EPiC Series in Computing
Volume 48, 2017, Pages 65–74
ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems

A Benchmark for Component-based Hybrid Systems Safety Verification* (Benchmark Proposal)

Andreas Müller[1], Stefan Mitsch[2], Werner Retschitzegger[1], Wieland Schwinger[1], and André Platzer[2]

[1] Department of Cooperative Information Systems
Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria
{andreas.mueller,wieland.schwinger,werner.retschitzegger}@jku.at
[2] Computer Science Department
Carnegie Mellon University, Pittsburgh PA 15213, USA
{smitsch,aplatzer}@cs.cmu.edu

Abstract

At scale, formal verification of hybrid systems is challenging, but a potential remedy is the observation that systems often come with a number of natural components with certain local responsibilities. Ideally, such a compartmentalization into more manageable components also translates to hybrid systems verification, so that safety properties about the whole system can be derived from local verification results. We propose a benchmark consisting of a sequence of three case studies, where components interact to achieve system safety. The baseline for the benchmark is the verification effort from a monolithic fashion (i.e., the entire system without splitting it into components). We describe how to split the system models used in these case studies into components with local responsibilities, and what is expected about their interaction to guarantee system safety. The benchmark can be used to assess the performance, automation, and verification features of component-based verification approaches.

*Work partly funded by the Austrian Science Fund (FWF) P28187-N31. This research was sponsored by the AFOSR under grant number FA9550-16-1-0288. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

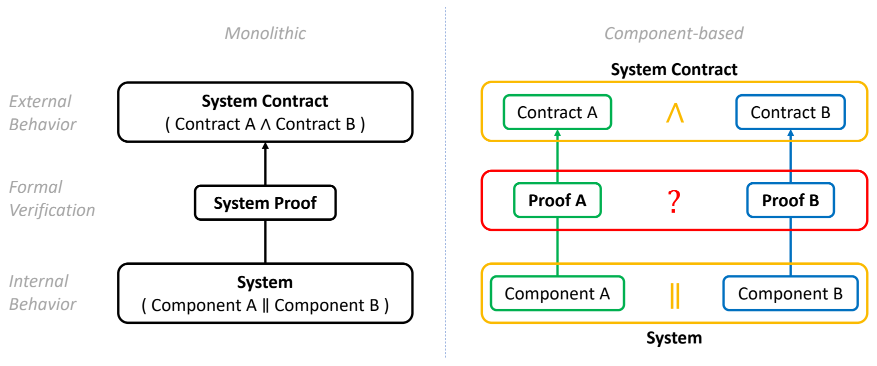


Figure 1: Monolithic verification combines systems before analysis, component-based verification derives system safety from component proofs

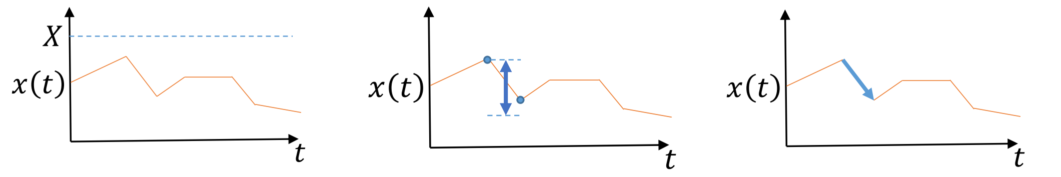
1 Introduction

As the complexity of hybrid system models increases, monolithic modeling and subsequent monolithic analysis techniques become increasingly challenging. However, since complex systems are typically composed of multiple interacting subsystems, *component-based modeling* can help alleviate the challenges. A proper verification benefit requires splitting the analysis into isolated questions about subsystems and their interaction. The ultimate goal is to verify properties of the entire system, without analyzing it as a whole, but by mere analysis of its components, cf. Fig. 1.

Since components in a system typically expose only some aspects of their internal behavior upon interaction, *contracts* can be used to make the relevant aspects of the individual behavior of components available in the system. A contract describes the (output) guarantees of a component under certain (input) assumptions. When composing components to form a system, compatibility of their interaction and communication is required, i.e., the assumptions of a component must be satisfied from the guarantees of another component providing input. Aspects such as communication delay and sensor uncertainty have to be considered. Component-based safety verification approaches (e.g., [5, 6]) verify safety properties about a system from just local contract compliance and compatibility of its components.

Especially in the context of hybrid I/O automata, a contract is often termed as an *abstraction* of a component. Such component-based safety verification approaches usually state that, given abstractions of multiple components (e. g., their contract), the composition of these abstractions is again an abstraction of the composition of the original components. Compositionality with respect to an abstraction is the central part of component-based verification. Common abstractions include labeled transition systems as abstractions for hybrid I/O automata (e. g., [1]) and first-order logic formulas as contracts for hybrid programs (e. g., [5, 6]).

In this paper, we propose a benchmark for component-based hybrid systems safety verification approaches. The benchmark comprises three case studies that were originally modeled and verified in a monolithic way, but that can be split into components easily. The first case study is a robot collision avoidance system inspired by [4]. For component-based analysis, the system is split into a robot component and a moving obstacle component: The robot drives in two-dimensional space and measures the position of the obstacle to avoid collision. The second case study is based on the European Train Control system presented in [9]. A radio-block controller component issues desired target speed and track permission to a train component,



(a) Global contract: bounded region (b) Change contract: bounded magnitude of change (c) Delay contracts: bounded rate of change

Figure 2: Bounds on the interaction between components expressible in contracts

which must stay inside the permitted area on the track. The third case study is an adaptive cruise control system inspired by [3]. A leader car communicates its position and speed to a follower car, which must guarantee to not collide with the leader. We describe all three case studies using differential dynamic logic (d$\mathcal{L}$), see [7, 8] for syntax and semantics.

The benchmark case studies are designed to asses

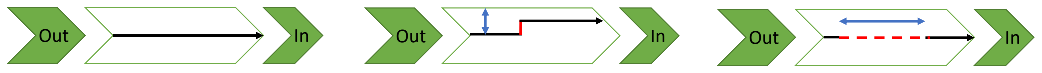
- the performance and degree of automation in comparison to monolithic analysis [3, 4, 9] and in comparison to our own component-based experiments [6], and
- the practical applicability of component-based verification approaches: continuous dynamics range from simple linear straight-line driving to non-linear curved trajectories; physical components interact with other physical entities (e. g., robot interacts with obstacle) or with purely virtual entities (e. g., train interacts with radio-block controller); interaction ranges from ideal-world loss-less and instantaneous interaction to sensor uncertainty to communication delay.

2 Contracts and Component Interaction

When working with component-based hybrid system verification, two major challenges arise: handling local component contracts and dealing with component interaction and communication. In a monolithic system, these interactions are baked into the system model, so a system contract defines properties of the initial states and overall system safety: starting from the initial state, all runs of the system stay in safe states. When working with components, however, a local contract for each component together with a way of handling component interaction is needed, and might include assumptions and guarantees about the communication and interaction between components. For example, a contract might restrict a vehicle's movement to prevent it from teleporting, or specify the acceptable degree of input sensor uncertainty.

Contracts. Local component contracts ensure local safety properties of a single component in isolation. They define assumptions and guarantees of a component regarding the interaction and communication with its environment. The benchmarks in this paper scale contract complexity as follows, cf. Fig. 2:

Global contracts restrict values to globally known (symbolic) regions, cf. Fig. 2a. For instance, a robot might be confined in a known, fixed area, e. g., the robot's position must always be in a fixed interval (e. g., $-9 \le x \le X$, where X is a symbolic constant).



(a) Instantaneous, loss-less communication (b) Sensor uncertainty (c) Communication delay

Figure 3: Communication between components

Change contracts restrict the magnitude of change (e. g., relate a value from the last control cycle or a previously communicated value with the current value), cf. Fig. 2b. For instance, a robot may guarantee to not request sudden changes in the sense that the current request x and the previous request x^- are not too far apart (e. g., $-9 \leq x - x^- \leq X$).

Delay contracts restrict the rate of change cf. Fig. 2c. For instance, a robot may guarantee to change its position according to speed, so the current position x and the previous position x^- are related by time t as in $-9 \cdot t \leq x - x^- \leq X \cdot t$.

Communication. Communication and interaction between components can be subject to uncertainty and delay, cf. Fig. 3:

Instantaneous, loss-less communication is often used as a first approximation of sensing and communication, and can be modeled by a direct assignment of variables, e. g., $\hat{x} := x$, cf. Fig. 3a.

Sensor uncertainty occurs, since sensor measurements often provide slightly off approximations of the actual values, cf. Fig. 3b. Even though the actual error might be unknown, the maximum error is often bounded (e. g., according to a sensor specification). Sensor uncertainty could, for instance, be modeled using a non-deterministically chosen error value λ ($\lambda := *$ means non-deterministically assigns any real value to λ), bounded by the maximum error Λ (the test $?\,|\lambda| \leq \Lambda$ ensures that the value of λ is between $-\Lambda$ and Λ), which distorts the communicated value, so $\lambda := *;\ ?\,|\lambda| \leq \Lambda;\ \hat{x} := x + \lambda$.

Communication delay results in accumulated error, e. g., a distance sensor in a car may report slightly outdated values, cf. Fig. 3c. Their error depends on speed and time delay.

The complexity of the benchmarks in the next section can be scaled according to these contract and communication options.

3 Benchmark

In this section, we present the case studies in more detail, using hybrid programs and differential dynamic logic [7, 8] in the paper, and KeYmaera X [2] syntax in the accompanying models. In general, each system consists of a *controller* followed by a *plant*, which are executed arbitrarily often. The control decisions steer continuous evolution in the plant. Here an ODE ($x' = \theta \& H$) describes a continuous evolution of x (x' denotes derivation with respect to the internal system time τ, i. e., $\frac{\partial x(\tau)}{\partial \tau}$) within the evolution domain H. For instance, a vehicle choosing velocity v as its control decision then moves position x according to the ODE $x' = v$. After some time, the controller can revise its decision, e. g., modeled by executing the controller and the plant sequentially in a loop, when control decisions have instantaneous effect as commonly assumed in hybrid systems models.

Model 1 Obstacle Component

$$ctrl_r \equiv dx_o := *;\ dy_o := *;\ ?\mathsf{SAFE_{obs}} \tag{1}$$

$$plant_r \equiv t^- := t;\ (\underbrace{t' = 1}_{\frac{\partial t(\tau)}{\partial \tau}=1},\ \underbrace{x_o' = dx_o}_{\frac{\partial x_o(\tau)}{\partial \tau}=dx_o},\ \underbrace{y_o' = dy_o}_{\frac{\partial y_o(\tau)}{\partial \tau}=dy_o}\ \&\ t - t^- \leq \varepsilon) \tag{2}$$

When composing components, their controllers and plants are composed. As time passes simultaneously in all components, the plants must be composed truly in parallel. We assume that control statements take instantaneous effect, but their execution order is crucial. As the actual order in which the component controllers are executed is unknown, we choose the order of controller executions non-deterministically, e. g., for two component controllers: execute $ctrl_1$ followed by $ctrl_2$ (; represents sequential composition), or $ctrl_2$ followed by $ctrl_1$, i. e., $ctrl_1; ctrl_2 \cup ctrl_2; ctrl_1$ ($\cup$ represents non-deterministic choice). A system expert might furthermore have additional insights into the system and might thus know in which order the components are actually executed. The original monolithic models of all three case studies were modeled with specific controller orderings.

Throughout the models, we use x^- to refer to the previous value of x (e. g., the value of x from the previous controller execution, or a previously sensed value). We use $\hat{x}$ to refer to component inputs, such as sensor measurements or values transmitted to the component.

In all three case studies, the overall safety property of the monolithic system is known. The safety properties for the components may vary with a specific component-based verification approach. Here, we present *ctrl* and *plant* for each component, together with a description of potential assumptions. Furthermore, we suggest possible local component safety properties for loss-less and instantaneous communication.

3.1 Robot Collision Avoidance

Inspired by [4], in the Robot Collision Avoidance case study, a robot moves on a plane along curved trajectories. It must avoid (active) collision with a moving obstacle.

System Contract. The overall safety property ensures that the robot's position never coincides with the obstacle's position while the robot moves, i. e., $s_r > 0 \rightarrow \| (x_r, y_r) - (x_o, y_o) \| > 0$. The responsibility of the robot is to stop before collision occurs. It assumes that the obstacle obeys a speed limit.

Obstacle. The obstacle (cf. Model 1), e. g., a stationary wall or a moving person, is deliberately liberal to allow for many concrete implementations. For the sake of simplicity and following [4], the obstacle moves arbitrarily in straight lines by choosing velocity and direction (dx_o, dy_o), such that the property $\mathsf{SAFE_{obs}}$ is satisfied, cf. (1). The plant stores the plant start time, measures duration $t' = 1$ and moves the obstacle according to velocity and direction $(x_o' = dx_o, y_o' = dy_o)$, but no longer than $t - t^- \leq \varepsilon$ time to ensure that the controller can react at least once every ε time, cf. (2).

The obstacle's maximum speed is assumed non-negative $S \geq 0$ and it has positive maximum plant duration $\varepsilon > 0$. The property $\mathsf{SAFE_{obs}}$ is used to check if the chosen direction is safe; it scales with the type of contract as follows:

Model 2 Robot Component

$$
\begin{aligned}
ctrl_r \equiv\ & \{\ a_r := -B\ \} && (3)\\
& \cup \{\ ?s_r = 0;\ a_r := 0;\ w := 0\ \} && (4)\\
& \cup \{\ a_r := *;\ ?-B \le a_r \le A; && (5)\\
& \quad k := *;\ w := *;\ ?s_r \cdot k = w;\ ?\mathsf{SAFE_{rob}}\ \} && (6)\\
plant_r \equiv\ & t^- := t;\ (t' = 1, x_r' = s_r \cdot dx_r, y_r' = s_r \cdot dy_r, && (7)\\
& \quad dx_r' = -w \cdot dy_r, dy_r' = w \cdot dx_r, s_r' = a_r, w' = a_r \cdot k\ \&\ t - t^- \le \varepsilon \wedge s_r \ge 0) && (8)
\end{aligned}
$$

Global contract. The obstacle stays inside a fixed region bounded by X_o and Y_o, i. e., $x_o \le X_o \wedge y_o \le Y_o$. The property $\mathsf{SAFE_{obs}}$ ensures that the obstacle will not leave the designated region until the next controller run, i. e. $\mathsf{SAFE_{obs}} \equiv x_o + dx_o\varepsilon \le X_o \wedge y_o + dy_o\varepsilon \le Y_o$.

Delay contract. The obstacle guarantees to stay inside a circle around its previous position, with radius according to the duration of motion. In other words, it guarantees to obey the maximum speed S. The resulting property is $\|(x_o, y_o) - (x_o^-, y_o^-)\| \le S \cdot (t - t^-)$.

In this case, the property $\mathsf{SAFE_{obs}}$ ensures that the obstacle's velocity never exceeds the maximum velocity, i. e., $\mathsf{SAFE_{obs}} \equiv dx_o^2 + dy_o^2 \le S^2$.

Robot. The robot (cf. Model 2) is responsible for collision avoidance. It can either brake on the current curve ($a_r := *$, cf. (3)), remain stopped $a_r := 0$ without steering $w := 0$ if it is stopped already (cf. (4)) or choose a new safe curve (cf. (5)–(6)). The model allows non-deterministic choice between these control decisions, but specifies conditions that prevent certain choices in certain situations. When selecting a new curve, the controller first chooses any acceleration $a_r := *$ between maximum braking and maximum acceleration $?-B \le a_r \le A$, cf. (5). Then it chooses any steering $k := *$ and the appropriate rotational velocity $w := *$ that fits to steering and speed as ensured by the test $?s_r \cdot k = w$. It only executes these choices when $\mathsf{SAFE_{rob}}$ indicates that motion is safe until the next control cycle, cf. (6). In (7), first the current time is stored. Then, the plant (7)–(8) models curved trajectories with a non-linear differential equation $dx_r' = -w \cdot dy_r, dy_r' = w \cdot dx_r$, so the direction vector rotates. Position follows from speed and direction $x_r' = s_r \cdot dx_r, y_r' = s_r \cdot dy_r$. Speed in turn follows from acceleration $s_r' = a_r$, with rotational speed $w' = a_r \cdot k$ changing to preserve the motion equation $s_r \cdot k = w$. The plant duration is restricted by ε to guarantee controller runs at least every ε time units. Furthermore, negative speeds are not allowed.

Again, $\mathsf{SAFE_{rob}}$ scales with the contract:

Global contract. The obstacle is assumed to start inside a fixed area ($\hat{x}_o \le X_o \wedge \hat{y}_o \le Y_o$), while the robot starts outside this area ($x_r > X_o \vee y_r > Y_o$), and stays outside, even when accelerating: $\mathsf{SAFE_{rob}} \equiv \max\big((x_r - X_o), (y_r - Y_o)\big) > \frac{s_r^2}{2 \cdot B} + \left(\frac{A}{B} + 1\right) \cdot \left(\frac{A}{2} \cdot \varepsilon^2 + \varepsilon \cdot s_r\right)$.

Delay contract. The obstacle must be far enough away initially to guarantee that the robot can stop from its initial velocity before a collision occurs, i. e., $\|(x_r, y_r) - (\hat{x}_o, \hat{y}_o)\| > \frac{s_r^2}{2 \cdot B} + S \cdot \frac{s_r}{B}$. Likewise, $\mathsf{SAFE_{rob}}$ only allows acceleration if the distance to the obstacle is safe: $\mathsf{SAFE_{rob}} \equiv \|(x_r, y_r) - (\hat{x}_o, \hat{y}_o)\| > \frac{s_r^2}{2 \cdot B} + S \cdot \frac{s_r}{B} + \left(\frac{A}{B} + 1\right) \cdot \left(\frac{A}{2} \cdot \varepsilon^2 + \varepsilon \cdot (s_r + S)\right)$.

Model 3 RBC Component

$$
\begin{aligned}
ctrl_{rbc} \equiv \quad & \{\mathrm{s} := \mathsf{brake}\} && (9)\\
\cup\, & \{\mathrm{s} := \mathsf{drive};\ m := *;\ d := *;\ v_{des} := *; && (10)\\
& \quad ?d \geq 0 \land v_{des} \geq 0 \land (d^-)^2 - d^2 \leq 2 \cdot B \cdot \big(m - m^-\big)\} && (11)\\
plant_{rbc} \equiv\ & \mathtt{skip} && (12)
\end{aligned}
$$

Further variations on $\mathsf{SAFE}_{\mathsf{rob}}$ to account for sensor uncertainty, together with arithmetic simplifications (e. g., over-approximate Euclidian distance $\|(x_r, y_r) - (\hat{x}_o, \hat{y}_o)\|$ with infinity norm $\|(x_r, y_r) - (\hat{x}_o, \hat{y}_o)\|_\infty$) are listed in [4].

The robot assumes positive maximum breaking power $B > 0$, non-negative maximum acceleration $A \geq 0$, non-negative maximum speed $S \geq 0$ and positive maximum plant duration $\varepsilon > 0$. Its direction vector is normalized, i. e., $dx_r^2 + dy_r^2 = 1$ and initial speed is assumed to be non-negative, i. e., $s_r \geq 0$.

Since in this example the obstacle is allowed to be malicious, it is the robot's responsibility to ensure system safety and hence the local safety property of the robot resembles the desired global system safety property, i. e., $s_r > 0 \to \|(x_r, y_r) - (\hat{x}_o, \hat{y}_o)\| > 0$.

3.2 European Train Control System

Inspired by [9], the European Train Control System (ETCS) case study comprises a radio-block controller (RBC, cf. Model 3) that sends commands to a train (cf. Model 4).

System Contract. The RBC communicates a track permission to the train: inside the permitted area m, the train should not exceed desired speed v_{des}; when exiting the area, the train must obey a strict speed limit d. The RBC may also request emergency braking. The system guarantees that the train moves at most with velocity $v \leq d$ when its position z is outside the permitted area, i. e., $z \geq m \to v \leq d$.

Radio-Block Controller RBC. The RBC either requests emergency braking ($\mathrm{s} := \mathsf{brake}$) (9) or permits driving ($\mathrm{s} := \mathsf{drive}$) in (10). Together with the driving permission, the RBC chooses the end of the track permission m, the speed limit d for exiting and the speed advice v_{des} for driving inside the permitted area. The maximum exit speed d and the speed advice v_{des} are non-negative and chosen such that the train can follow, i. e., exit m with at most speed d when applying brakes B, cf. (11). The RBC is a controller without plant.

The RBC initializes the constants $\mathsf{brake} = 1$ and $\mathsf{drive} = 0$. Initially, the state is assumed to be $\mathrm{s} = \mathsf{drive}$ with initial desired speed $v_{des} = 0$. The maximum exit speed is assumed to be non-negative $d >= 0$, and the maximum braking power is assumed to be positive $B > 0$.

The contract of the RBC guarantees that it either requests emergency braking with unchanged maximum exit speed d and unchanged track permission area m, or the state is drive and the new values for maximum exit speed and movement authority are chosen such that the train can obey the exit speed, i. e., $(\mathrm{s} = \mathsf{brake} \land m^- = m \land d^- = d) \lor \big(\mathrm{s} = \mathsf{drive} \land d \geq 0 \land v_{des} \geq 0 \land (d^-)^2 - d^2 \leq 2 \cdot B \cdot (m - m^-)\big)$. This property requires a change contract as it includes previous values of several variables.

Model 4 Train Component

$$\begin{aligned}
ctrl_{tr} \equiv \{ \quad & \{\ ?v \leq \hat{v}_{des};\ a := *;\ ? - B \leq a \leq A\ \} && (13)\\
\cup & \{\ ?v \geq \hat{v}_{des};\ a := *;\ ? - B \leq a \leq 0\ \}\ \}; && (14)\\
& \mathrm{SB} := \frac{v^2 - \hat{d}^2}{2 \cdot B} + \left(\frac{A}{B} + 1\right) \cdot \left(\frac{A}{2} \cdot \varepsilon^2 + \varepsilon \cdot v\right); && (15)\\
& \{ \quad \{\ ?\hat{m} - z \leq \mathrm{SB} \vee \hat{\mathrm{s}} = \mathsf{brake};\ a := -B\ \} && (16)\\
& \cup \{\ ?\neg\,(\hat{m} - z \leq \mathrm{SB} \vee \hat{\mathrm{s}} = \mathsf{brake})\ \}\ \} && (17)\\
plant_{tr} \equiv\ & t^- := t;\ (t' = 1, z' = v, v' = a\ \&\ t - t^- \leq \varepsilon \wedge v \geq 0) && (18)
\end{aligned}$$

Train. In normal operation, the train performs velocity control to follow v_{des}. The train engages brakes in case it receives an emergency braking request or when it is about to exit the permitted track area.

If the desired maximum speed v_{des} is not yet exceeded ($v \leq \hat{v}_{des}$), the train chooses any acceleration $a := *$ in the braking and acceleration limits $-B \leq a \leq A$, cf. (13). If the train is too fast ($?v \geq \hat{v}_{des}$), it may slow down ($a := *;\ ? - B \leq a \leq 0$), cf. (14). Then, the distance SB required for the train to obey the maximum exit speed d is calculated in (15). In (16), the train engages brakes ($a := -B$) if required by this distance or requested from the RBC ($?\hat{m} - z \leq \mathrm{SB} \vee \hat{\mathrm{s}} = \mathsf{brake}$). Otherwise, the chosen acceleration is executed, cf. (17). In the plant, the train moves according to the chosen acceleration, restricted by the maximum plant duration ε. Furthermore, velocity must remain non-negative, cf (18).

Constants agree with the RBC, so $\mathsf{brake} = 1$, $\mathsf{drive} = 0$, $\hat{\mathrm{s}} = \mathsf{drive}$, the train has working brakes $B > 0$ and a functional engine $A \geq 0$, and maximum control cycle duration $\varepsilon > 0$. Initially the train is assumed to be stopped inside the permitted area $z \leq \hat{m}$, with desired speed $\hat{v}_{des} = 0$, initial velocity $v = 0$, and maximum exiting speed $\hat{d} = 0$.

The train assumes to only receive commands that are compliant with the RBC contract. It guarantees to exit the permitted track area no faster than allowed, i. e., $z \geq \hat{m} \rightarrow v \leq \hat{d}$.

3.3 Adaptive Cruise Control

Based on [3], a follower car (cf. Model 6) uses adaptive cruise control on a highway to drive safely behind a leader car (cf. Model 5), neither colliding with nor overtaking the leader.

System Contract. The system contract guarantees that the follower remains behind the leader, i. e., $x_f < x_l$. The leader is driving forward, applying at most braking power B. The follower reads the leader's position and velocity and guarantees to stay safely behind.

Leader Car. The leader car (cf. Model 5) drives forward on a road modeled as a one-dimensional straight line. The leader chooses any acceleration $a_l := *$ between the maximum braking $-B$ and maximum acceleration A, cf. (19). In the plant, the leader car moves according to the chosen acceleration, restricted by ε and at most as long as its velocity remains non-negative, cf. (20).

Model 5 Leader Component

$$ctrl_l \equiv a_l := *; ?-B \leq a_l \leq A \tag{19}$$

$$plant_l \equiv t^- := t;\ (t' = 1, x_l' = v_l, v_l' = a_l \ \&\ t - t^- \leq \varepsilon \wedge v_l \geq 0) \tag{20}$$

The leader assumes positive maximum braking power $B > 0$, non-negative maximum acceleration $A \geq 0$, positive maximum plant duration $\varepsilon > 0$, and a non-negative initial speed $v_l \geq 0$. The leader's contract guarantees non-negative velocity $v_l \geq 0$ with a rate of change according to the braking and acceleration bounds, i. e., $-B\cdot(t-t^-) \leq v_l - v_l^- \leq A\cdot(t-t^-)$. In turn, position is guaranteed to change at most with the mean velocity $\frac{v_l+v_l^-}{2}$, so $x_l - x_l^- \leq \frac{v_l+v_l^-}{2}\cdot(t-t^-)$.

Follower Car. The follower car (cf. Model 6) is responsible for staying safely behind the leader. The follower chooses non-deterministically to either brake, remain stopped or accelerate, similar to the robot in Section 3.1. Braking is always allowed, cf. (21). If the car is stopped it may stay stopped, cf. (22). If the follower can accelerate safely, i. e., without collision with the leader until the next control cycle (cf. (23)), it chooses any acceleration in $-B \leq a_f \leq A$, cf. (24). In the plant, the follower car moves according to the chosen acceleration, restricted by ε and at most as long as its velocity remains non-negative, cf. (25).

The follower may assume positive maximum braking power $B > 0$, non-negative maximum acceleration $A \geq 0$, positive maximum plant duration $\varepsilon > 0$, and a positive initial speed $v_f \geq 0$. The follower is assumed to start behind the leader ($x_f < \hat{x}_l$) and the leader is assumed to not move backwards ($0 \leq \hat{v}_l$). Additionally, the follower must be far enough behind the leader, such that it can stop to avoid collision with the leader, i. e., $\frac{x_f+v_f^2}{2\cdot B} < \frac{\hat{x}_l+\hat{v}_l^2}{2\cdot B}$. If the leader complies with its contract, the follower guarantees to stay behind the leader, i. e., $x_f < \hat{x}_l$.

Model 6 Follower Component

$$\begin{aligned}
ctrl_f \equiv\ & \{\ a_f := -B\ \} && (21)\\
& \cup \{\ ?v_f = 0;\ a_f := 0\ \} && (22)\\
& \cup \{\ ?\frac{x_f + v_f^2}{2\cdot B} + \left(\frac{A}{B}+1\right)\cdot\left(\frac{A}{2}\cdot\varepsilon^2 + \varepsilon\cdot v_f\right) < \frac{\hat{x}_l + \hat{v}_l^2}{2\cdot B}; && (23)\\
& \quad a_f := *;\ ?-B \leq a_f \leq A\ \} && (24)\\
plant_f \equiv\ & t^- := t;\ (t' = 1, x_f' = v_f, v_f' = a_f\ \&\ t - t^- \leq \varepsilon \wedge v_f \geq 0) && (25)
\end{aligned}$$

4 Experiments

The KeYmaera X models and contracts described in this paper are attached to the benchmark and organized into one folder per case study. Each case study folder contains two component models and a monolithic model resulting from applying the parallel composition of [6]. The robot collision avoidance case study in the `robix` folder contains the robot component `robot.kyx`, the obstacle component `obstacle.kyx` and the monolithic model `sys-robix.kyx`.

The European train control system case study in the `etcs` folder contains the train component `train.kyx`, the RBC component `rbc.kyx` and the monolithic model `sys-etcs.kyx`. The adaptive cruise control case study in the `llc` folder contains the leader component `leader.kyx`, the follower component `follower.kyx` and the monolithic model `sys-llc.kyx`.

We applied component-based hybrid systems theorem proving [5, 6] to the components of all three case studies and verified their correctness when using loss-less and instantaneous communication between the components, which can serve as a benchmark baseline together with the monolithic case studies [3, 4, 9].

As a composition operation, the *plant*s were joined in parallel, whereas the controllers were executed sequentially in arbitrary non-deterministic order. For composition, [6] requires connecting inputs of one component to outputs of another component (e. g., the obstacle position was connected to the obstacle sensor in the robot). If the connected components are compatible, [6, Theorem 1] guarantees that composing two provably contract-compliant components result in a joined system property. Components are compatible, if the output guarantees of one component are at least as strict as the respective input assumptions of the connected second component. Detailed results, such as proof durations and proof script size, of our experiments with all three examples can be found in [6]. In summary, our results indicate that component-based verification can lead to performance improvements and smaller user-provided proof scripts.

References

[1] Goran Frehse, Zhi Han, and B. Krogh. Assume-guarantee reasoning for hybrid I/O-automata by over-approximation of continuous interaction. In *43rd IEEE Conference on Decision and Control, CDC*, volume 1, pages 479–484 Vol.1, 2004.

[2] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In Amy P. Felty and Aart Middeldorp, editors, *25th International Conference on Automated Deduction, Proceedings*, volume 9195 of *LNCS*, pages 527–538. Springer, 2015.

[3] Sarah M. Loos, André Platzer, and Ligia Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In Michael J. Butler and Wolfram Schulte, editors, *17th International Symposium on Formal Methods*, volume 6664 of *LNCS*, pages 42–56. Springer, 2011.

[4] Stefan Mitsch, Khalil Ghorbal, and André Platzer. On provably safe obstacle avoidance for autonomous robotic ground vehicles. In Paul Newman, Dieter Fox, and David Hsu, editors, *Robotics: Science and Systems IX*, 2013.

[5] Andreas Müller, Stefan Mitsch, Werner Retschitzegger, Wieland Schwinger, and André Platzer. A component-based approach to hybrid systems safety verification. In Erika Ábrahám and Marieke Huisman, editors, *Integrated Formal Methods - 12th International Conference, Proceedings*, volume 9681 of *LNCS*, pages 441–456. Springer, 2016.

[6] Andreas Müller, Stefan Mitsch, Werner Retschitzegger, Wieland Schwinger, and André Platzer. Change and delay contracts for hybrid system component verification. In *20th International Conference on Fundamental Approaches to Software Engineering (FASE)*, 2017 (to appear).

[7] André Platzer. The complete proof theory of hybrid systems. In *LICS*, pages 541–550. IEEE, 2012.

[8] André Platzer. A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reas.*, pages 1–47, 2016.

[9] André Platzer and Jan-David Quesel. European train control system: A case study in formal verification. In Karin K. Breitman and Ana Cavalcanti, editors, *Formal Methods and Software Engineering*, volume 5885 of *LNCS*, pages 246–265. Springer, 2009.