditions related to a single outputport must be a tautology.

In Def. 6 we defined the externally visible interface of a component C . Components may also use *local* variables internally and we denote the set of these variables with Var_C^{loc} . In the following definition we use $C(\alpha)$ (resp. $C(\beta)$) to denote the component C the port α (resp. β) belongs to, i.e. $\alpha \in A_C^{out}$ (resp. $\beta \in A_C^{in}$).

Definition 8 (Transition Composition of Components). *Let C be a component and let $\{C_1, \dots, C_n\}$ be a finite set of basic or composed components with*

- $Var_C^{in} = Var_{C_i}^{in}$ for all i ,
- $Var_C^{out} = Var_{C_i}^{out}$ for all i ,
- all $A_{C_i}^{out}, A_{C_i}^{in}$, as well as A_C^{out} and A_C^{in} are disjoint.

We define $Var_C^{loc} := \bigcup_i Var_{C_i}^{loc}$ and call C a transition composition of the components C_1, \dots, C_n with port connection $(\mathcal{P}, \mathcal{Q})$ (we use $\mathcal{S}_{(\mathcal{P}, \mathcal{Q})}(C_1, \dots, C_n)$ to denote it) iff

(a) \mathcal{P} is given as a set of tuples describing transitions of the form

$$(\alpha, \{(\beta_1, g_1, \mathcal{A}_1), \dots, (\beta_k, g_k, \mathcal{A}_k)\}, \{(\alpha_1, g'_1), \dots, (\alpha_l, g'_l)\}),$$

with

- $\alpha \in \bigcup_i A_{C_i}^{out}$,
 - $k, l \geq 0, k + l > 0$,
 - $\beta_i \in \bigcup_i A_{C_i}^{in}$ with $\beta_i \neq \beta_j$ for all $i \neq j \in \{1, \dots, k\}$,
 - $\alpha_i \in A_C^{out}$ with $\alpha_i \neq \alpha_j$ for all $i \neq j \in \{1, \dots, l\}$,
 - g_i and g'_i are first-order predicates over $Var_C^{in} \cup Var_C^{loc} \cup Var_C^{out}$, such that for each tuple $\bigvee_i g_i \vee \bigvee_i g'_i$ holds,
 - for all $(\alpha, \{(\beta_1, g_1, \mathcal{A}_1), \dots, (\beta_k, g_k, \mathcal{A}_k)\}, \{(\alpha_1, g'_1), \dots, (\alpha_l, g'_l)\}) \in \mathcal{P}$, α belongs to a different component than all β_i (no loops, $C(\alpha) \neq C(\beta_i)$),
 - \mathcal{A}_i is a set of assignments for $Var_{C(\alpha_i)}^{in} \setminus S$ depending on $Var_{C(\alpha)}^{in} \cup Var_{C(\alpha)}^{out}$,
 - for all α , there exists exactly one tuple $(\alpha, S_1, S_2) \in \mathcal{P}$ (each outgoing alarm is connected to exactly one family of incoming alarms).
- (b) The second component \mathcal{Q} of the port connection is a totally defined mapping from A_C^{in} to $\bigcup_i A_{C_i}^{in}$.

This definition connects the components C_1, \dots, C_n that have equal Var^{in} and Var^{out} and the result is a component C . The composition C and its components C_1, \dots, C_n carry control in- and outputs that are connected appropriately in the above definition. A tuple

$$p = (\alpha, \{(\beta_1, g_1, \mathcal{A}_1), \dots, (\beta_k, g_k, \mathcal{A}_k)\}, \{(\alpha_1, g'_1), \dots, (\alpha_l, g'_l)\}),$$

connects an output α of a component C_i with some inports β_i of the other components and with some outputs α_j of the composition C . The idea is that the outgoing signal of α is forwarded to the β_i 's and α_j 's such that each receiver of this signal is able to respond appropriately and to take over. However, this action

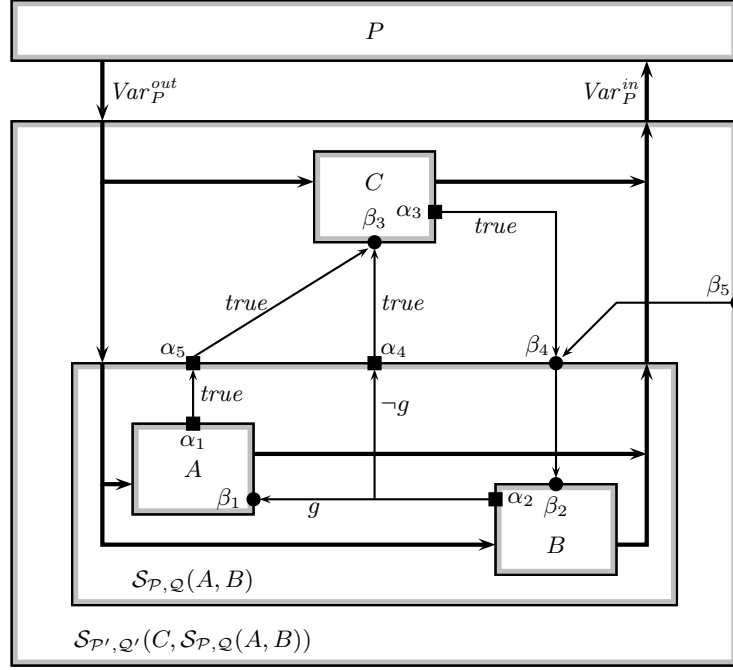
requires that the corresponding entry condition of the activated component is met. Hence, the composition must be able to ensure that and therefore p adds a guard for each receiving port. Moreover, it adds assignments to each connection between α and an inport β_i of another component. This allows for setting non-sensor variables as required. The above definition lists sanity conditions for a port connection p . There must be at least one receiver, no inport is used twice as receiver, no outputport is used twice as receiver, there is always at least one of the guards satisfied, and there are no loops, i.e., no component receives its own outgoing signal. All these tuples are collected in the set \mathcal{P} and for each outputport that appears in C_1, \dots, C_n we have exactly one $p \in \mathcal{P}$ describing the receivers of this signal.

The inports of the composition are receiving signals and therefore this information must be forwarded to some inports of its components. In the above definition this is done by an one-to-one mapping \mathcal{Q} . In Fig. 3 an example for transition composition is given. It consists of a composition $\mathcal{S}_{\mathcal{P}, \mathcal{Q}}(A, B)$ of A and B which is then composed with C .

We now return to the role of the control signals in component interface specifications. Consider the transition composition of components A , B , and C in Figure 3, and assume a distributed implementation, where A and B are allocated on ecu_1 , and C is allocated on ecu_2 , and consider an alarm raised by component B at outputport α_2 . We will now informally describe a distributed agreement protocol, ensuring that all ECUs agree on the component to handle the alarm, called *distributed identification of helpers*. The challenge arises from the distributed execution, and the possibility of multiple helpers. Much as in distributed cache coherency protocols, we need to enforce a serialization point so as to avoid race conditions leading to inconsistent states, where say both A and C believe to be the chosen helper. We describe the protocol informally using the sequence chart given in Figure 4. Formally, we will define with each component wrapper automata jointly implementing the protocol. Such wrapper automata are constructed both for outputports – such as $H_3(B)$ –, for port connections – such as $H_{\mathcal{P}}$ and $H_{\mathcal{P}'}$ –, and for inports, such as $H_{\beta_1}(A)$ and $H_{\beta_2}(C)$ in Figure 4.

The protocol is initiated from the basic component B : it raises an alarm at its outputport α_2 , which causes the control signal b_{α_2} associated with this outputport to be generated. There are two levels of hierarchy enclosing component B , each specifying through port connections potential helper components. From the inner hierarchy, we see that either the alarm can be handled locally – this is reflected by requesting help from inport β_1 of component A through generation of the control signal c_{β_1} associated with this inport, or externally, which is represented by generating an alarm at the outputport α_4 of the composed system, represented by setting its control signal to 1. This is handled by the wrapper automaton $H_{\mathcal{P}}$ for this port connection running on ecu_1 .

We now have two independent message flows – in which the generated control signals are propagated either locally within ecu_1 or externally to ecu_2 (typically with different arrival times). The local flow will lead to a response of component



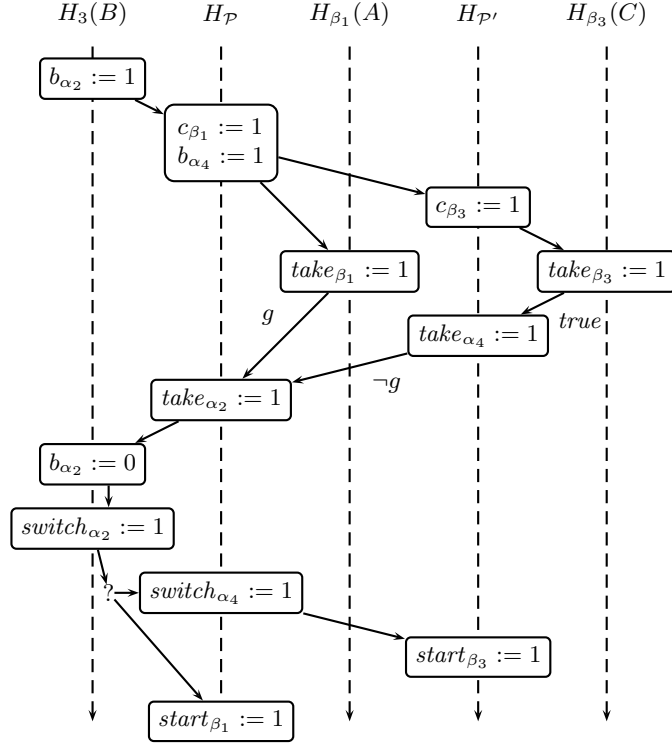
Bottom: $\mathcal{S}_{\mathcal{P}, \mathcal{Q}}(A, B)$ with $\mathcal{P} = \left\{ (\alpha_1, \emptyset, \{(\alpha_5, true)\}), (\alpha_2, \{(\beta_1, g)\}, \{(\alpha_4, \neg g)\}) \right\}$ and $\mathcal{Q}(\beta_4) = \beta_2$.

Overall: $\mathcal{S}_{\mathcal{P}', \mathcal{Q}'}(C, \mathcal{S}_{\mathcal{P}, \mathcal{Q}}(A, B))$ with $\mathcal{P}' = \left\{ (\alpha_3, \{(\beta_4, true, \emptyset)\}, \emptyset), (\alpha_4, \{(\beta_3, true, \emptyset)\}, \emptyset), (\alpha_5, \{(\beta_3, true, \emptyset)\}, \emptyset) \right\}$ and

$\mathcal{Q}'(\beta_5) = \beta_4$.

Fig. 3. An example for a composition

A through its inport β_1 to be ready for taking over, as indicated by setting the associated control signal $take_{\beta_1}$. The external flow will lead to the generation of a help request (control signal c_{β_3}) to the inport of component C , generated by a wrapper automaton interpreting the port connection \mathcal{P}' of the outer hierarchy level. Component C will indicate its readiness to accept the alarm by setting the control variable $take_{\beta_3}$. Outports of composed components act as a proxy for the environment, in that the existence of at least one helper component in the environment of the sequential composition of A and B , such as component C , is then communicated internally, with the outport behaving on behalf of an inport of such an environment component. Thus outport α_4 generates $take_{\alpha_4}$ in response to receiving $take_{\beta_3}$. The protocol automaton $H_{\mathcal{P}}$ associated with port connection \mathcal{P} will for each offer for help consult guards of port connections, and only register this offer for help, if the guard condition is true. Note that this condition might change dynamically; $H_{\mathcal{P}}$ thus constantly monitors guard conditions associated



The diagram represents the sequence of signals that are set as a consequence of setting b_{α_2} . In this example there is a race between the inport β_1 and β_3 . As soon as a *take* is observed by H_P , the automaton forwards this to B . When the *switch* signal is set, the automaton H_P decides by considering the guards who takes over.

Fig. 4. Sequence of communications when b_{α_2} is set

with registered helpers, and removes registration if associated guards become false. The protocol will ensure that helpers maintain their readiness for help, until finally a helper is selected. This selection occurs as follows. As soon as there is at least one registered helper, H_P signals this information to the outport α_2 which raised the alarm, by setting its in-control signal $take_{\alpha_2}$. If this signal arrives prior to the expiration of the rescue period, taking into account the time needed for context switching, the context switch is initiated by the declaration on part of B to now delegate control to *some* helper, through setting $switch_{\alpha_2}$. H_P , the single point of serialization in this protocol, consults upon receiving $switch_{\alpha_2}$ the list of registered helpers, and nondeterministically picks one of these. In the example above, depending on the guard condition g at helper selection time, this can either be the local helper A , in which case it passes control to helper A by setting its in-control-signal $start_{\beta_1}$, or some external helper, in which case H_P

delegates control to the helper in the environment of the sequential composition of A and B by setting $switch_{\alpha_4}$. In the former case, the environment would be informed, that a helper has been found by resetting b_{α_4} , which in turn would cause component C to withdraw its offer. In the latter case, b_{α_2} is withdrawn, and any local helper such as B will withdraw its offer. We note that watchdogs are employed in various places in the protocol to monitor the deadline for context switching, as defined by the rescue period of the alarm raising outport and the time needed for context switching. In case such a watchdog expires, a failure state is reached.

We now complement the definition of hybrid component interfaces and transition composition of components with three steps addressing in particular their semantic foundation. We will first show how to systematically derive basic components from hybrid automata and define the induced semantics of basic components. We then turn to the semantics of transition composition, which is derived inductively from the semantics of its components and automata implementing the distributed helper agreement protocol. Finally, we define the satisfaction relation between component implementations and hybrid interface specifications. The section is wrapped up by completing the construction of the automatic cruise controller.

Definition 9 (Hybrid Automaton for a Basic Component). *Let*

$$H = (\mathbb{M}, Var^{loc}, Var^{in}, Var^{out}, R^D, R^C, R^I, \Phi, \Theta)$$

be a hybrid automaton with disjoint Var^{loc} , Var^{in} , and Var^{out} . Define

- *for each incoming port β in A^{in} , a first-order predicate Φ_β on Var^{loc} , describing admissible initial values for the local variables when $start_\beta$ is received,*
- *for each incoming port β in A^{in} , an initial mode $m_\beta \in \mathbb{M}$.*

We call H an admissible hybrid automaton for a basic component C associated with a plant P if and only if

- $Var_C^{in} = Var^{in}$,
- $Var_C^{out} = Var^{out}$,
- $\Phi \implies \bigvee_{\beta \in A_C^{in}} \Phi_\beta$, and
- $\varphi_C^{assm} \implies \bigwedge_{m \in \mathbb{M}} \Theta_C(m)$

In this case we use H_C to denote the hybrid automaton H of the basic component C .

Now consider an admissible hybrid automaton for a given interface specification. From an implementation perspective, we now wrap the code generated from this hybrid automaton with code supporting suspension and activation, an automaton providing a single point of serialization for all outports, and automata implementing the distributed agreement protocol requirements for inports. Cast as a formal definition of the semantics of basic components, this can be formalized as a parallel composition of a relaxation of the hybrid automata

providing the component implementation, and as timed automata implementing the distributed agreement protocol. This relaxation of the admissible hybrid automaton enforces no constraints on the input and output variables whenever the component is inactive.

Definition 10 (Semantics of a Basic Component). *Let*

$$H = (\mathbb{M}, \text{Var}^{loc}, \text{Var}^{in}, \text{Var}^{out}, R^D, R^C, R^I, \Phi, \Theta)$$

be a hybrid automaton, and C a basic component such that H is admissible for C . The semantics $\llbracket C \rrbracket$ of C is the parallel composition of hybrid automata

$$I_C = H_1 \parallel H_2 \parallel H_3 \parallel \left(\parallel_{\beta \in A_C^{in}} H_\beta \right),$$

which are defined as follows.

- H_1 is a hybrid automaton that is basically H but augmented with a mode m_{inact} for inactivity and a mode m_{failed} for failures:

$$H_1 = (\mathbb{M} \cup \{m_{inact}, m_{failed}\}, \text{Var}^{loc}, \text{Var}_{H_1}^{in}, \text{Var}^{out} \cup \{fail_C\}, \\ R_{H_1}^D, R_{H_1}^C, true, \Phi \wedge M = m_{inact}, \Theta_{H_1}),$$

where

- $m_{inact}, m_{failed} \notin \mathbb{M}$,
- $\text{Var}_{H_1}^{in} = \text{Var}^{in} \cup \{active_C\} \cup \{start_\beta \mid \beta \in A_C^{in}\}$,
-

$$R_{U, H_1}^D = R_U^D \cup \{(m, active_C = 0, \emptyset, m_{inact}) \mid m \in \mathbb{M}\} \quad (1)$$

$$\cup \{(m_{inact}, active_C = 1 \wedge start_\beta \wedge \varphi_\beta^{entry}, \quad (2)$$

$$\Phi'_\beta \wedge \bigwedge_{v \in \text{Var}^{out}} (v' = v), m_\beta) \mid \beta \in A_C^{in}\},$$

$$\cup \{(m_{inact}, active_C = 1 \wedge start_\beta \wedge \neg \varphi_\beta^{entry}, \quad (3)$$

$$fail_C := 1, m_{failed}) \mid \beta \in A_C^{in}\},$$

where Φ'_β is Φ_β with every variable primed,

- $R_{L, H_1}^D = R_L^D$
- $R_{H_1}^C(m) = R^C(m)$, if $m \in \mathbb{M}$, and $R_{H_1}^C(m_{inact}) = true$,
- $\Theta_{H_1}(m) = \Theta(m)$, if $m \in \mathbb{M}$, and $\Theta_{H_1}(m_{inact}) = true$,
- Automaton H_2 ensures that the variables $active_C$ and $fail_C$ are only changed by discrete transitions (Fig. 5). It also reacts to $suspend_C$ signals by deactivating the component.
- The hybrid automaton H_3 handles all outgoing ports. It is defined as follows:

$$H_3 = (\mathbb{M}_3, V_3^{loc}, V_3^{in}, V_3^{out}, R_3^D, R_3^C, R_3^I, \Phi_3, \Theta_3)$$

with

$$\mathbb{M}_3 = \left(2^{A_C^{out}} \times 2^{A_C^{out}}\right) \cup \{Fail\}, \quad (4)$$

$$V_3^{loc} = \{t_\alpha \mid \alpha \in A_C^{out}\},$$

$$V_3^{in} = \{active_C\} \cup Var_C^{in} \cup Var_C^{out} \cup \{take_\alpha \mid \alpha \in A_C^{out}\},$$

$$V_3^{out} = \{fail_C\} \cup \{b_\alpha, switch_\alpha \mid \alpha \in A_C^{out}\},$$

$$R_{U,3}^D = \{((X, Y), \varphi_\alpha^{alarm} \wedge active_C, b_\alpha := 1, t_\alpha := 0, \\ (X \cup \{\alpha\}, Y)) \mid \alpha \notin X \cup Y\} \quad (5)$$

//Alarm α is set

$$\cup \{((X, Y), t_\alpha \geq \alpha, fail_C := 1, Fail) \mid \alpha \in X\} \quad (6)$$

// Failure: alarm α has exceeded the maximum duration

$$\cup \{((X, Y), take_\alpha \wedge t_\alpha < \Delta_\alpha - \tau, b_\alpha := 0, t_\alpha := 0, \\ (X \setminus \{\alpha\}, Y \cup \{\alpha\})) \mid \alpha \in X \setminus Y\} \quad (7)$$

//Alarm α is reset because of a take

$$\cup \{((X, Y), t_\alpha = 0, \\ switch_\alpha := 1, \forall v \in V_3^{out} \cup V_3^{loc} \setminus \{switch_\alpha\} : v := 0, \\ (\emptyset, \emptyset)) \mid \alpha \in Y\} \quad (8)$$

//A $switch_\alpha$ is set because of a previous take $_\alpha$

$$R_{L,3}^D = \{((X, Y), \mu_\alpha \leq t_\alpha < \Delta_\alpha \wedge \neg \varphi_\alpha^{alarm}, b_\alpha := 0, t_\alpha := 0, \\ (X \setminus \{\alpha\}, Y)) \mid \alpha \in X\} \quad (9)$$

//Alarm α can be reset

$$R_3^C(v) = 0 \text{ for all } v \in V_3^{out}$$

$$R_3^C(v) = 1 \text{ for all } v \in V_3^{loc}$$

$$R_3^I = true$$

$$\Phi_3 = (\mathbb{M} = (\emptyset, \emptyset) \wedge \bigwedge_{v \in V_3^{out} \cup V_3^{loc}} v = 0)$$

$$\Theta_3 = true \quad (10)$$

– The automaton H_β for an incoming port $\beta = (c_\beta, \lambda_\beta, take_\beta, start_\beta) \in A_C^{in}$ is given in Fig. 6.

On H_1 of the semantics of a basic component: The purpose of H_1 is to add the intended control strategy of H into the semantics. However, we have to augment this by additional locations and some transitions in order to satisfy $SPEC_C$. The automaton H_1 behaves like H except for the case that $active_C$ becomes false. In this case H_1 switches by (1) from an arbitrary location to the additional location m_{inact} which stands for inactivity. As soon as H_1 discovers a rising edge of $active_C$ together with $start_\beta$ the automaton fires transition (2) because

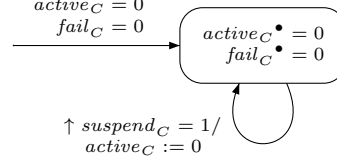


Fig. 5. Automaton H_2

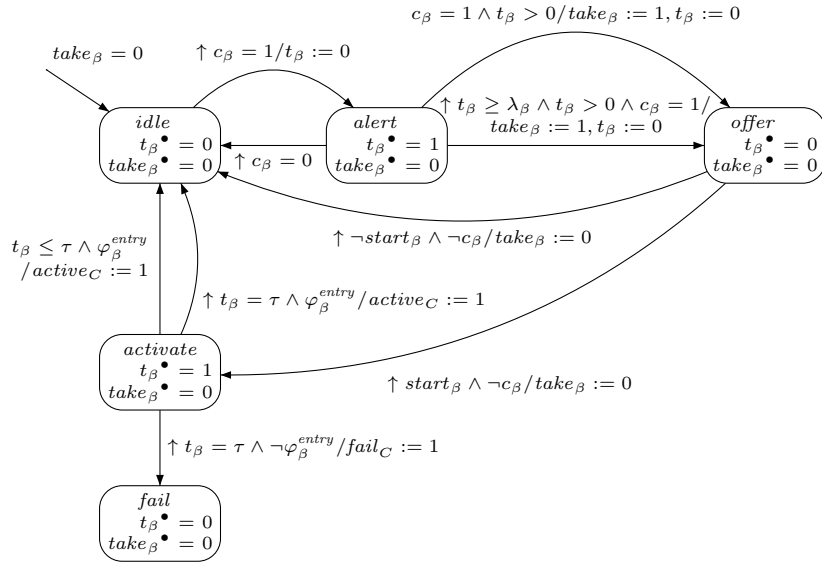


Fig. 6. Automaton H_β

it knows that is activated again via the incoming port β . The assignment that belongs to this transition keeps all variables of Var_C^{out} stable but executes the assignments Φ_β given by the definition of β when switching to m_β . Note that both Φ_β and m_β were given in Def. 9. However, if the activation takes place while the entry condition φ_β^{entry} is *not* given, then H_1 switches to a location m_{failed} and sets $fail_C$.

On H_3 of the semantics of a basic component: The purpose of H_3 is to manage the outgoing alarm signals b_α . To this end H_3 has locations defined in (4) of

the form (X, Y) with $X, Y \in 2^{A_C^{out}}$ with only one exception needed for failures. The idea of a location (X, Y) is that for all $\alpha \in X$ the corresponding signal b_α is currently set. If $\alpha \in Y$ then this means that the component has observed a rising edge of $take_\alpha$. The transitions are defined to keep these invariants. In (5) the signal b_α is set as soon as φ_α^{alarm} is satisfied while C is active. With this transition a timer t_α is reset to observe whether a $take_\alpha$ is set in time or φ_α^{alarm} becomes false in time. If this is not the case, then the transitions in (6) will switch to the exceptional failure state and will set $fail_C$. The transitions (7) are fired provided that a $take_\alpha$ signal arrives in time, i.e. it arrives τ time units before the component is signaling b_α for Δ_α time units. This transition moves α from the X set into the Y set. For an α being in Y the automaton has to react within 0 time units by setting $switch_\alpha$ as given in (8). If b_α is set for at least μ_α time units the automaton is able to reset this signal provided that φ_α^{alarm} is not satisfied anymore (9). Note that these transitions are not urgent, hence the automaton is not forced to do this if the transition is enabled.

On H_β of the semantics of a basic component: The purpose of H_β is to handle incoming requests at inport β . Initially it waits in *idle* until a rising edge of c_β is observed. This enforces a transition to *alert* where the automaton might remain for a while. Due to the timer t_β and the urgent transition with guard $t_\beta \geq \lambda_\beta$ the automaton sets $take_\beta$ after at most λ_β time units and switches to state *offer*. A prerequisite is that the c_β is still set, otherwise H_β moves back to *idle*. In *offer* it waits for a falling edge of c_β . If this happens with $start_\beta$ being set, the automaton switches to *activate* and activates the component after at most τ seconds. Otherwise, i.e. $start_\beta$ is not set, the automaton proceeds to *idle* without activating the component. In case that the activation must take place (τ seconds elapsed) and the entry condition φ_C^{entry} is not true, the automaton sets *fail*.

We can now define the semantics of the transition composition of components C_1, \dots, C_n , where in taking a white box view we assume as given the semantics $\llbracket C_j \rrbracket$ of the components. The semantics is defined in terms of the parallel composition of the hybrid automata representing the semantics of its components, and three timed automata which define activation and failure of the composed systems in terms of the status of its components, interpret the connection of inports of the composed system to local inports in propagating control signals outside-in, and interpret all port connections for all local outports to implement the distributed helper identification protocol. Note that control signals ensure that there is always at most one active component inside a transition composition of components.

Definition 11 (Semantics of a Transition Composition). *Let C be a component obtained by a transition composition of the components C_1, \dots, C_n with port connection $(\mathcal{P}, \mathcal{Q})$. The semantics $\llbracket C \rrbracket$ of C is the parallel composition of hybrid automata*

$$I_C = \left(\prod_i \llbracket C_i \rrbracket \right) \parallel H_T \parallel H_{\mathcal{P}} \parallel H_{\mathcal{Q}}$$

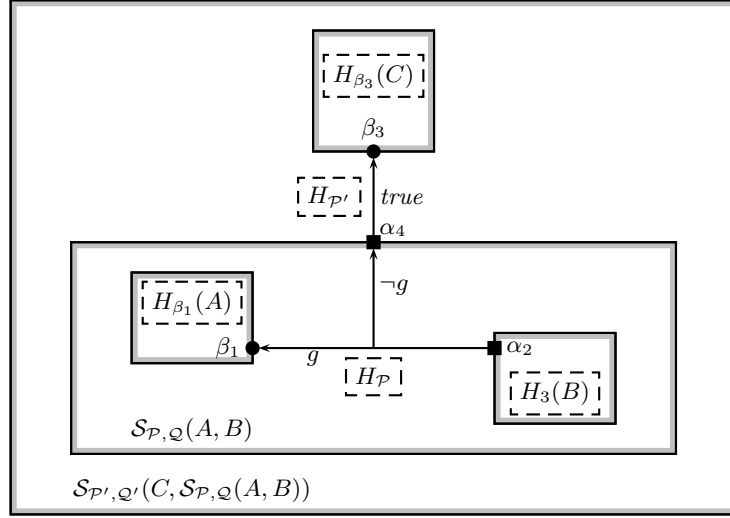


Fig. 7. Semantics: Automata involved for α_2

with the following components.

- H_T is a hybrid automaton that handles the reactions to $suspend_C$, $start_\beta$, and $Fail_{C_i}$ appropriately:

$$\begin{aligned}
\mathbb{M}_T &= \{inactive, active, failed\} \\
V_T^{loc} &= \emptyset, \\
V_T^{in} &= \{start_\beta \mid \beta \in A_C^{in}\} \cup \{active_{C_i}, fail_{C_i} \mid i \in \{1, \dots, n\}\} \\
&\quad \cup \{suspend_C\} \\
V_T^{out} &= \{suspend_{C_i} \mid i \in \{1, \dots, n\}\} \cup \{active_C, fail_C\}
\end{aligned}$$

$$\begin{aligned}
R_{U,T}^D = & \{(m, fail_{C_i}, fail_C := 1, failed) \mid i \in \{1, \dots, n\}, m \in \mathbb{M}_T\} \\
& // A component failed, hence the composition fails. \\
& \cup \{(m, suspend_C, \\
& \quad active_C := 0, suspend_{C_1} := 1, \dots, suspend_{C_n} := 1, \\
& \quad inactive) \mid m \in \mathbb{M}_T \setminus \{failed\}\} \\
& // Suspend is forwarded to all components. \\
& \cup \{(inactive, \neg suspend_C, \\
& \quad suspend_{C_1} := 0, \dots, suspend_{C_n} := 0, inactive)\} \\
& // Reset of suspend is forwarded to all components. \\
& \cup \{(inactive, start_\beta \wedge \varphi_\beta^{entry}, active_C := 1, active)\} \\
& // A start_\beta activates the composition C. \\
& \cup \{(inactive, start_\beta \wedge \neg \varphi_\beta^{entry}, fail_C := 1, failed)\} \\
& // A start_\beta without meeting the entry condition. \\
& \cup \{(active, active_{C_i}, \\
& \quad suspend_{C_1} := 0, \dots, suspend_{C_{i-1}} := 0, suspend_{C_{i+1}} := 0, \\
& \quad \dots, suspend_{C_n} := 0, active)\} \\
& // When a C_i becomes active, all others are suspended.
\end{aligned}$$

$$\begin{aligned}
R_{L,T}^D &= \emptyset \\
R_T^C(v) &= 0 \text{ for all } v \in V_T^{out} \cup V_T^{loc} \\
R_T^I &= true \\
\Phi_T &= (\mathbb{M} = inactive \wedge \bigwedge_{v \in V_T^{out} \cup V_T^{loc}} v = 0) \\
\Theta_T &= true
\end{aligned}$$

– H_Q is a hybrid automaton that implements the invariant

$$\begin{aligned}
& \forall \beta \in A_C^{in} : (c_\beta \longleftrightarrow c_{Q(\beta)}) \\
& \quad \wedge (take_{Q(\beta)} \longleftrightarrow take_\beta) \\
& \quad \wedge (start_\beta \longleftrightarrow start_{Q(\beta)})
\end{aligned}$$

– Let $A^{src} = \bigcup_i A_{C_i}^{out}$ and $A^{dst} = \bigcup_i A_{C_i}^{in}$. We set

$$H_{\mathcal{P}} = (\mathbb{M}_{\mathcal{P}}, V_{\mathcal{P}}^{loc}, V_{\mathcal{P}}^{in}, V_{\mathcal{P}}^{out}, R_{\mathcal{P}}^D, R_{\mathcal{P}}^C, R_{\mathcal{P}}^I, \Phi_{\mathcal{P}}, \Theta_{\mathcal{P}})$$

with

$$\begin{aligned}
\mathbb{M}_{\mathcal{P}} &= 2^{A^{src}} \times 2^{A^{src} \times (A^{dst} \cup A_C^{out})}, \\
V_{\mathcal{P}}^{loc} &= \emptyset,
\end{aligned}$$

$$\begin{aligned}
V_{\mathcal{P}}^{in} &= \text{Var}_C^{in} \cup \text{Var}_C^{out} \cup \{b_\alpha, \text{switch}_\alpha \mid \alpha \in A^{src}\} \\
&\quad \cup \{\text{take}_\beta \mid \beta \in A^{dst}\} \cup \{\text{take}_\alpha \mid \alpha \in A_C^{out}\}, \\
V_{\mathcal{P}}^{out} &= \{c_\beta, \text{start}_\beta \mid \beta \in A^{dst}\} \cup \{b_\alpha, \text{switch}_\alpha \mid \alpha \in A_C^{out}\} \\
&\quad \cup \{\text{take}_\alpha \mid \alpha \in A^{src}\}, \\
R_{U,\mathcal{P}}^D &= \bigcup_{p \in \mathcal{P}} R_{U,\mathcal{P}}^D(p) \quad \text{where } R_{U,\mathcal{P}}^D(p) \text{ is defined in Fig. 8} \\
R_{L,\mathcal{P}}^D &= \emptyset \\
R_{\mathcal{P}}^C(v) &= 0 \text{ for all } v \in V_{\mathcal{P}}^{out} \cup V_{\mathcal{P}}^{loc} \\
R_{\mathcal{P}}^I &= \text{true} \\
\Phi_{\mathcal{P}} &= (\mathbb{M} = (\emptyset, \emptyset) \wedge \bigwedge_{v \in V_{\mathcal{P}}^{out} \cup V_{\mathcal{P}}^{loc}} v = 0) \\
\Theta_{\mathcal{P}} &= \text{true}
\end{aligned}$$

In summary, the semantics of a transition composition is the parallel composition of semantics of its components together with three additional hybrid automata H_T , $H_{\mathcal{P}}$ and H_Q . The automaton H_T implements the following properties:

- Whenever a $fail_{C_i}$ of a component occurs this leads to a $fail_C$ signal. There is no way to reset $fail_C$.
- Whenever a $start_\beta$ signal is given then the composition is activated provided that the entry condition φ_β^{entry} is met. Otherwise H_C produces a $fail_C$ signal.
- It observes $active_{C_i}$ signals of the components and provide $suspend_{C_j}$ signals to all $j \neq i$. This is needed in case of an internal take over situation.

In the case of H_Q this automaton just implements invariants over the variables. As the details of this automaton is straightforward they are not given here. The invariants implemented are also fairly obvious. Not trivial is the construction of $H_{\mathcal{P}}$. Its purpose is to implement the correct handling of the outgoing signals of the components. A location of $H_{\mathcal{P}}$ is a pair (X, Y) , where

- X contains all $\alpha \in A^{src}$ which are currently set. The set A^{src} is the union of all output of the components C_1, \dots, C_n .
- Y is set containing pairs (α, β_i) or (α, α_j) where $\alpha \in A^{src}$, β_i is an import of a component and α_j is an output of the composition C . A pair being in Y stands for the information that α is set and a $take$ from β_i resp. α_j has been set.

The transitions of $H_{\mathcal{P}}$ are given in Fig. 8 and they are implementing the invariants for the locations given above. Whenever b_α becomes true, then α is added to the X set (11). Moreover, this transition forwards this signal to all connected import and output. As soon as a $take$ is received from there the corresponding pair is added to the Y set (12–13). To do so, it is required that the given guard g_i resp. g_j are satisfied. The transitions (14–17) handle the various cases that

For a tuple

$$p = (\alpha, \{(\beta_1, g_1, \mathcal{A}_1), \dots, (\beta_k, g_k, \mathcal{A}_k)\}, \{(\alpha_1, g'_1), \dots, (\alpha_l, g'_l)\}) \in \mathcal{P}$$

we set $R_{U, \mathcal{P}}^D(p)$ to

$$\begin{aligned} & \{(X, Y), b_\alpha, c_{\beta_1} := 1, \dots, c_{\beta_k} := 1, b_{\alpha_1} := 1, \dots, b_{\alpha_l} := 1, \\ & (X \cup \{\alpha\}, Y) \mid \alpha \notin X \} \end{aligned} \quad (11)$$

//Alarm α is set, all connected β_i, α_j are set

$$\cup \{(X, Y), take_{\beta_i} \wedge g_i, take_\alpha := 1, (X, Y \cup \{(\alpha, \beta_i)\}) \mid \alpha \in X, (\alpha, \beta_i) \notin Y\} \quad (12)$$

//There is a *take* from a connected β_i which is forwarded

$$\cup \{(X, Y), take_{\alpha_j} \wedge g'_j, take_\alpha := 1, (X, Y \cup \{(\alpha, \alpha_j)\}) \mid \alpha \in X, (\alpha, \alpha_j) \notin Y\} \quad (13)$$

//There is a *take* from a connected α_j which is forwarded

$$\cup \{(X, Y), \neg(take_{\beta_i} \wedge g_i), take_\alpha := 1, \\ (X, Y \setminus \{(\alpha, \beta_i)\}) \mid \alpha \in X, (\alpha, \beta_i) \in Y, \exists \gamma : (\alpha, \gamma) \in Y \setminus \{(\alpha, \beta_i)\}\} \quad (14)$$

// β_i is not able to take over anymore, but an alternative is left

$$\cup \{(X, Y), \neg(take_{\beta_i} \wedge g_i), take_\alpha := 0, \\ (X, Y \setminus \{(\alpha, \beta_i)\}) \mid \alpha \in X, (\alpha, \beta_i) \in Y, \neg(\exists \gamma : (\alpha, \gamma) \in Y \setminus \{(\alpha, \beta_i)\})\} \quad (15)$$

// β_i is not able to take over anymore and no alternative is left

$$\cup \{(X, Y), \neg(take_{\alpha_i} \wedge g'_i), take_\alpha := 1, \\ (X, Y \setminus \{(\alpha, \alpha_j)\}) \mid \alpha \in X, (\alpha, \alpha_j) \in Y, \exists \gamma : (\alpha, \gamma) \in Y \setminus \{(\alpha, \alpha_j)\}\} \quad (16)$$

// α_j is not able to take over anymore, but an alternative is left

$$\cup \{(X, Y), \neg(take_{\alpha_i} \wedge g'_i), take_\alpha := 0, \\ (X, Y \setminus \{(\alpha, \alpha_j)\}) \mid \alpha \in X, (\alpha, \alpha_j) \in Y, \neg(\exists \gamma : (\alpha, \gamma) \in Y \setminus \{(\alpha, \alpha_j)\})\} \quad (17)$$

// α_j is not able to take over anymore and no alternative is left

$$\cup \{(X, Y), switch_\alpha, start_{\beta_i} := 1, \mathcal{A}_i, RESET(Var_{\mathcal{P}}^{out} \setminus \{\beta_i\}), \\ (\emptyset, \emptyset) \mid \alpha \in X, (\alpha, \beta_i) \in Y)\} \quad (18)$$

//There is a *switch* which is forwarded to a connected β

$$\cup \{(X, Y), switch_\alpha, switch_{\alpha_j} := 1, RESET(Var_{\mathcal{P}}^{out} \setminus \{\alpha_j\}), \\ (\emptyset, \emptyset) \mid \alpha \in X, (\alpha, \alpha_j) \in Y)\} \quad (19)$$

//There is a *switch* which is forwarded to a connected α

where $RESET(\{v_1, \dots, v_M\})$ stands for $v_1 := 0, \dots, v_M := 0$.

Fig. 8. Urgent transitions for a $p \in \mathcal{P}$

either a guard becomes not satisfied or the corresponding *take* signal was reset. They have to distinguish the case whether the *take*_α signal has to be reset or not. If a (α, β_i) or (α, α_j) must be removed from the Y set, then the question is whether an alternative is left. If it is, the *take*_α can remain set, otherwise it must be reset.

As soon as a *switch*_α occurs the H_P automaton has to react without delay due to the transitions (18–19). They select a *switch*_{α_j} or *start*_{β_i} signal where (α, α_j) resp. (α, β_i) must be in the current Y set. This ensures that the corresponding guard is currently satisfied. In case of *start*_{β_i} the assignments are also executed. Note that the selection is done nondeterministically if more than one option is available. However, the reaction to a *switch*_α happens without delay because the transitions are defined as being urgent.

Definition 12 (Semantics of a Closed Loop). *The semantics of a closed loop consisting of a component C and a plant P is defined as $\llbracket C \parallel P \rrbracket := \llbracket C \rrbracket \parallel P$.*

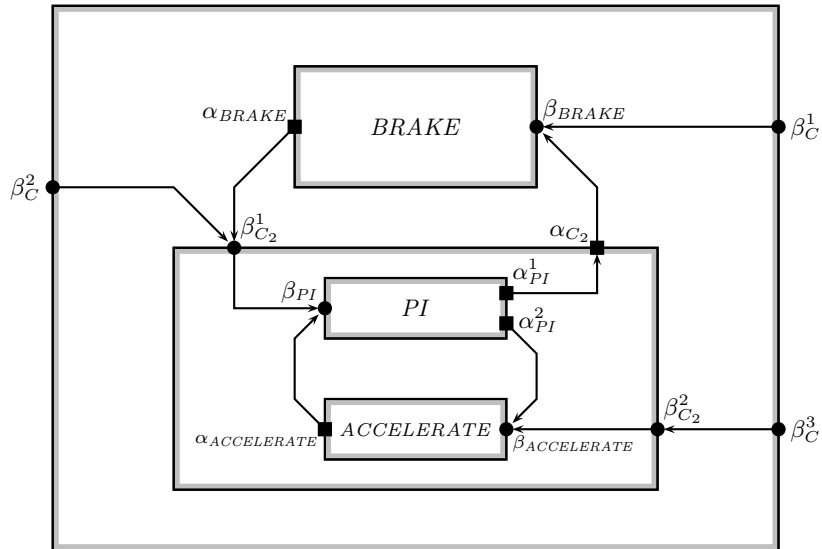


Fig. 9. Interconnection Structure for the ACC Example

We now complete the design of the ACC controller.

Example (cont.) We complete the design by adding a *BRAKE* component to cater for the so far unserved alarm of the PI controller for plant situations where the actual velocity is much higher than the desired velocity, and arrive at the hierarchical composition depicted in Figure 9.

Comp.	Var^{in}	Var^{out}	Var^{loc}	φ^{assm}	φ^{prom}	Δ^{stable}
C	$\{v\}$	$\{a\}$	$\{x\}$	$-30 \leq v \leq 30$	$-2 \leq v^\bullet \leq 1.5$	300
$BRAKE$	$\{v\}$	$\{a\}$	\emptyset	$5 \leq v \leq 30$	$v^\bullet = -2$	300
C_2	$\{v\}$	$\{a\}$	$\{x\}$	$-30 \leq v \leq 15$	$-1.4 \leq v^\bullet \leq 1.5$	300
PI	$\{v\}$	$\{a\}$	$\{x\}$	$-15 \leq v \leq 15$	$-1.4 \leq v^\bullet \leq 1.4$	300
$ACCELERATE$	$\{v\}$	$\{a\}$	\emptyset	$-30 \leq v \leq -5$	$v^\bullet = 1.5$	300

Table 1. Component Interfaces (and local variables) for ACC Example

Output	Comp.	$\varphi_\alpha^{alarmOn}$	μ_α	Δ_α	φ_α^{exit}
α_{BRAKE}	$BRAKE$	$v \leq 6$	0.005	0.01	$v \leq 6$
α_{C_2}	C_2	$v \geq 14$	0.006	0.01	$13.9 \leq v \leq 14.1$
α_{PI}^1	PI	$v \geq 14$	0.006	0.01	$13.9 \leq v \leq 14.1$
α_{PI}^2	PI	$v \leq -14$	0.006	0.01	$-14.1 \leq v \leq -13.9$
$\alpha_{ACCELERATE}$	$ACCELERATE$	$v \geq -6$	0.005	0.01	$v \geq -6$

Table 2. Outputs for ACC Example

All specifications of components are summarized in Tables 1, 2, and 3. We provide admissible hybrid automata for the basic components $BRAKE$, PI and $ACCELERATE$ as follows.

$$H_{BRAKE} = (\{m_{BRAKE}\}, \emptyset, \{v\}, \{a\}, \emptyset, true, true, true, \Theta_{BRAKE})$$

with

$$- \Theta_{BRAKE}(m_{BRAKE}) = (a = -2).$$

$$H_{PI} = (\{m_{C_{PI}}\}, \{x\}, \{v\}, \{a\}, \emptyset, R_{PI}^C, true, \Phi_{PI}, \Theta_{PI})$$

with

$$- \Theta_{PI}(m_{PI}) = (a = -0.001x - 0.052v).$$

$$- R_{PI}^C(m_{C_{PI}}) = (x^\bullet = v)$$

$$- \Phi_{PI} = (x = 0)$$

Inport	Comp.	λ_β	φ_β^{entry}	Φ_β
β_C^1	C	0.004	$10 \leq v \leq 30$	n/a
β_C^2	C	0.004	$-13.5 \leq v \leq 13.5$	n/a
β_C^3	C	0.004	$-30 \leq v \leq -10$	n/a
β_{BRAKE}	$BRAKE$	0.003	$10 \leq v \leq 30$	n/a
$\beta_{C_2}^1$	C_2	0.004	$-13.5 \leq v \leq 13.5$	n/a
$\beta_{C_2}^2$	C_2	0.004	$-30 \leq v \leq -10$	n/a
β_{PI}	PI	0.0025	$-13.5 \leq v \leq 13.5$	$x = 0$
$\beta_{ACCELERATE}$	$ACCELERATE$	0.0025	$-30 \leq v \leq -10$	n/a

Table 3. Inports (and initialization of local variables) for ACC Example

$$H_{ACCELERATE} = (\{m_{ACCELERATE}\}, \emptyset, \{v\}, \{a\}, \emptyset, true, true, true, \Theta_{ACCELERATE})$$

with

$$- \Theta_{ACCELERATE}(m_{ACCELERATE}) = (a = 1.5).$$

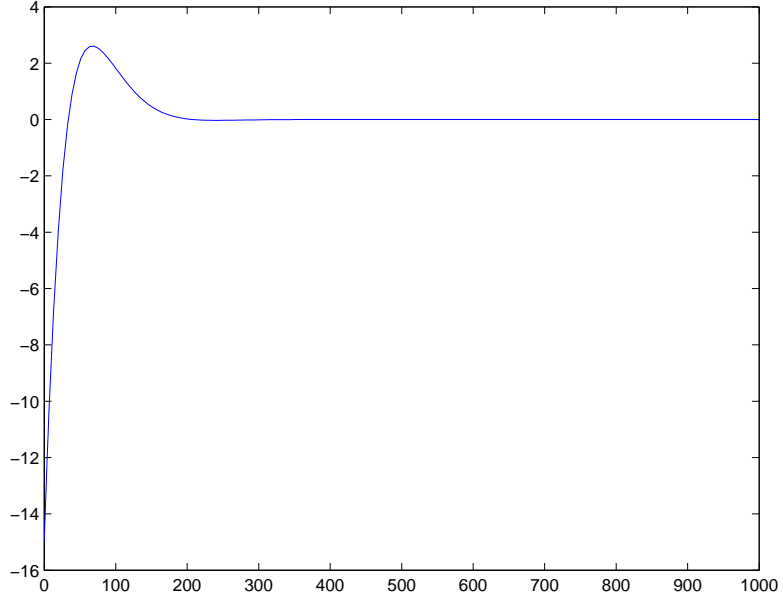


Fig. 10. Example trajectory of $H_{PI} \parallel P$: time (horizontal axis) vs. velocity differential v (vertical axis)

Figure 10 gives a sample trajectory with the closed loop of the PI controller and the plant model.

We conclude this section by formally defining the satisfaction relation for hybrid interface specifications.

Definition 13 (Satisfaction of Component Interface Specification). *An implementation I (in terms of a hybrid automaton) satisfies the interface specification $SPEC_C$ of a component C associated with a plant P (denoted by $I \parallel P \models SPEC_C$) iff*

$$(A) \text{Var}_I^{out} \supseteq \text{Var}_C^{out} \cup C_C^{out},$$

- (B) $\text{Var}_I^{\text{in}} \supseteq \text{Var}_C^{\text{in}} \cup C_C^{\text{in}}$,
(C) $\models \varphi_C^{\text{entry}} \implies \neg \varphi_\alpha^{\text{alarm}}$ for all $\alpha \in A_C^{\text{out}}$,
(D) $\models \varphi_C^{\text{entry}} \implies \varphi_P^{\text{safe}} \wedge \varphi_C^{\text{assm}}$

and for all runs that are failure-free (i.e., $\Box \neg \text{fail}_C$) satisfy the following requirements:

- (E) $\neg \text{active}_C$ UNLESS $(\bigvee_{\beta \in A_C^{\text{in}}} (\text{start}_\beta \wedge \varphi_\beta^{\text{entry}}))$ (a component is inactive initially unless it is started via an inport β .)
(F) $\forall \alpha \in A_C^{\text{out}} : \neg \Diamond (\neg \varphi_\alpha^{\text{alarm}} \cup (\varphi_\alpha^{\text{alarm}} \wedge \text{active}_C \wedge \neg b_\alpha))$ (when $\varphi_\alpha^{\text{alarm}}$ becomes true while the component is active, b_α holds)
(G) $\forall \alpha \in A_C^{\text{out}} : \neg \Diamond (\neg b_\alpha \cup (b_\alpha \cup_{<\mu_\alpha} \neg b_\alpha \wedge \neg \text{take}_\alpha))$ (when b_α becomes false without receiving a take, b_α was on for at least μ_α time units)
(H) $\forall \beta \in A_C^{\text{in}} : \neg \Diamond (\neg \text{start}_\beta \cup \text{start}_\beta \wedge \neg \text{active}_C) \wedge \neg \Diamond (\neg \text{active}_C \cup \text{active}_C \wedge \neg \bigvee_{\beta \in A_C^{\text{in}}} \text{start}_\beta)$ (a component is activated exactly when a start_β has a rising edge)
(I) $\neg \Diamond (\neg \text{suspend}_C \cup \text{suspend}_C \wedge \text{active}_C) \wedge \neg \Diamond (\text{active}_C \cup \neg \text{active}_C \wedge \neg \text{suspend}_C)$ (a component is deactivated exactly when suspend_C has a rising edge)
(J) $\forall \beta \in A_C^{\text{in}} : \neg \Diamond (c_\beta \wedge \neg \text{take}_\beta \cup_{>\lambda_\beta} \text{true})$ (an incoming alarm is answered by take_β after at most λ_β time units)
(K) $\forall \beta \in A_C^{\text{in}} : \Box (\text{take}_\beta \wedge \neg c_\beta \cup_{=0} \neg \text{take}_\beta)$ (take_β is only set when an alarm is incoming)
(L) $\forall \beta \in A_C^{\text{in}} : \neg \Diamond (\text{take}_\beta \wedge c_\beta \cup (\neg \text{take}_\beta \wedge c_\beta))$ (take_β is not withdrawn if the incoming alarm remains)
(M) $\neg \Diamond (\neg \text{active}_C \cup (\text{active}_C \wedge \neg \varphi_C^{\text{entry}}))$ (φ^{entry} holds whenever a component is activated)
(N) $\Box (\text{active}_C \implies \varphi_P^{\text{safe}} \wedge \varphi_C^{\text{assm}} \wedge \varphi_C^{\text{prom}})$ (active components guarantee $\varphi_P^{\text{safe}} \wedge \varphi_C^{\text{assm}} \wedge \varphi_C^{\text{prom}}$)
(O) $\Box (\text{active}_C \implies ((\Diamond_{\Delta_C^{\text{stable}}} (\Box \varphi_P^{\text{stable}})))$ UNLESS $(\neg \text{active}_C)$ (active components guarantee convergence to $\varphi_P^{\text{stable}}$ within Δ_C^{stable} time units)
(P) $\forall \alpha, \alpha' \in A_C^{\text{out}} : \neg \Diamond (\neg \text{switch}_\alpha \cup (\text{switch}_\alpha \wedge b_{\alpha'}))$ (when switch_α has a rising edge, then all outgoing alarms are reset)
(Q) $\forall \beta \in A_C^{\text{in}}, \alpha \in A_C^{\text{in}} : \neg \Diamond (\neg \text{switch}_\alpha \cup (\text{switch}_\alpha \wedge \text{take}_\beta))$ (when any switch_α has a rising edge, then all take_β are reset)
(R) $\forall \alpha \in A_C^{\text{out}} : \neg \Diamond (\neg \text{switch}_\alpha \cup (\text{switch}_\alpha \wedge \neg \varphi_\alpha^{\text{exit}}))$ (when switch_α becomes true, $\varphi_\alpha^{\text{exit}}$ is guaranteed)

and for all trajectories that initially fulfill φ^{entry} and fail for the first time at time t (i.e., $\neg \text{fail}_C \cup_{=t} \text{fail}_C$):

- (S) $(\exists \alpha \in A_C^{\text{out}}, t_\alpha \leq t - \Delta_\alpha : \forall t' \in [t_\alpha, t] : b_\alpha(t')) \vee (\exists \beta \in A_C^{\text{in}} : \text{start}_\beta(t) \wedge \neg \varphi_\beta^{\text{entry}}(t))$ (a failure is preceded by an alarm that becomes and stays active at least Δ_α time units earlier or the failure is due to an entry condition for an inport β that is not met during the activation.)
(T) $\Box (\text{fail}_C \implies (\Box \text{fail}_C))$ (once a failure occurs, fail_C remains true)

4 Hierarchical Verification of Robust Safety and Stability

In the following we give verification conditions for basic components and transition compositions thereof. These verification conditions imply that the component fulfills its interface specification, so that both safety and stability properties are guaranteed while the component is active. The verification conditions are of three types:

- inequalities on scalars: these include the different timing constraints involving the λ_β , μ_α , or Δ_α ,
- implications on first order predicates: these relate the predicates that are part of the interface specification (e.g., φ_C^{entry} , $\varphi_\alpha^{\text{alarmOn}}$, $\varphi_\beta^{\text{entry}}$) to one another, and
- Lyapunov function conditions: these are used to prove stability of the system, using parameterized Lyapunov functions.

Lyapunov functions are abstract energy functions of the closed loop. Intuitively, a Lyapunov function maps each system state onto a nonnegative energy value, with the restriction that the function must always decrease along every trajectory, unless a designated equilibrium point is reached. Since the Lyapunov function also has its minimum at this equilibrium, it serves as a tool to prove convergence. In the following, we define Lyapunov functions for single-mode systems. Lyapunov functions will be used in the verification conditions for stability. In particular, there will be constraints on the existence of Lyapunov functions with a particular parameterization. Specialized software (SDP solvers [Bor99,RPS99]) can be used to carry out the automatic computation of these functions.

Definition 14 (Lyapunov Functions). *Let*

$$H = (\{m\}, \text{Var}^{\text{loc}}, \text{Var}^{\text{in}}, \text{Var}^{\text{out}}, \emptyset, R^C, R^I, \Phi, \Theta)$$

be a hybrid automaton and $\text{Var} = \text{Var}^{\text{loc}} \cup \text{Var}^{\text{in}} \cup \text{Var}^{\text{out}}$.

Let $X = [(X^O)^T, (X^L)^T, (X^I)^T]^T$ be a vector of valuations of Var , with subvectors X^O , X^L and X^I pertaining to the variables in Var^{out} , Var^{loc} , and Var^{in} , respectively. Furthermore, define the vector of controlled variables as $X^C := [(X^O)^T, (X^L)^T]^T$. Assume that $R^C(m)$ is given by a differential inclusion

$$X^{C^\bullet} \in F(X), F : \text{Var} \rightarrow 2^{\mathbb{R}^{|\text{Var}^{\text{loc}}| + |\text{Var}^{\text{out}}|}}$$

and R^I by a differential inclusion

$$X^{I^\bullet} \in G(X^I), G : \text{Var}^{\text{in}} \rightarrow 2^{\mathbb{R}^{|\text{Var}^{\text{in}}|}}$$

A Lyapunov function wrt.

- *an equilibrium state $X_e^O \in \mathbb{R}^{|\text{Var}^{\text{out}}|}$, $X_e^O \models \varphi_P^{\text{stable}}$ and*
- *a nonnegative function $f : \text{Var} \rightarrow \mathbb{R}$ such that $f(X) = 0$ if $X^O = X_e^O$*

is a function $\mathcal{V} : \mathbb{R}^{|Var|} \rightarrow \mathbb{R}$ for which there exist $k_1, k_2, k_3 > 0$ such that for all $X \models \Theta(m)$:

- (1) $k_1 \|X^O - X_e^O\|^2 \leq \mathcal{V}(X) \leq k_2 f(X)$
- (2) $\mathcal{V}^\bullet(X) := \sup_{\bar{X}^I \in G(X^I), \bar{X}^C \in F(X)} \left(\frac{d\mathcal{V}}{dX}(X) \cdot \begin{bmatrix} \bar{X}^C \\ \bar{X}^I \end{bmatrix} \right) \leq -k_3 f(X)$

The values of k_2 and k_3 will later be used to estimate convergence time toward a φ_P^{stable} . Note that the set of Lyapunov functions for a given system is closed under conic combination, i.e., if the \mathcal{V}_i are Lyapunov functions for a system, then, for all $\lambda_i \geq 0$ such that at least one $\lambda_i > 0$, $\sum_i \lambda_i \mathcal{V}_i$ is also a Lyapunov function. Moreover, the constants k_1 , k_2 and k_3 also add up in the same manner, as the Lyapunov function $\sum_i \lambda_i \mathcal{V}_i$ will have constants k_1 , k_2 , and k_3 , that are the weighted sums over the λ_i of the corresponding constants of the \mathcal{V}_i . This property will be exploited heavily in the verification conditions. The function f can, in general, be chosen arbitrarily. However, some choices will result in better convergence time estimates than others. Ideally, the function $f(X)$ should be of similar form as $\mathcal{V}(X)$ and $\mathcal{V}^\bullet(X)$, since this provides the tightest overapproximations.

Note that, while the equilibrium state X_e^O is only defined in terms of output variables of a system, the function \mathcal{V} can potentially talk about all variables in Var , but is not required to do so. This is sufficient, because we are only interested in convergence of variables that are a) externally visible, and b) actually under the control of the system. However, it is sometimes helpful to define a Lyapunov also in terms of non-output variables, especially if the output variables show non-monotonic behavior. The ACC provided as a running example is such a case.

For the stability analysis, we will apply Lyapunov functions to the closed loop only, i.e., the input variables of the system to be analyzed will be viewed as external disturbances, and the only local variables will be those of the controller, as the local variable set of the plant is per definition empty. Any sensor or actuator variables will also appear as output variables in the closed loop, as per Def. 4.

For a vector $v \in \mathbb{R}^n$, defined $v^{(i)}$ as the i -th component of v . Next, we give the verification conditions for basic components, given a plant P .

Theorem 1 (Verification Conditions for Basic Components). *Let C be a basic component with implementation semantics I as defined in Def. 10, with an admissible hybrid automaton H_C , which just consists of a single mode and no discrete transitions. Define $H = (\{m\}, Var^{loc}, Var^{in}, Var^{out}, \emptyset, R^C, R^I, \Phi, \Theta)$ as the hybrid automaton obtained by the parallel composition $H_C \parallel P$. For a valuation of the variables $Var^{in} \cup Var^{out}$, define the vector $X^{IO} \in \mathbb{R}^{|Var^{in}|+|Var^{out}|}$. Define the parameter space for the Lyapunov functions as \mathbb{R}^{r_C} for an arbitrary r_C . Define the predicate P_C^{Lyap} on the elements $\theta_C^{(i)}$ of a $\theta_C \in \mathbb{R}^{r_C}$ as*

$$P_C^{Lyap} := \forall 1 \leq i \leq r_C : \theta_C^{(i)} \geq 0 \wedge \exists 1 \leq i \leq r_C : \theta_C^{(i)} > 0.$$

If

1. $\bigvee_{\beta}(\text{reach}(H, \varphi_{\beta}^{\text{entry}} \wedge \Phi_{\beta})) \wedge \bigwedge_{\alpha} \text{reach}(H, \neg \varphi_{\alpha}^{\text{alarm}}, \Delta_{\alpha}) \implies \varphi_P^{\text{safe}} \wedge \varphi_C^{\text{assm}}$
2. $\forall \alpha \in A_C^{\text{out}}, \beta \in A_C^{\text{in}} : \varphi_{\beta}^{\text{entry}} \implies \neg \varphi_{\alpha}^{\text{alarm}}$
3. $\bigvee_{\beta}(\text{reach}(H, \varphi_{\beta}^{\text{entry}} \wedge \Phi_{\beta})) \wedge R^C(m) \wedge \varphi_C^{\text{assm}} \implies \varphi_C^{\text{prom}}$
4. $\forall \alpha \in C_C^{\text{out}} : \text{reach}(H, \partial \varphi_{\alpha}^{\text{alarmOn}}, \Delta_{\alpha}) \implies \varphi_{\alpha}^{\text{exit}}$, where $\partial \Phi$ is the border of a closed predicate Φ ,
5. there exist r_C Lyapunov functions $\mathcal{V}_C^i : \mathbb{R}^{|\text{Var}_C|} \rightarrow \mathbb{R}, 1 \leq i \leq r_C$ for H wrt. the same equilibrium $X_e^O \models \varphi_P^{\text{stable}}$ and function $f(X)$, with constants $k_1(C, i), k_2(C, i), k_3(C, i)$
6. for all $i, 1 \leq i \leq r_C$, there exist two constants $c_{\text{entry}}(C, i)$ and $c_{\text{stab}}(C, i)$ such that $(X \models (\bigvee_{\beta \in C_C^{\text{in}}} \varphi_{\beta}^{\text{entry}} \wedge \Phi_{\beta})) \implies \mathcal{V}_C^i(X) \leq c_{\text{entry}}(C, i)$ and $(\mathcal{V}_C^i(X) \leq c_{\text{stab}}(C, i) \implies X \models \varphi_P^{\text{stable}})$
7. there exists a $\theta_C \models P_C^{\text{Lyap}}$ such that

$$\Delta_C^{\text{stable}} \geq \frac{1}{\text{rate}_C(\theta_C)} \ln \left(\frac{\sum_i \theta_C^{(i)} c_{\text{entry}}(C, i)}{\sum_i \theta_C^{(i)} c_{\text{stab}}(C, i)} \right),$$

where

$$\text{rate}_C(\theta_C) = \frac{\sum \theta_C^{(i)} k_3(C, i)}{\sum \theta_C^{(i)} k_2(C, i)}$$

then $I \parallel P \models \text{SPEC}_C$.

The Lyapunov function conditions require a detailed explanation. First, assume that r_C is set to 1. Note that, in condition (O) of Definition 13, we are interested not only in convergence to $\varphi_P^{\text{stable}}$, but in convergence in bounded time. If this were not the case, then stability of the parallel composition between component and plant would already be proven if we could find a Lyapunov function for the system. To verify such a time bound, we additionally need to keep track of the decrease rate of the Lyapunov function value (i.e., the “energy”). This rate is characterized by the ratio $k_3(C, 1)/k_2(C, 1)$, which gives an exponential decrease rate of the Lyapunov function value over time. To bound the convergence time, we also need to know a bound on the initial Lyapunov function value upon activation ($c_{\text{entry}}(C, 1)$) and a Lyapunov function value that is low enough for $\varphi_P^{\text{stable}}$ to be fulfilled ($c_{\text{stab}}(C, 1)$). Together, these scalar values can be used to conservatively estimate a time bound for convergence, exploiting the exponential convergence of \mathcal{V}_C^1 .

If we do not intend to further compose C , then setting $r_C = 1$ is indeed enough, as a single Lyapunov function already guarantees the stability property. However, if we want to support further composition, we need to provide means to show that the newly composed component again guarantees $\varphi_P^{\text{stable}}$ in bounded time, i.e., that there also exists a Lyapunov function for this transition composition. However, our function \mathcal{V}_C^1 can also talk about local variables of C , which are invisible in the transition composition. Nevertheless, we have to provide verification conditions for transition compositions that guarantee that the “energy” does not increase when a switch to a new component takes place.

For this reason, a component has to communicate the Lyapunov function values it can have at two time instants: when it is activated and when it is deactivated. Since we cannot talk about local variables at the interface, we associate a *projection* of the Lyapunov function on the externally visible variables with each in- and output of the component. For the composition, we then compare the two projected functions for all in- and outputs to be connected to ensure a decrease of the energy upon switching.

Here lies also the motivation for setting r_C to a number larger than 1. In general, systems can have infinitely many possible Lyapunov functions. By picking just one function, we risk that, for the transition composition, the stability proof will not be possible. Usually only some Lyapunov functions for the sub-components will be suitable for verifying the non-increasingness conditions. It is therefore helpful for the sub-components to communicate as many Lyapunov function projections to the outside as reasonably possible. We can then exploit the conic closure property of Lyapunov functions to construct *infinitely many* different Lyapunov functions for the subcomponent, while trying to satisfy the non-increasingness conditions for all port connections of the composed component.

Proof (of Theorem 1). We have to show that $I_C \parallel P \models SPEC_C$ holds when the assumption of the theorem are given for I_C . To prove that we have to show that all of the requirements given in Def. 13 hold.

(A):

$$\begin{aligned}
Var_{I_C}^{out} &= Var_{H_C}^{out} \cup \{fail_C, active_C\} && // \text{Def. 10} \\
&\cup \{take_\beta \mid \beta \in A_C^{in}\} \\
&\cup \{b_\alpha, switch_\alpha \mid \alpha \in A_C^{out}\} \\
&= Var_{H_C}^{out} \cup C_C^{out} && // H_C \text{ admissible, Def. 9} \\
&= Var_C^{out} \cup C_C^{out}
\end{aligned}$$

(B):

$$\begin{aligned}
Var_{I_C}^{in} &= (Var_{H_C}^{in} \cup \{active_C, suspend_C\}) && // \text{Def. 10} \\
&\cup \{start_\beta, c_\beta \mid \beta \in A_C^{in}\} \\
&\cup \{take_\alpha \mid \alpha \in A_C^{out}\} \setminus Var_{I_C}^{out} \\
&= Var_{H_C}^{in} \cup C_C^{in} && // H_C \text{ admissible, Def. 9} \\
&= Var_C^{in} \cup C_C^{in}
\end{aligned}$$

(C) follows directly from the assumption 2 of the theorem.

(D) follows from the assumptions 1 and 2:

$$\begin{aligned}
\varphi_C^{entry} &\implies \bigvee_{\beta} \varphi_{\beta}^{entry} \\
&\implies \bigvee_{\beta} \varphi_{\beta}^{entry} \wedge \bigwedge_{\alpha} \neg \varphi_{\alpha}^{alarm} && // \text{ Assm. 2} \\
&\implies \bigvee_{\beta} \varphi_{\beta}^{entry} \wedge \bigwedge_{\alpha} \text{reach}(H, \neg \varphi_{\alpha}^{alarm}, \Delta_{\alpha}) \\
&\implies \bigvee_{\beta} \text{reach}(H, \varphi_{\beta}^{entry}) \wedge \bigwedge_{\alpha} \text{reach}(H, \neg \varphi_{\alpha}^{alarm}, \Delta_{\alpha}) \\
&\implies \varphi_P^{safe} \wedge \varphi_C^{assm} && // \text{ Assm. 1}
\end{aligned}$$

(E): This holds due to the construction of H_1 . The initial location is m_{inact} and it requires a $start_{\beta}$ to activate the component. Moreover, whenever the activation takes place H_1 checks whether the entry condition φ_{β}^{entry} is given. If this is not the case, then H_1 sets $fail_C$.

(F): For a fixed α we consider the automaton H_3 of Def. 10 and a run in which the φ_{α}^{alarm} becomes true while b_{α} is not set (during all discrete steps at this time point).

If H_3 is in mode *Fail* when the φ_{α}^{alarm} becomes true, then $fail_C$ is set and nothing is to show. Otherwise it is in a mode (X, Y) . Due to the construction of H_3 we have the invariant $\alpha \in X \iff b_{\alpha}$. So, we have to consider two cases: $\alpha \notin X \cup Y$ and $\alpha \in Y \setminus X$. In the first case, there is an urgent transition that becomes enabled as soon as φ_{α}^{alarm} becomes true provided that the component is active. Hence, we can conclude that b_{α} will be set within that time point. In the second case we know that $Y \neq \emptyset$ and by the construction of the urgent transitions H_3 will switch to a mode with $\alpha \notin X \cup Y$ (first case) without delay.

(G): For a fixed α we consider the automaton H_3 of Def. 10 and a run of C that violates the given property. That means that we have two points in time $t_1 \leq t_2$ (with $t_2 - t_1 < \mu_{\alpha}$) where b_{α} was set and reset, respectively. Due to the construction of H_3 it is clear that setting b_{α} resets the clock t_{α} to 0. There are three kinds of transitions resetting b_{α} . They are guarded by $take_{\alpha}$ or guarded by $t_{\alpha} \geq \mu_{\alpha}$ or not enabled because of the invariant $b_{\alpha} \iff \alpha \notin Y$.

(H): Satisfied by the construction of H_1 .

(I): The component can only reset *active* in the automaton H_2 . The only transition there require to have a *suspend*. Since this transition is urgent, a rising edge of *suspend* with deactivate the component.

(J): This holds because H_{β} ensures that a rising edge of c_{β} starts a timer t_{β} and the $take_{\beta}$ signal is given at latest when this timer reaches the time bound λ_{β} .

(K): Satisfied by H_{β} because setting $take_{\beta}$ requires c_{β} to be true. If c_{β} is reset, then $take_{\beta}$ will follow within finitely many discrete steps because of urgent transitions in H_{β} .

(L): Satisfied by H_{β} because resetting $take_{\beta}$ requires $\neg c_{\beta}$ to hold.

(M): Satisfied by H_β because setting *active* requires φ_β^{entry} to hold.

(N): Since the system is initially in φ_C^{entry} or is activated in φ_C^{entry} (M) we can conclude that the system state belongs to

$$\bigvee_{\beta} (reach(H, \varphi_\beta^{entry})) \wedge \bigwedge_{\alpha} reach(H, \neg\varphi_\alpha^{alarm}, \Delta_\alpha)$$

because *fail* is not set. Moreover, we have

$$reach(H_C, \varphi^{entry}) \wedge R^C(m)$$

as conservative approximation of the evolution of the system's state. With the assumptions 1 and 3 we get the desired implication immediately.

(O): Let $X(t)$ be a the projection of a trajectory of $I||P$ onto $Var_C \cup Var_P$ with $X(0) \models \bigvee_{\beta \in C_C^{in}} \varphi_\beta^{entry} \wedge \Phi_\beta$. Assume, without loss of generality, that *active*_C receives a rising edge at time 0. By condition (N) of Def. 13, we know that, as long as C is active, φ^{assm} holds, and therefore also $\Theta(m)$. Per verification condition 7, there exists a $\theta_C \in \mathcal{P}$ such that $\theta_C \models P_C^{Lyp}$ and the inequality on Δ_C^{stable} is fulfilled. Define $\mathcal{V}_C(\theta_C, X) := \sum_i \theta_C^{(i)} \mathcal{V}_C^i(X)$. Since for all i , $\mathcal{V}_C^i(X) \leq -k_3(C, i) \|X^O - X_e^O\|^2$, we obtain, by linear combination:

$$\mathcal{V}_C^\bullet(\theta_C, X) = \sum_i \theta_C^{(i)} \mathcal{V}_C^i(X) \leq \left(- \sum_i \theta_C^{(i)} k_3(C, i) \right) f(X).$$

In the same manner, we can derive that

$$\mathcal{V}_C(\theta_C, X) \leq \left(\sum_i \theta_C^{(i)} k_2(C, i) \right) f(X).$$

Since $rate_C(\theta_C) = \sum_i \theta_C^{(i)} k_3(C, i) / \sum_i \theta_C^{(i)} k_2(C, i)$ this gives us

$$\mathcal{V}_C^\bullet(\theta_C, X) \leq \left(- \sum_i \theta_C^{(i)} k_3(C, i) \right) f(X) \leq -rate_C(\theta_C) \mathcal{V}_C(\theta_C, X)$$

and therefore

$$\mathcal{V}_C(\theta_C, X(t)) \leq e^{-rate_C(\theta_C)t} \mathcal{V}_C(\theta_C, X(0))$$

for all t such that C remains active on the interval $[0, t]$. Furthermore, since

$$\Delta_C^{stable} \geq \frac{\sum_i \theta_C^{(i)} k_2(C, i)}{\sum_i \theta_C^{(i)} k_3(C, i)} \ln \left(\frac{\sum_i \theta_C^{(i)} c_{entry}(C, i)}{\sum_i \theta_C^{(i)} c_{stab}(C, i)} \right),$$

we have that

$$\begin{aligned}
\mathcal{V}_C(\theta_C, X(\Delta_C^{stable})) &\leq \exp\left(\frac{-rate_C(\theta_C)}{rate_C(\theta_C)} \ln\left(\frac{\sum_i \theta_C^{(i)} c_{entry}(C, i)}{\sum_i \theta_C^{(i)} c_{stab}(C, i)}\right)\right) \mathcal{V}_C(\theta_C, X(0)) \\
&= \exp\left(\ln\left(\frac{\sum_i \theta_C^{(i)} c_{stab}(C, i)}{\sum_i \theta_C^{(i)} c_{entry}(C, i)}\right)\right) \mathcal{V}_C(\theta_C, X(0)) \\
&= \frac{\sum_i \theta_C^{(i)} c_{stab}(C, i)}{\sum_i \theta_C^{(i)} c_{entry}(C, i)} \mathcal{V}_C(\theta_C, X(0)).
\end{aligned}$$

Since $X(0) \models (\bigvee_{\beta \in C^{in}} \varphi_\beta^{entry} \wedge \Phi_\beta)$, this implies

$$\mathcal{V}_C(\theta_C, X(0)) \leq \sum_i \theta_C^{(i)} c_{entry}(C, i),$$

and we obtain

$$\mathcal{V}_C(\theta_C, X(\Delta_C^{stable})) = \sum_i \theta_C^{(i)} \mathcal{V}_C^i(X(\Delta_C^{stable})) \leq \sum_i \theta_C^{(i)} c_{stab}(C, i).$$

Note that this implies that there exists an i with $\mathcal{V}_C^i(X(\Delta_C^{stable})) \leq c_{stab}(C, i)$. This gives us $X(\Delta_C^{stable}) \models \varphi_P^{stable}$. Since $\mathcal{V}_C^\bullet(\theta_C, X) < 0$, we obtain that

$$\square(active_C \implies ((\diamond_{\Delta_C^{stable}}(\square\varphi_C^{stable}))) \text{ UNLESS } (\neg active_C)).$$

(P),(Q): Immediately clear by the construction of the transitions of H_3 setting $switch_\alpha$.

(R): By the construction of H_3 we know that whenever a $switch_\alpha$ becomes true this was preceded by setting b_α at most $\Delta_\alpha - \tau$ time units before. The rising edge of b_α was induced by an urgent transition of H_3 because φ_α^{alarm} became true. Hence, the time point in which $switch_\alpha$ becomes true belongs to the set $reach(H, \partial\varphi_\alpha^{alarm}, \Delta_\alpha)$. Because of assumption 4 we can conclude that φ_α^{exit} holds at this time point.

(S): The first reason for failure is managed by H_3 . There is only one kind of transition that sets $fail_C$ in this automaton. These transitions are guarded by a constraint of the form $t_\alpha \geq \Delta_\alpha$ for an $\alpha \in A_C^{out}$. By the construction of H_3 it is clear that this transition is only enabled if there has been a preceding period of duration Δ_α where b_α was set.

The second reason for a failure is managed by H_β . If this automaton observes that φ_β^{entry} is not given when it tries to activate the component, then it sets $fail_C$. This is the only way for H_β to do this.

(T): Obvious by the construction of all automata in the semantics: There is no transition setting $fail_C$ to 0.

The following definition gives the induction invariants of a basic component C , that is, all additional information that has to be provided by the component, in order to allow stability proofs of transition compositions containing C .

Definition 15 (Induction Invariants for Basic Components). For each basic component, the following induction invariants need to be provided, to facilitate further composition:

1. for all $\alpha \in A_C^{out}$ and $1 \leq j \leq r_C$, a function $\mathcal{V}_\alpha^j : \mathbb{R}^{|Var_C^{in}|+|Var_C^{out}|} \rightarrow \mathbb{R}$ such that $(X^L, X^{IO}) \models reach(H, \varphi_\alpha^{exit}, \tau) \wedge \bigvee_\beta (reach(H, \varphi_\beta^{entry} \wedge \Phi_\beta)) \implies \mathcal{V}_\alpha^j(X^{IO}) \leq \mathcal{V}_C^j(X)$,
2. for all $\beta \in A_C^{in}$ and $1 \leq j \leq r_C$, a function $\mathcal{V}_\beta^j : \mathbb{R}^{|Var_C^{in}|+|Var_C^{out}|} \rightarrow \mathbb{R}$ such that $(X^L \models \Phi_\beta \wedge X^{IO} \models \varphi_\beta^{entry}) \implies \mathcal{V}_\beta^j(X^{IO}) \geq \mathcal{V}_C^j(X)$,
3. for all $1 \leq j \leq r_C$, the constants $k_2(C, j)$ and $k_3(C, j)$ of the Lyapunov functions \mathcal{V}_C^j ,
4. for all $1 \leq j \leq r_C$, the constants $c_{entry}(C, j)$ and $c_{stab}(C, j)$.

Example (cont.) To verify that the example ACC system satisfies its component interface specification, we need to prove that all the verification conditions in Theorem 1 hold. Using component PI as an example, the closed-loop automaton $H = H_{PI}||P$ can be described by the following differential inclusion:

$$\begin{aligned} x(t)^\bullet &= v(t) \\ v(t)^\bullet &\in co\{-0.001025x(t) - 0.0533v(t), -0.0009075x(t) - 0.0507v(t)\} \end{aligned}$$

where co denotes the convex hull. We arrive at this description by eliminating the variables that do not appear in differential equations, but only in invariants, namely a and s .

First, we must compute Lyapunov functions for H , such that verification conditions 5, 6 and 7 are fulfilled. To compute quadratic Lyapunov functions for linear or affine dynamics, *semidefinite programming (SDP)* [BEFB94] tools can be used, for instance CSDP [Bor99] or SeDuMi [RPS99]. In a nutshell, the problem of finding a Lyapunov function according to some parameterized template is mapped onto a nonlinear, but still convex optimization problem that can be solved numerically [Pet99]. For polytopic differential inclusions like the one given above, it is sufficient to identify a Lyapunov function that works for the extremal dynamics (i.e., $v(t)^\bullet = -0.001025x(t) - 0.0533v(t)$ and $v(t)^\bullet = -0.0009075x(t) - 0.0507v(t)$). One such example Lyapunov function for H is

$$\mathcal{V}_{PI}^1(x, v) = 46.7455x^2 + 1101.9vx + 20729v^2,$$

with constants $k_2 = 1$ and $k_3 = 0.018$. Here, the function $f(X)$ was simply set to the Lyapunov function itself and the equilibrium point was $v = 0$. To compute associated constants $c_{stab}(PI, 1)$ and $c_{entry}(PI, 1)$, one needs to find a value c , such that the contour line $\mathcal{V}_{PI}^1(x, v) = c$ is entirely within φ_P^{stable} , and an upper bound on $\mathcal{V}_{PI}^1(x, v)$ for $x \models \bigvee_{\beta \in C_{PI}^{in}} (\varphi_\beta^{entry} \wedge \Phi_\beta)$. These are simple one-dimensional optimization problems. For \mathcal{V}_{PI}^1 , possible values are $c_{entry}(PI, 1) = 4700000$ and $c_{stab}(PI, 1) = 50000$. From here, it is straightforward to show that the inequality on Δ_{PI}^{stable} is also satisfiable. Since we computed only one Lyapunov function, θ_{PI} is simply a scalar, and by setting $\theta_{PI} = 1$, the inequality is satisfied.

For several verification conditions, we require a reach set of H , starting from $\varphi_{\beta_{PI}}^{entry} \wedge \Phi_{\beta_{PI}}$. There are various ways of arriving at such a set, for example the tool PHAVer [Fre08], or barrier certificates computed from Lyapunov functions [PJ04]. The latter method is especially suitable in this case, as we already have a barrier certificate: we know that $\mathcal{V}_{PI}^1(x, v) \leq c_{entry}(PI, 1)$, as long as PI is active. This sublevel set forms an ellipsoid in the state space, which serves as an over-approximation of the reach set. For convenience, we again over-approximate this ellipsoid by a box and obtain

$$-500 \leq x \leq 500 \wedge -30 \leq v \leq 30.$$

Now we are ready to show the remaining verification conditions, which are linear arithmetic constraints that can be proven with tools like iSat⁵ (including the time bounded reachability computation in condition 4) of Theorem 1.

To allow for further composition, we also need to provide the remaining induction invariants, namely projections of $\mathcal{V}_{PI}^1(x, v)$ onto the inports and outports of PI .

For the inport $\beta := \beta_{PI}$, this is a function $\mathcal{V}_{\beta}^1(v)$ that is larger than all possible values of $\mathcal{V}_{PI}^1(x, v)$ when PI is activated. Since $\Phi_{\beta} = (x = 0)$, we can just substitute $x = 0$ and obtain $\mathcal{V}_{\beta}^1(v) = 20729v^2$.

Similarly, for the outports $\alpha_1 := \alpha_{PI}^1$ and $\alpha_2 := \alpha_{PI}^2$, we must provide functions $\mathcal{V}_{\alpha_1}^1(v)$ and $\mathcal{V}_{\alpha_2}^1(v)$, bounding the values of $\mathcal{V}_{PI}^1(x, v)$ from below when PI is deactivated via the corresponding outport. Here, we can use the unbounded reach set approximation we already computed to bound the values of x and a τ -bounded reach set computation on $\varphi_{\alpha_i}^{exit}$ to bound the values of v . For simplicity, we use constant functions $\mathcal{V}_{\alpha_1}^1(v) = \mathcal{V}_{\alpha_2}^1(v) = 2400000$ here.

The same procedure can then be applied to the other basic components *BRAKE* and *ACCELERATE*. Here, possible Lyapunov functions are the linear functions

$$\mathcal{V}_{BRAKE}^1(v) = v$$

and

$$\mathcal{V}_{ACCELERATE}^1(v) = -v.$$

Next, we give the verification conditions for composed components, which are the result of a transition composition of either basic or composed components. For all subcomponents involved in such a transition composition, the principle of information hiding stipulates that we do not have access to its internals, but only an external view of the subcomponent. This view consists of the information in a subcomponent's external interface specification, of which we assume that it is fulfilled, and the induction invariants listed above. For the Lyapunov function projections given in the induction invariant, the verification conditions imply the satisfaction of a nonincreasingness condition upon a component switch. To

⁵ <http://isat.gforge.avacs.org/index.html>

guarantee safety, it is sufficient to require that the system states at which a switch can occur always fulfill the entry condition of the component we are switching to. Furthermore, we need to enforce constraints that make sure the composed component does not guarantee a behavior that the constituent subcomponents cannot guarantee themselves. This involves the different timing variables and the system's promises on the dynamics.

In the following, we assume that each outport of a composed components is only connected to a single outport of a subcomponent. This is done purely to simplify the formal description of the verification conditions. If two subcomponent outports α_i are connected to one outport α of a composed component, one can, for the purpose of the analysis that follows, simply duplicate the outport α . Therefore, this is not a real limitation in the analysis.

Theorem 2 (Verification Conditions for Composed Components). *Let C be a composed component with $C = \mathcal{S}_{(\mathcal{P}, \mathcal{Q})}(C_1, \dots, C_n)$ with semantics I_C . Define the parameter space for the Lyapunov functions as $\mathcal{R} = \mathbb{R}^{r_{C_1}} \times \dots \times \mathbb{R}^{r_{C_n}}$ and let $r_C > 0$. For a vector $\theta_C \in \mathcal{R}$, we refer to the subvector containing the entries pertaining to C_i as $\theta_{C|C_i} \in \mathbb{R}^{r_{C_i}}$. Define the predicate P_C^{Lyap} on the elements $\theta_{C|C_i}^{(j)}$ of a $\theta_C \in \mathcal{R}$ as*

$$\begin{aligned} P_C^{Lyap} := & \forall i : (\forall 1 \leq j \leq r_C : \theta_{C|C_i}^{(j)} \geq 0 \wedge \exists 1 \leq j \leq r_{C_i} : \theta_{C|C_i}^{(j)} > 0) \wedge \\ & \bigwedge \forall (\alpha, \{(\beta_1, g_1, \mathcal{A}_1), \dots, (\beta_k, g_k, \mathcal{A}_k)\}, \cdot) \in \mathcal{P}, \forall 1 \leq i \leq k : \\ & \forall X^{IO} \models reach(\varphi_{C(\alpha)}^{prom}, \varphi_{\alpha}^{exit} \wedge g_i, \tau) : \\ & \sum_j \theta_{C|C(\alpha)}^{(j)} \nu_{\alpha}^j(X^{IO}) \geq \sum_j \theta_{C|C(\beta_i)}^{(j)} \nu_{\beta_i}^j(\mathcal{A}_i(X^{IO})). \end{aligned}$$

If:

1. $\forall i : I_{C_i} \parallel P \models SPEC_{C_i}$, where I_{C_i} is the semantics of C_i ,
2. $\forall \alpha \in A_C^{out}, \beta \in A_C^{in} : \varphi_{\beta}^{entry} \implies \neg \varphi_{\alpha}^{alarm}$,
3. $\forall (\alpha, \{(\beta_1, g_1, \mathcal{A}_1), \dots, (\beta_k, g_k, \mathcal{A}_k)\}, \{(\alpha_1, g'_1), \dots, (\alpha_l, g'_l)\}) \in \mathcal{P} :$
 - (a) $\forall i : \lambda_{\beta_i} \leq \mu_{\alpha}$
 - (b) $\forall i : \mu_{\alpha_i} \leq \mu_{\alpha} \wedge \Delta_{\alpha} \leq \Delta_{\alpha_i}$
 - (c) $\forall i : reach(\varphi_{C(\alpha)}^{prom}, \varphi_{\alpha}^{exit} \wedge g_i, \tau) \implies \varphi_{\beta_i}^{entry}[\mathcal{A}_i]$
 - (d) $\forall i : \varphi_{\alpha}^{exit} \wedge g'_i \implies \varphi_{\alpha_i}^{exit}$
 - (e) $\forall i : \varphi_{\alpha}^{alarmOn} \implies \varphi_{\alpha_i}^{alarmOn}$
4. $\forall \beta \in A_C^{in} : \lambda_{\beta} \geq \lambda_{\mathcal{Q}(\beta)}$
5. $\forall_i \varphi_{C_i}^{assm} \implies \varphi_C^{assm}$
6. $\forall_i \varphi_{C_i}^{prom} \implies \varphi_C^{prom}$
7. $\forall \beta \in A_C^{in} : \varphi_{\beta}^{entry} \implies \varphi_{\mathcal{Q}(\beta)}^{entry}$
8. require that P_C^{Lyap} is satisfiable and that the solution, given as a valuation of θ_C , satisfies

$$\Delta_C^{stable} \geq \frac{1}{rate_C(\theta_C)} \ln \left(\frac{\max_i \sum_j \theta_{C|C_i}^{(j)} c_{entry}(C_i, j)}{\min_i \sum_j \theta_{C|C_i}^{(j)} c_{stab}(C_i, j)} \right),$$

where

$$rate_C(\theta_C) = \min_i \frac{\sum_{j=1}^{r_{C_i}} \theta_{C|C_i}^{(j)} k_3(C_i, j)}{\sum_{j=1}^{r_{C_i}} \theta_{C|C_i}^{(j)} k_2(C_i, j)},$$

then $I_C \parallel P \models SPEC_C$.

Again, the same induction invariants as for basic components need to be provided. The Lyapunov function projections on in- and outports are derived from some solution θ_C of the constraint system P_C^{Lyap} , each of which represents a valid (but not explicitly specified) continuous energy function of the system, which cannot increase at its discontinuities. Remember that the θ_C serve as multipliers in a conic combination of Lyapunov functions. To prove stability of C , it is again sufficient to find a single θ_C solving P_C^{Lyap} that fulfills the timing constraint. With the same argument as for basic components, we however allow for the computation of r_C such solutions $\tilde{\theta}_{C,j}$. The Lyapunov function projections attached to the inports (outports) are the Lyapunov function projections of the connected inports (outports) of subcomponents C_i , but weighted by the solutions $\tilde{\theta}_{C,j}$. Remember that we assumed that each outport of C is only connected to one outport of a C_i – in this case, there is a one-to-one mapping of Lyapunov function projections. Due to the conic property of Lyapunov functions, any conic combination of the $\tilde{\theta}_{C,j}$ will again represent a solution of P_C^{Lyap} , which in turn implies the existence of a discontinuous, but decreasing energy function so that the propagation of information to the next higher level again works correctly. The rates and constants of the composed component can also be computed from the rates of the subcomponents C_i . Again, they are weighted by θ_C for each component, and then the worst case over all subcomponents is taken for the transition composition C .

Definition 16. *For each composed component, the following induction invariants need to be provided, to facilitate further composition:*

- for each $1 \leq j \leq r_C$ where r_C is a positive integer:
 - for all outports α a function $\mathcal{V}_\alpha^j : \mathbb{R}^{|Var_C^{in}|+|Var_C^{out}|} \rightarrow \mathbb{R}$
 - for all inports β a function $\mathcal{V}_\beta^j : \mathbb{R}^{|Var_C^{in}|+|Var_C^{out}|} \rightarrow \mathbb{R}$
- such that there exists a $\tilde{\theta}_{C,j} \models P_C^{Lyap}$ with subvectors $\tilde{\theta}_{C|C_i,j}$ for each subcomponent C_i , and
- $\mathcal{V}_\alpha^j = \sum_{k=1}^{r_{C(\alpha')}} \tilde{\theta}_{C|C(\alpha'),j}^{(k)} \mathcal{V}_{\alpha'}^j$, where α' is the unique outport connected to α ,
 - $\mathcal{V}_\beta^j = \sum_{k=1}^{r_{C(Q(\beta))}} \tilde{\theta}_{C|C(Q(\beta)),j}^{(k)} \mathcal{V}_{Q(\beta)}^j$.
- for each $1 \leq j \leq r_C$
 - $k_2(C, j) = \max_i \sum_{k=1}^{r_{C_i}} \tilde{\theta}_{C|C_i,j}^{(k)} k_2(C_i, k)$,
and $k_3(C, j) = \min_i \sum_{k=1}^{r_{C_i}} \tilde{\theta}_{C|C_i,j}^{(k)} k_3(C_i, k)$
 - $c_{entry}(C, j) = \max_i \sum_{k=1}^{r_{C_i}} \tilde{\theta}_{C|C_i,j}^{(k)} c_{entry}(C_i, k)$,
and $c_{stab}(C, j) = \min_i \sum_{k=1}^{r_{C_i}} \tilde{\theta}_{C|C_i,j}^{(k)} c_{stab}(C_i, k)$

Proof (of Theorem 2). We have to show that $I_C \parallel P \models SPEC_C$ holds when the assumption of the theorem are given for I_C . To prove that we have to show that all of the requirements given in Def. 13 hold.

(A):

$$\begin{aligned}
Var_{I_C}^{out} &= \bigcup_i Var_{C_i}^{out} \cup Var_{H_C}^{out} \cup Var_{H_P}^{out} \cup Var_{H_Q}^{out} && // \text{Def. 11} \\
&\supseteq Var_{C_1}^{out} \cup \{fail_C, active_C\} \\
&\cup \{take_\beta \mid \beta \in A_C^{in}\} \cup \{b_\alpha, switch_\alpha \mid \alpha \in A_C^{out}\} \\
&= Var_C^{out} \cup C_C^{out}.
\end{aligned}$$

(B):

$$\begin{aligned}
Var_{I_C}^{in} &= \left(\bigcup_i Var_{C_i}^{in} \cup Var_{H_C}^{in} \cup Var_{H_P}^{in} \cup Var_{H_Q}^{in} \right) \setminus Var_{I_C}^{out} \\
&\supseteq \left(Var_{C_1}^{in} \cup \{active_C, suspend_C\} \right. \\
&\quad \left. \cup \{start_{\beta, c_\beta} \mid \beta \in A_C^{in}\} \cup \{take_\alpha \mid \alpha \in A_C^{out}\} \right) \setminus Var_{I_C}^{out} \\
&= Var_{C_1}^{in} \cup C_C^{in} \\
&= Var_C^{in} \cup C_C^{in}.
\end{aligned}$$

(C) follows directly from the assumption 2 of the theorem.

(D) follows from the assumptions 1, 6 and 8 in conjunction with the properties of the component's interface specifications:

$$\begin{aligned}
\varphi_C^{entry} &\implies \bigvee_\beta \varphi_\beta^{entry} \\
&\implies \bigvee_\beta \varphi_{Q(\beta)}^{entry} && // \text{Assm. 8} \\
&\implies \bigvee_\beta \varphi_{C(Q(\beta))}^{assm} && // \text{Assm. 1, (D) for } C_i \\
&\implies \bigvee_i \varphi_{C_i}^{assm} \\
&\implies \varphi_C^{assm} && // \text{Assm. 6}
\end{aligned}$$

$$\begin{aligned}
\varphi_C^{entry} &\implies \bigvee_\beta \varphi_{\beta, C}^{entry} \\
&\implies \bigvee_\beta \varphi_{Q(\beta)}^{entry} && // \text{Assm. 8} \\
&\implies \bigvee_i \varphi_{C_i}^{entry} \\
&\implies \varphi_P^{safe} && // \text{(D) for } C_i
\end{aligned}$$

(E): This holds due to the construction of H_T . All transitions setting $active_C$ require that there is a β such that $start_\beta \wedge \varphi_\beta^{entry}$ holds.

(F): When φ_α^{alarm} becomes true, a connected $\alpha_i \in A_{C_i}^{out}$ exists. For this outpost we have this property already, hence we know that b_{α_i} is set because $\varphi_\alpha^{alarm} \implies \varphi_{\alpha_i}^{alarm}$ is given in 3. As H_P forwards b_{α_i} to b_α we know that this holds.

(G): This holds again due to H_P which forwards both the b_{α_i} from a component C_i to b_α and the corresponding $take_\alpha$ to $take_{\alpha_i}$. Since C_i satisfies (G) we know that b_{α_i} holds for at least μ_{α_i} seconds before it can be reset without a $take_{\alpha_i}$ signal. As $\mu_{\alpha_i} \leq \mu_\alpha$ is given in assumption 3 we can conclude (G) for the outpost α .

(H), (I): Both properties are satisfied because of H_T 's construction.

(J): This holds because H_Q ensures that a rising edge of c_β produces a rising edge of $c_{Q(\beta)}$ and with (J) for this component we can conclude that there is a $take_{Q(\beta)}$ after at most $\lambda_{Q(\beta)}$ seconds. This is forwarded by H_Q . With $\lambda_{Q(\beta)} \leq \lambda_\beta$ of assumption 4 this property holds.

(K): Again this holds because H_Q forwards the signals c_β and $take_{Q(\beta)}$ to $c_{Q(\beta)}$ resp. $take_\beta$ appropriately and the component itself satisfies this requirement.

(L): This given because H_Q just forwards both c_β (to $c_{Q(\beta)}$) and $take_{Q(\beta)}$ (to $take_\beta$). As (L) holds for the C_i the import $Q(\beta)$ belongs to, this property can be transferred directly to C .

(M): This is satisfied because the composition cannot be activated by a $start_\beta$ while $\neg\varphi_\beta^{entry}$ holds. The reason is that in this case H_T produces a $fail_C$ signal.

(N): Whenever C is active then exactly one of its components C_i is active. Exploiting (N) for C_i we get $\varphi_P^{safe} \wedge \varphi_{C_i}^{assm} \wedge \varphi_{C_i}^{prom}$. With the assumptions 6 and 7 we get the $\varphi_P^{safe} \wedge \varphi_C^{assm} \wedge \varphi_C^{prom}$.

(O): Let $X(t)$ be a the projection of a trajectory of $I_C \parallel P$ onto $Var_C \cup Var_P$. Assume, without loss of generality, that $active_C$ receives a rising edge at time 0. Let (C_l) be the finite or infinite sequence of active subcomponents of C along trajectory $X(t)$, in order of activation, and let (t_l) be the sequence of activation times. If (t_l) is finite, set $t_{l+1} = \infty$. Since $\tau > 0$, we know that (t_l) is strictly increasing. For $l > 1$, define $prev(X(t_l)) = \lim_{t \uparrow t_l} X(t)$. For a time t , let $C(t)$ be the unique subcomponent that is active at time t .

Inductively define the functions $\mathcal{V}_{C'}(\theta_{C'}, X(t))$ and $\mathcal{V}_{C'}^j(X(t))$ as follows:

- for a basic component C' , $\mathcal{V}_{C'}(\theta_{C'}, X(t)) := \sum_k \theta_{C'}^{(k)} \mathcal{V}_{C'}^k(X(t))$,
where $\mathcal{V}_{C'}^k(X(t))$ is the function as given in the verification conditions for basic components.
- for a composed component C' , $\mathcal{V}_{C'}(\theta_{C'}, X(t)) := \sum_k \theta_{C'|C'(t)}^{(k)} \mathcal{V}_{C'(t)}^k(X(t))$,
and $\mathcal{V}_{C'}^j(X(t)) = \sum_k \tilde{\theta}_{C'|C'(t),j}^{(k)} \mathcal{V}_{C'(t)}^k(X(t))$

Now, show that for all $\theta_C \models P_C^{Lyp}$, the following hold:

- 1) for all $t \in \mathbb{R} \cup \{\infty\}$ with $\forall t' \in [0, t) : active_C(t')$:

$$\mathcal{V}_C(\theta_C, X(t)) \leq e^{-rate_C(\theta_C)t} \mathcal{V}_C(\theta_C, X(0)).$$

- 2) $\mathcal{V}_C(\theta_C, X) \leq \min_l \sum_j \theta_{C|C_l}^{(j)} c_{stab}(C_l, j) \implies \varphi_P^{stable}$
3) $\mathcal{V}_C(\theta_{C|C_l}, X(0)) \leq \max_l \sum_j \theta_{C|C_l}^{(j)} c_{entry}(C_l, j)$

First, show that 1) holds for C . The proof is by induction. For basic components C_l , 1) holds, which was shown in the proof of Theorem 1, For composed components C_l , assume per induction that 1) holds. In particular, that gives us

$$\mathcal{V}_{C_l}^j(X(t)) \leq e^{-rate_{C_l}(\tilde{\theta}_{C_l, j})t} \mathcal{V}_{C_l}^j(X(0)).$$

Define

$$\mathcal{V}_{C_l}(\theta_{C|C_l}, X) := \sum_j \theta_{C|C_l}^{(j)} \mathcal{V}_{C_l}^j(X)$$

Then, per linear combination of the Lyapunov constraints, $\forall \theta_C \models P_C^{Lyap}, \forall t' \in [t_l, t_{l+1})$:

$$\begin{aligned} \mathcal{V}_{C_l}(\theta_{C|C_l}, X(t')) &\leq e^{-\sum_j \theta_{C|C_l}^{(j)} rate_{C_l}(\tilde{\theta}_{C_l, j})t'} \mathcal{V}_{C_l}(\theta_{C|C_l}, X(t_l)) \\ &= e^{-\sum_j \theta_{C|C_l}^{(j)} \frac{k_3(C_l, j)}{k_2(C_l, j)} t'} \mathcal{V}_{C_l}(\theta_{C|C_l}, X(t_l)) \\ &\leq e^{-rate_C(\theta_C)t'} \mathcal{V}_{C_l}(\theta_{C|C_l}, X(t_l)) \end{aligned}$$

Therefore, we know that $\mathcal{V}_C(\theta_C, X(t))$ will decrease according to $rate_C(\theta_C)$, whenever no component switch occurs. Now we need to make sure that the switches will never result in an increase of $\mathcal{V}_C(\theta_C, X(t))$ at any t_l . Observe that there exists a transition $(\alpha, \{(\beta_1, g_1, \mathcal{A}_1), \dots, (\beta_k, g_k, \mathcal{A}_k)\}, \cdot) \in \mathcal{P}$, such that, by closedness of $reach(\varphi_{C(\alpha)}^{prom}, \varphi_\alpha^{exit} \wedge g_i, \tau)$, for all t_l , we have $prev(X(t_l)) \in reach(\varphi_{C(\alpha)}^{prom}, \varphi_\alpha^{exit}, \tau)$. Satisfaction of P_C^{Lyap} implies that, for all transitions:

$$\begin{aligned} \forall 1 \leq j \leq k : \forall X^{IO} \models reach(\varphi_{C(\alpha)}^{prom}, \varphi_\alpha^{exit} \wedge g_j, \tau) : \\ \sum_i \theta_{C|C(\alpha)}^{(i)} \mathcal{V}_\alpha^i(X^{IO}) \geq \sum_i \theta_{C|C(\beta_j)}^{(i)} \mathcal{V}_{\beta_j}^i(\mathcal{A}_j(X^{IO})), \end{aligned}$$

which gives us

$$\begin{aligned} \forall t_l, l > 1 : \mathcal{V}_{C_{l-1}}(\theta_{C|C_{l-1}}, prev(X(t_l))) &\geq \sum_j \theta_{C|C_{l-1}}^{(j)} \mathcal{V}_\alpha^j(prev(X^{IO}(t_l))) \\ &\geq \sum_j \theta_{C|C_l}^{(j)} \mathcal{V}_{\beta_j}^j(\mathcal{A}_j(X^{IO}(t_l))) \geq \mathcal{V}_{C_l}(\theta_{C|C_l}, X(t_l)). \end{aligned}$$

Together, this results in the desired inequality:

$$\forall t' \in [0, t) : active_C(t') \implies \mathcal{V}_C(\theta_C, X(t)) \leq e^{-rate_C(\theta_C)t} \mathcal{V}_C(\theta_C, X(0))$$

Now show 2), again by induction. For basic components C_l , we know that $\mathcal{V}_{C_l}^j(X) \leq c_{stab}(C_l, j) \implies \varphi_P^{stable}$. For composed components C_l , assume that 2) holds, and this follows directly, by setting $\theta_{C_l} = \tilde{\theta}_{C_l, j}$. If

$$\mathcal{V}_C(\theta_C, X(t)) \leq \min_l \sum_j \theta_{C|C_l}^{(j)} c_{stab}(C_l, j),$$

then this implies

$$\sum_j \theta_{C|C(t)}^{(j)} \mathcal{V}_{C(t)}^j(X(t)) \leq \sum_j \theta_{C|C(t)}^{(j)} c_{stab}(C(t), j)$$

For this inequality to hold, there must exist at least one j with

$$\mathcal{V}_{C'(t)}^j(X(t)) \leq c_{stab}(C(t), j),$$

which implies that $X(t) \models \varphi_P^{stable}$.

The proof of 3) is also done inductively. Again, for basic components C_l we know that $\mathcal{V}_{C_l}^j(X_l) \leq c_{entry}(C_l, j)$. For composed components, this property again follows, if we assume that 3) holds for C_l . Then

$$\begin{aligned} \mathcal{V}_C(\theta_C, X(0)) &= \sum_j \theta_{C|C(0)}^{(j)} \mathcal{V}_{C(0)}^j(X(0)) \\ &\leq \sum_j \theta_{C|C(0)}^{(j)} c_{entry}(C(0), j) \leq \max_l \sum_j \theta_{C|C_l}^{(j)} c_{entry}(C_l, j) \end{aligned}$$

Finally, we will use 1), 2) and 3) to prove the desired property. We know there exists a θ_C with:

$$\Delta_C^{stable} \geq \frac{1}{rate_C(\theta_C)} \ln \left(\frac{\max_i \sum_j \theta_{C|C_i}^{(j)} c_{entry}(C_i, j)}{\min_i \sum_j \theta_{C|C_i}^{(j)} c_{stab}(C_i, j)} \right),$$

$$\mathcal{V}_C(\theta_C, X(\Delta_C^{stable}))$$

$$\begin{aligned} &\leq \exp \left(\frac{-rate_C(\theta_C)}{rate_C(\theta_C)} \ln \left(\frac{\max_i \sum_j \theta_{C|C_i}^{(j)} c_{entry}(C_i, j)}{\min_i \sum_j \theta_{C|C_i}^{(j)} c_{stab}(C_i, j)} \right) \right) \mathcal{V}_C(\theta_C, X(0)) \\ &= \exp \left(\ln \left(\frac{\min_i \sum_j \theta_{C|C_i}^{(j)} c_{stab}(C_i, j)}{\max_i \sum_j \theta_{C|C_i}^{(j)} c_{entry}(C_i, j)} \right) \right) \mathcal{V}_C(\theta_C, X(0)) \\ &= \frac{\min_i \sum_j \theta_{C|C_i}^{(j)} c_{stab}(C_i, j)}{\max_i \sum_j \theta_{C|C_i}^{(j)} c_{entry}(C_i, j)} \mathcal{V}_C(\theta_C, X(0)) \end{aligned}$$

This implies that $\mathcal{V}_C(\theta_C, X(0)) \leq \max_i \sum_j \theta_{C|C_i}^{(j)} c_{entry}(C_i, j)$, and we obtain

$$\mathcal{V}_C(\theta_C, X(\Delta_C^{stable})) \leq \min_i \sum_j \theta_{C|C_i}^{(j)} c_{stab}(C_i, j).$$

which implies that $X(\Delta_C^{stable}) \models \varphi_P^{stable}$. Since $\mathcal{V}_C(\theta_C, X(t))$ is nonincreasing, we obtain

$$\Box(active_C \implies ((\Diamond_{\Delta_C^{stable}}(\Box \varphi_C^{stable}))) \text{ UNLESS } (\neg active_C)).$$

(P),(Q): Immediately clear by the construction of the transitions of $H_{\mathcal{P}}$ setting $switch_{\alpha}$.

(R): By the construction of $H_{\mathcal{P}}$ we know that whenever a $switch_{\alpha}$ becomes true, this happens at the same time point as a $switch_{\alpha'}$ of a component C_i becomes true with α' being connected to α with guard g . Since (R) holds for this α' we know that $\varphi_{\alpha'}^{exit}$ must hold. The guard g must also be satisfied because otherwise the $switch_{\alpha}$ would not be set by $H_{\mathcal{P}}$. In sum, we have $\varphi_{\alpha'}^{exit} \wedge g$ which allows us to conclude that φ_{α}^{exit} must hold because of assumption 3.

(S): If C fails, then this can be due to a component C_i that failed. In this case we know that (S) holds for C_i and we can conclude that there is an $\alpha_i \in A_{C_i}^{out}$ such that b_{α_i} was set at least Δ_{α_i} time units before the failure or there was an unsatisfied entry condition $\varphi_{\beta_i}^{entry}$ of an inport β_i of C_i . In case of a time-out we know that the automaton $H_{\mathcal{P}}$ forwards b_{α_i} to an outport $\alpha' \in A_C^{out}$ the whole time by its construction. With assumption 3 we know that $b_{\alpha'}$ was set at least $\Delta_{\alpha'}$ before $fail_C$ is set. In case of an unsatisfied entry condition $\varphi_{\beta_i}^{entry}$ we know by 7 that $\varphi_{\beta'}^{entry}$ with $\mathcal{Q}(\beta') = \beta_i$ is a stronger condition. Hence, if this $\varphi_{\beta'}^{entry}$ is not met when $start_{\beta'}$ is set, then H_T will set $fail_C$. If it is satisfied, then follows that $\varphi_{\mathcal{Q}(\beta')}^{entry}$ is satisfied, too. Hence, the component C_i cannot set $fail_{C_i}$.

(T): Clear as $fail_C$ is the disjunction of all $fail_{C_i}$ and since all C_i satisfy (T) this property holds for $fail_C$, because there is no transition in H_T , $H_{\mathcal{P}}$, and $H_{\mathcal{Q}}$ resetting $fail_C$.

Example (cont.) To verify that the component C_2 fulfills its interface specification, we have to prove the verification conditions in Theorem 2. We have already shown verification condition 1. For verification condition 8, we again need to find Lyapunov function parameters. Since we just computed single Lyapunov functions for PI and $ACCELERATE$, the vector θ_{C_2} is of dimension 2, with one coefficient $\theta_{C_2}^1$ serving as a multiplier for the Lyapunov function pertaining to PI and the other coefficient $\theta_{C_2}^2$ for the Lyapunov function pertaining to $ACCELERATE$. The constraint system calls for the identification of values for θ_{C_2} , such that, whenever a new component is activated, there is no increase in the Lyapunov function value.

In particular, the constraint system looks like this:

$$P_{C_2}^{Lyap} = (\theta_{C_2}^1 \geq 0 \wedge \theta_{C_2}^2 \geq 0) \vee (\theta_{C_2}^1 > 0 \vee \theta_{C_2}^2 > 0)$$

$$\wedge \forall v \models reach(\varphi_{PI}^{prom}, \varphi_{\alpha_{PI}}^{exit}, \tau) : \theta_{C_2}^1 \mathcal{V}_{\alpha_{PI}}^1(v) \geq \theta_{C_2}^2 \mathcal{V}_{\beta_{ACCELERATE}}^1(v)$$

$$\wedge \forall v \models reach(\varphi_{ACCELERATE}^{prom}, \varphi_{\alpha_{ACCELERATE}}^{exit}, \tau) : \theta_{C_2}^2 \mathcal{V}_{\alpha_{ACCELERATE}}^1(v) \geq \theta_{C_2}^1 \mathcal{V}_{\beta_{PI}}^1(v)$$

Once the reach sets have been computed or overapproximated, this is again a constraint system that can be solved with SDP methods. A possible solution is $\theta_{C_2}^1 = 1$ and $\theta_{C_2}^2 = 150000$.

From this, we can now check whether the inequality on $\Delta_{C_2}^{stable}$ also holds, which it does, since

$$\begin{aligned} \Delta_{C_2}^{stable} &= 300 \\ &\geq \frac{1}{rate_{C_2}(\theta_{C_2})} \ln \left(\frac{\max\{\theta_{C_2}^1 c_{entry}(PI, 1), \theta_{C_2}^2 c_{entry}(ACCELERATE, 1)\}}{\min\{\theta_{C_2}^1 c_{stab}(PI, 1), \theta_{C_2}^2 c_{stab}(ACCELERATE, 1)\}} \right) \\ &= \frac{1}{rate_{C_2}(\theta_{C_2})} \ln \left(\frac{4700000}{50000} \right) = 252.4 \end{aligned}$$

where

$$rate_{C_2}(\theta_{C_2}) = \min \left\{ \frac{0.018}{1}, \frac{10000}{200000} \right\} = 0.018$$

In case this inequality would be violated, we could try to find a better solution to the above constraint system, since SDP based methods inherently support the specification of objective functions on the parameters.

The remaining verification conditions are either simple scalar inequalities or implications between non-parametric predicates and therefore easily verified.

5 Conclusion

We have presented a design methodology for hybrid systems, which supports component based incremental design processes and prepares the way for a fully distributed implementation of such controllers as relevant for Autosar based automotive development processes. In addressing this industrial need, we have developed a mathematical theory for incremental compositional verification of both safety and stability properties of hybrid controllers based on the key concept of hybrid interface specifications, and provided verification conditions both for basic components as well as hierarchically composed open systems. Concepts, methodology, and incremental verification have been illustrated using a simple automatic cruise control system as a running example.

While the presented methodology and verification approach is self-contained and of value in itself, we see several extensions which will be addressed in further work.

First, we would like to close the gap between idealized plant models and physical plants by a notion of robust plant refinement, which allows to measure degrees of deviation still tolerated without endangering the satisfaction of hybrid interface specifications. Much as safety analysis methods distinguish between nominal behavior and failure behaviors which nevertheless are still restricted by failure hypothesis, we expect to be able to characterize conditions under which we can then establish under non-nominal behavior within the failure hypothesis, that alarms are raised in time.

Secondly, we will extend the approach to handle parallel composition of components.

Acknowledgments

This research is carried out in the context of the transregional collaborative research project AVACS (www.avacs.org) – we thank our colleagues in the project area on Hybrid Systems for many inspiring discussions, and Uwe Waldmann for providing efficient methods for solving time bounded reachability problems appearing as verification conditions in Chapter 4.

In Memoriam

How can I express my thanks to Amir for all the inspiration he has given, for the sharing of concerns on many issues in life and politics, for the fun and the concerts, for his keen questioning, probing, for offering clean and concise perspectives on what seemed to be a jungle of problems?

He was to me like a scientific father, who was always open to new ideas, gently pushing in his humble and friendly style our thoughts in the right direction.

His impact on science is overwhelming, so is his impact on me.

Thanks, Amir

Werner

References

- [BEFB94] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics (SIAM), 1994.
- [Bor99] B. Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods and Software*, 10(1):613–623, 1999. <https://projects.coin-or.org/Csdp/>.
- [DMO⁺07] W. Damm, A. Mikschl, J. Oehlerking, E.-R. Olderog, Jun Pang, A. Platzer, M. Segelken, and B. Wirtz. Automating Verification of Cooperation, Control, and Design in Traffic Applications. In C.B. Jones, Zhiming Liu, and J. Woodcock, editors, *Formal Methods and Hybrid Real-Time Systems, Essays in Honor of Dines Bjørner and Chaochen Zhou on the Occasion of Their 70th Birthdays*, volume 4700 of *Lecture Notes in Computer Science*, pages 115–169. Springer-Verlag, 2007.
- [DPJ09] W. Damm, T. Peikenkamp, and B. Josko. Contract Based ISO CD 26262 Safety Analysis. In *SAE World Congress – Session on Safety-Critical Systems*, 2009.
- [Fre05] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 258–273. Springer-Verlag, 2005.

- [Fre06] G. Frehse. On Timed Simulation Relations for Hybrid Systems and Compositionality. In E. Asarin and P. Bouyer, editors, *Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 4202 of *Lecture Notes in Computer Science*, pages 200–214. Springer-Verlag, 2006.
- [Fre08] G. Frehse. PHAVer: algorithmic verification of hybrid systems past HyTech. *STTT – International Journal on Software Tools for Technology Transfer*, 10(3):263–279, 2008.
- [HMP01] T. Henzinger, M. Minea, and V.S. Prabhu. Assume-Guarantee Reasoning for Hierarchical Hybrid Systems. In M. di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 275–290, Rome, Italy, March 2001. Springer-Verlag.
- [JBS07] S. Jha, B.A. Brady, and S.A. Seshia. Symbolic Reachability Analysis of Lazy Linear Hybrid Automata. In J.-F. Raskin and P.S. Thiagarajan, editors, *Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 4763 of *Lecture Notes in Computer Science*, pages 241–256. Springer-Verlag, 2007.
- [JMM08] B. Josko, Q. Ma, and A. Metzner. Designing Embedded Systems using Heterogeneous Rich Components. In *Proceedings of the INCOSE International Symposium*, 2008.
- [Lya07] M.A. Lyapunov. Problème général de la stabilité du mouvement. *Ann. Fac. Sci. Toulouse*, 9:203–474, 1907. (Translation of a paper published in *Comm. Soc. Math. Kharkow*, 1893, reprinted *Ann. Math. Studies No. 17*, Princeton Univ. Press, 1949).
- [OT09] J. Oehlerking and O. Theel. Decompositional construction of Lyapunov functions for hybrid systems. In *12th International Conference on Hybrid Systems: Computation and Control*, *Lecture Notes in Computer Science*. Springer, 2009.
- [Pet99] S. Pettersson. *Analysis and Design of Hybrid Systems*. PhD thesis, Chalmers University of Technology, Gothenburg, 1999.
- [PJ04] S. Prajna and A. Jadbabaie. Safety Verification of Hybrid Systems Using Barrier Certificates. In R. Alur and G.J. Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 477–492. Springer-Verlag, 2004.
- [RPS99] O. Romanko, I. Pólik, and J. F. Sturm. *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*, 1999.
- [Sta01] T. Stauner. *Systematic Development of Hybrid Systems*. PhD thesis, Technische Universität München, 2001.
- [Sta02] T. Stauner. Discrete-time refinement of hybrid automata. In *Proc. 5th International Workshop on Hybrid Systems: Computation and Control*, *Lecture Notes in Computer Science*, pages 407–420. Springer, 2002.
- [TPL04] P. Tabuada, G. J. Pappas, and P. Lima. Compositional abstractions of hybrid control systems. *Discrete Event Dynamic Systems*, 14(2), 2004.