Parallel Composition and Modular Verification of Computer Controlled Systems in Differential Dynamic Logic*

Simon Lunel[1,2], Stefan Mitsch[3], Benoit Boyer[1], and Jean-Pierre Talpin[2]

[1] Mitsubishi Electric R&D Centre Europe, 1 allée de Beaulieu, CS 10806, 35708 Rennes CEDEX 7, FRANCE b.boyer@fr.merce.mee.com
[2] Inria, Centre de recherche Rennes - Bretagne - Atlantique, Campus universitaire de Beaulieu, 35042 Rennes Cedex, FRANCE jean-pierre.talpin@inria.fr
[3] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA smitsch@cs.cmu.edu

Abstract. Computer-Controlled Systems (CCS) are a subclass of hybrid systems where the periodic relation of control components to time is paramount. Since they additionally are at the heart of many safety-critical devices, it is of primary importance to correctly model such systems and to ensure they function correctly according to safety requirements. Differential dynamic logic $d\mathcal{L}$ is a powerful logic to model hybrid systems and to prove their correctness. We contribute a component-based modeling and reasoning framework to $d\mathcal{L}$ that separates models into components with timing guarantees, such as reactivity of controllers and controllability of continuous dynamics. Components operate in parallel, with coarse-grained interleaving, periodic execution and communication. We present techniques to automate system safety proofs from isolated, modular, and possibly mechanized proofs of component properties parameterized with timing characteristics.

1 Introduction

A computer-controlled system (CCS) is a hybrid system with discrete hardware-software components that control a specific physical phenomenon, *e.g.* the water level of a tank in a water-recycling plant. CCSs are widely used in industry to monitor time-critical and safety-critical processes. While CCS defines a large class of hybrid systems, systems mixing physical phenomena and natural discrete interactions (*e.g.* a bouncing-ball) are neither CCSs nor the focus of this work, although most could easily be given verification models in $d\mathcal{L}$. Tools to model, verify, and design CCSs need to capture mixed discrete and continuous dynamics, as well as mixed logical, discretized real-time and continuous time in, resp., computer programs, electronics, and physics models.

* This material is based upon work supported by the United States Air Force and DARPA under Contract No. FA8750-18-C-0092.

CCSs are difficult to model since they subsume the problem of designing a software controller and its real-timed hardware. Our aim is to develop a *component-based approach* to engineer such systems in a modular manner while accounting for time domain boundaries across components. In component-based design, a system is constructed from smaller elements that are modeled and individually verified, then assembled and checked for consistency to form larger components and subsystems. The CCS components typically execute in parallel, and concurrency must be accounted for in the envisioned verification framework.

In this paper, we contribute a component-based verification technique that aims at the definition of a bottom-up and modular verification methodology through a *correct-by-construction* system design methodology, in which component *contracts* formalize the domain, timing, and invariants of components. The proof of a system model is built by assembling the contracts of its components through formally defined composition mechanisms. Contracts-based approaches have been successfully implemented for several paradigms such as programming languages [16] or automata [2], because contracts are very efficient to make proofs easier scalable. Following the component-based design widely used for CCS, contracts provide a natural way to get modularity and abstraction in proofs.

To meet the time-criticality requirements of CCSs, we start from earlier compositionality results in $d\mathcal{L}$ [7] to elaborate a timed model of parallel composition as the foundation of our modeling and verification framework. In Sec. 3, we detail modeling and verification in our framework on a simple system where only one reactive controller monitors a plant. In Sec. 4, we generalize it to systems where multiple controllers monitor multiple parallel plants; we show how to compose: multiple reactive controllers into a component called Multi-Choice Reactive Controllers (**MRCtrl**, see Sec. 4.1); multiple controllable plants together (see Sec. 4.2); **MRCtrl** with controllable plants to form Multi Computer-Controlled Systems (**MCCS**, see Sec. 4.3); and finally how **MCCS** compose (see Sec. 4.4).

2 Differential Dynamic Logic

This section briefly recalls *differential dynamic logic* ($d\mathcal{L}$ [13]) and its proof system, which is implemented in the theorem prover KeYmaera X [3].

In $d\mathcal{L}$, hybrid programs are used as a programming language for expressing the combined discrete and continuous dynamics of hybrid systems (the programs operate over mathematical reals). The syntax and semantics of hybrid programs is summarized in Tab. 1. The set of reachable states from state ν by hybrid program α is noted $\rho_\nu(\alpha)$, and $[\![x]\!]_\nu$ denotes the value of x at state ν. Hybrid programs include discrete assignment $x := \theta$ and tests $?\phi$, as well as combinators for non-deterministic choice ($\alpha \cup \beta$), sequential composition ($\alpha; \beta$), and non-deterministic repetition (α^*). The notation $\{x' = \theta \ \& \ H\}$ denotes an ordinary differential equation (ODE) system (derivatives with respect to time) of the form $x'_1 = \theta_1, \ldots, x'_n = \theta_n$ within evolution domain H. For example, the ODE $\{t' = 1 \ \& \ t \geqslant 0\}$ describes that variable t evolves with constant slope 1, where $t \geqslant 0$ discards any negative values. Formulas of $d\mathcal{L}$ formalize properties, Def. 1.

Table 1. Syntax and semantics of hybrid programs

Program	Semantics
$?\phi$	Test whether formula ϕ is true, abort if false: $\rho_\nu(?\varphi) = \{\nu \mid \nu \models \varphi\}$
$x := \theta$	Assign value of term θ to variable x: $\rho_\nu(x := \theta) = \{\omega \mid \omega = \nu \text{ except that } [\![x]\!]_\omega = [\![\theta]\!]_\nu\}$
$\{x' = \theta \ \& \ H\}$	Evolve ODE $x' = \theta$ for any time $t \geqslant 0$ with evolution domain constraint H true throughout: $\rho_\nu(\{x' = \theta \ \& \ H\}) = \{f(r) \mid f(0) = \nu$ and for any duration $r \geqslant 0$ $f(r) \models x' = \theta$ and $f(r) \models H\}$
$\alpha; \beta$	Run α followed by β on resulting state(s): $\rho_\nu(\alpha; \beta) = \bigcup_{\omega \in \rho_\nu(\alpha)} \rho_\omega(\beta)$
$\alpha \cup \beta$	Run either α or β non-deterministically: $\rho_\nu(\alpha \cup \beta) = \rho_\nu(\alpha) \cup \rho_\nu(\beta)$
α^*	Repeat α n times, for any $n \in \mathbb{N}$: $\rho(\alpha^*) = \bigcup_{n \in \mathbb{N}} \rho(\alpha^n)$ with $\alpha^0 = ?\top$ and $\alpha^{n+1} = \alpha^n; \alpha$

Definition 1 ($d\mathcal{L}$ formulas). *The formulas ϕ, ψ of $d\mathcal{L}$ relevant in this paper consist of the following operators:*

$$\phi, \psi ::= \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \neg\phi \mid \theta_1 \sim \theta_2 \mid \forall x \ \phi \mid \exists x \ \phi \mid [\alpha]\phi$$

Connectives $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \rightarrow \psi$,$\neg\phi$, $\forall x \ \phi$, and $\exists x \ \phi$ are according to classical first-order logic. Formula $\theta_1 \sim \theta_2$ are any comparison operator $\sim \in \{\leqslant, <, =, \neq, >, \geqslant\}$ and θ_i are real-valued terms in operators $\{+, -, \cdot, /\}$. The modal operator $[\alpha]\phi$ is true iff ϕ holds in all states reachable by program α.

The notion of *free and bound variables* is defined in the static semantics of $d\mathcal{L}$ [13] and useful to characterize the interaction of a program α with its context. It is computed from the syntactic structure of programs: the bound variables $\mathrm{BV}(\alpha)$ can be updated by assignments (*e.g.* $x := 10$) or ODEs (*e.g.* $x' = 3$) in α, whereas free variables $\mathrm{FV}(\alpha)$ are those that the program depends on. For example, in program $\alpha \equiv (v := a \cup v := 2); \{x' = v \ \& \ x \leqslant 5\}$ the free variables are $\mathrm{FV}(\alpha) = \{a, x\}$ (the variable v is not free, because it is bound on all paths of α and so the result of α does not depend on the initial value of v; even though also modified, variable x is free because the result of α depends on the initial value of x) and the bound variables are $\mathrm{BV}(\alpha) = \{v, x\}$ (variable a is not bound because it is not modified anywhere in the program). We use $\mathrm{V}(\alpha)$ to denote $\mathrm{BV}(\alpha) \cup \mathrm{FV}(\alpha)$.

In [7], a component model $C_i = (\mathbf{disc}_i \cup \mathbf{cont}_i)^*$ is evaluated as the non-deterministic interleaving of its discrete specifications $\mathbf{disc}_i$ and ODE $\mathbf{cont}_i = \{x_i' = \theta_i \ \& \ H_i\}$. For $i \in \{1, 2\}$, the parallel composition of C_1 and C_2 builds a component of the same structure, *i.e.* the dicrete parts $\mathbf{disc}_1$ and $\mathbf{disc}_2$ are non-deterministically interleaved within the evolution of the ODE obtained from the mathematical composition of $\mathbf{cont}_1$ and $\mathbf{cont}_2$. In $d\mathcal{L}$, it is defined by

$$C_1 \otimes C_2 = (\mathbf{disc}_1 \cup \mathbf{disc}_2 \cup \{x_1' = \theta_1, x_2' = \theta_2 \ \& \ H_1 \wedge H_2\})^* \tag{1}$$

The logic $d\mathcal{L}$ further enjoys a proof calculus [12,13,14] based on uniform substitution from axioms. Its base axioms are those of a classical first-order sequent calculus, augmented with syntactical deconstruction of hybrid programs α for goals of the form $[\alpha]\phi$ and for iteration and ODEs [15].

3 Computer-Controlled Systems

We present a component-based approach to model and verify Computer Controlled Systems (CCS) based on the parallel composition pattern proposed in [7]. We construct the proof of a CCS from the isolated sub-proofs of its components by *syntactically* decomposing the CCS using the axioms of $d\mathcal{L}$, so that the theorems presented here can be implemented as tactics in the theorem prover KeYmaera X. This enables automation to reduce the proof complexity of analyzing a CCS to that of modularly analyzing its components.

In this section, we introduce the necessary concepts to adapt the framework of [7] to systematically model CCSs modularly. We achieve modularity by limiting the ways in which the free and bound variables of different components may overlap, and by taking into account the timing constraints of CCSs. The idea is to analyze the *controllability* of the plant, *i.e.* the period of time it can evolve safely without intervention of a controller, and the *reactivity* of the controller, *i.e.* the execution period of the controller. These concepts satisfy the associativity property of parallel composition and the ability to retain contracts from [7].

3.1 Modeling CCS

A CCS is classically composed of a *controller* and a *plant*. The controller measures the state of the plant through sensing and regulates the behavior of the plant through actuation. For example, the controller in the water tank regulates the water level by opening or closing a faucet. The key trait of a CCS is the periodic execution of the controller to regulate the plant. We associate periodic values δ and Δ with the controller and the plant, respectively: Control *reactivity* δ models the period in which control is guaranteed to happen. Plant *controllability* Δ models how long a plant can evolve safely without control intervention.

Time. To make timed reasoning available to any component, we use the ODE $\mathbf{Time}(t) \doteq \{t' = 1 \ \& \ t \geqslant 0\}$. The hence defined global variable t represents time passing with constant slope 1, initialized to 0.

Controller. The functional behavior of a controller is provided as a discrete program *ctrl* and the associated *reactivity* δ. The controller acts at least every δ units of time, see Def. 2.

Definition 2 (Reactive Controller). *A* reactive controller $\boldsymbol{RCtrl}(ctrl, \delta)$ *with* reactivity boundary δ *and fresh timestamp* τ *has the program shape*

$$\boldsymbol{RCtrl}(ctrl, \delta) \doteq \ (?t \leqslant \tau + \delta; \ ctrl; \ \tau := t)$$

Execution periodicity is ensured by a *fresh* variable τ time stamping $(\tau := t)$ the last execution of *ctrl*. The prefixing guard $?t \leqslant \tau + \delta$ forces *ctrl* to be executed within δ time since its last execution τ. This pattern models control frequency, since all runs not satisfying a test are aborted, see Sec. 2.

Example 1 (Water-level Controller). We consider the water level controller in a water plant[4]. When the level reaches a maximum (resp. minimum) threshold, we close the inlet faucet *fin* (resp. we open the inlet faucet). The resulting controller has the program shape $\mathbf{RCtrl}(wlctrl, \delta_{wlctrl})$ i.e. $(?t \leqslant \tau + \delta_{wlctrl};\ wlctrl;\ \tau := t)$, where $\delta_{wlctrl} = 0.05s$ ensures a control frequency of at least $20Hz$. The controller:

$$wlctrl \doteq wlm := wl; \big((?wlm \geqslant 6.5; fin := 0) \cup (?wlm \leqslant 3.5; fin := 1)\big)$$

measures the water level using $wlm := wl$ and then sets *fin* depending on whether the water level exceeds the minimum threshold 3.5 or the maximum threshold 6.5. This controller makes implicit assumptions on the maximum inflow and outflow of the water tank through the relation between its reactivity δ_{wlctrl} and the thresholds on wlm.

Plant. The functional behavior of the plant is provided as an ODE system $\{x' = \theta\ \&\ H\}$ with $t \notin \mathrm{V}(x' = \theta)$ and the *controllability bound* Δ. Controllability is implemented by adding the formula $t \leqslant \Delta$ to the evolution domain, see Def. 3.

Definition 3 (Controllable Plant). *A* controllable plant $\boldsymbol{CPlant}(\{x' = \theta\ \&\ H\}, \Delta)$ *with* controllability bound Δ *is a differential equation system of the shape* $\boldsymbol{CPlant}(\{x' = \theta\ \&\ H\}, \Delta) \doteq \{x' = \theta\ \&\ H \wedge t \leqslant \Delta\}$*, combined with time defined by* $\boldsymbol{Time}(t)$*.*

Example 2 (Water-level). The evolution of the water level wl in the tank is determined by the difference between the inlet flow *fin* and the outlet flow *fout*. The water level is always non-negative ($H \doteq wl \geqslant 0$), and so the controllable water level is the ODE with controllability $\Delta_{wl} = 0.2s$:

$$\{wl' = fin - fout, t' = 1\ \ \&\ \ t \geqslant 0 \wedge wl \geqslant 0 \wedge t \leqslant \Delta_{wl}\}$$

We compose the plant with the controller to a full system with repeated interaction between the plant and the controller.

Full system. The full system is obtained by applying parallel composition as defined in (1) to the plant and the controller, but with one important change: the formula $t \leqslant \Delta$ is replaced by the formula $t \leqslant \tau + \delta$ to ensure that the plant suspends when the controller is expected to run, see Def. 4.

Definition 4 (Computer-Controlled System). *A* computer-controlled system $\boldsymbol{CCS}$ *is a parallel composition of a reactive controller* $\boldsymbol{RCtrl}(ctrl, \delta)$ *and a controllable plant* $\boldsymbol{CPlant}(\{x' = \theta\ \&\ H\}, \Delta)$ *with* $\delta \leqslant \Delta$ *and the resulting hybrid program shape, assuming* $\boldsymbol{Time}(t)$ *and, initially,* $\tau = t$*:*

$$\boldsymbol{CCS} \doteq \big(\{x' = \theta\ \ \&\ \ H \wedge \underbrace{t \leqslant \tau + \delta}_{\delta \leqslant \Delta}\} \cup \boldsymbol{RCtrl}(ctrl, \delta)\big)^*$$

[4] Adapted from `http://symbolaris.com/info/KeYmaera-guide.html#watertank`

Execution between the controller and the plant switches based on the variable τ. At the beginning of each loop iteration, we have $t \geqslant \tau$. The difference $t - \tau$ grows according to the evolution of time until the point $t - \tau = \delta$. At the latest, then, the controller must act before the plant can continue. Safety requires $\delta \leqslant \Delta$, *i.e.* the reactivity of the controller is at most the controllability of the plant. Otherwise, there may be runs of the whole system where the controller executes too late for the plant to stay safe.

Example 3 (Water-tank). We compose the water level with its controller to obtain the water tank system with the following behavior:

$$\left(\begin{array}{l} \{wl' = fin - fout, t' = 1 \ \ \& \ \ t \geqslant 0 \wedge wl \geqslant 0 \wedge t \leqslant \tau + \delta_{wlctrl}\} \\ \cup \ (?t \leqslant \tau + \delta_{wlctrl}; wlctrl; \tau := t) \end{array}\right)^*$$

The composition is possible because the reactivity of the controller ($\delta_{wlctrl} = 0.05s$) does not exceed the controllability of the plant ($\Delta_{wl} = 0.2s$).

3.2 Modular Verification of a CCS

Based on the modular modeling capabilities offered by the concepts of Sec. 3.1 and through [7, Thm. 2], we provide techniques to verify the safety of a complete system from safety proofs of its components (which can be reactive controllers, controllable plants, or subsystems built from those following the computer-controlled systems composition). The proofs of our theorems are syntactic using the axioms of $d\mathcal{L}$ (as opposed to the semantic proofs in [7]) and are, thus, implementable as tactics in the theorem prover KeYmaera X [3].

Environment. A description of the global system environment $\mathcal{E}$ characterizing constants (either as exact values or through their relevant characteristics) is necessary. We require that $\mathrm{FV}(\mathcal{E}) \cap \mathrm{BV}(\alpha) = \varnothing$ for all system components α to ensure that the environment variables are constants: these constants are not controlled, but can be read by all components. (*e.g.* , gravity constant g).

Example 4 (Water tank environment). In the water tank example, the environment $\mathcal{E}_{wt} \doteq fout = 0.75 \wedge \delta_{wlctrl} = 0.05 \wedge \Delta_{wl} = 0.2$ is the outlet flow *fout* of 0.75, plant controllability Δ_{wl} of $0.2s$, and controller reactivity δ_{wlctrl} of $0.05s$.

Contracts. A designer specifies the assumptions A_{ctrl} and guarantees G_{ctrl} of the controller as well as the assumption A_{plant} and guarantees G_{plant} of the plant. In order to be compositional, the guarantees of the controller must not refer to outputs of the plant and inversely ($\mathrm{FV}(G_{ctrl}) \cap \mathrm{BV}(plant) = \varnothing$ and $\mathrm{FV}(G_{plant}) \cap \mathrm{BV}(ctrl) = \varnothing$). A component α satisfies its contract (A_α, G_α) in environment $\mathcal{E}$ under starting conditions $Init_\alpha$ if formula $(\mathcal{E} \wedge A_\alpha \wedge Init_\alpha) \rightarrow [\alpha^*]G_\alpha$ is valid (*e.g.* , proved using the $d\mathcal{L}$ proof calculus). Unlike the environment $\mathcal{E}$, the initial conditions $Init_\alpha$ and assumptions A_α of a component α can mention assumptions about the state of other components.

Example 5 (Water tank contracts). The water-level controller assumes that the actual water level in the tank ranges over the interval $[3, 7]$ (as guaranteed by the tank), and itself guarantees to drain the tank when the measured water level approaches the upper threshold 6.5, and fill the tank when below the lower threshold 3.5. The tank contract assumes that the tank is instructed correctly to drain or fill, and then guarantees to keep the water level in the limits $[3, 7]$.

$$\begin{cases} A_{wlctrl} : G_{wl} \\ G_{wlctrl} : wlm \leqslant 3.5 \rightarrow fin = 1 \\ \qquad\quad 6.5 \leqslant wlm \rightarrow fin = 0 \\ \qquad\quad (3.5 \leqslant wlm \leqslant 6.5) \rightarrow (fin = 0 \vee fin = 1) \end{cases} \qquad \begin{cases} A_{wl} : G_{wlctrl} \\ G_{wl} : 3 \leqslant wl \leqslant 7 \end{cases}$$

These contracts assume that the measured water level is correct, *i.e.* it corresponds to the true water level in the tank, so $Init_{ctrl} \equiv wl = wlm$ and also $Init_{plant} \equiv wl = wlm$. As we compose the controller and plant components to a full system, where the plant evolves for some time between controller runs (and thus measurements), we will need to find a condition that describes the relationship between the true water level evolution and the measured water level.

Full system. The contract $(A_{ctrl} \wedge A_{plant}, G_{ctrl} \wedge G_{plant})$ for the full system is the conjunction of the assumptions and of the guarantees.

Composition invariant. In the full system, the controller and the plant will run in a quasi-parallel fashion, so time passes between controller runs and thus in turn between measurements of the true plant values. With a composition invariant J_{cmp} we describe the relationship between the true values of the plant and the measured values in the controller. The formula J_{cmp} is a composition invariant for two components α and β if the formulas $J_{\text{cmp}} \rightarrow [\alpha]J_{\text{cmp}}$ and $J_{\text{cmp}} \rightarrow [\beta]J_{\text{cmp}}$ are valid (components maintain the composition invariant), and $Init_\alpha \wedge Init_\beta \rightarrow J_{\text{cmp}}$ is valid (composition invariant is initially satisfied).

Each component is responsible for satisfying its own guarantees and can assume that others will satisfy its assumptions. We also require that other components do not interfere with a component's guarantees. This notion of non-interference ensures that contracts focus on the behavior of their own component (but nothing else), as intuitively expected.

Definition 5 (Non-interfering Controller and Plant). *A controller ctrl and plant $\{x' = \theta \& H\}$ are* non-interfering *if they do not influence the guarantees of the respective other component, so $FV(G_{ctrl}) \cap BV(\{x' = \theta \& H\}) = \varnothing$ and $FV(G_{plant}) \cap BV(ctrl) = \varnothing$, and if they do not share the same outputs, so $BV(ctrl) \cap BV(\{x' = \theta \& H\}) = \varnothing$.*

For composition it is important that contracts are compatible, meaning that they mutually satisfy their assumptions from their respective guarantees.

Definition 6 (Compatible Contracts). *Contracts (A_α, G_α) and (A_β, G_β) of components α and β with composition invariant J_{cmp} are* compatible *if the formulas $A_\alpha \rightarrow [\beta](G_\beta \wedge J_{cmp} \rightarrow A_\alpha)$ and $A_\beta \rightarrow [\alpha](G_\alpha \wedge J_{cmp} \rightarrow A_\beta)$ are valid.*

Theorem 1 (Composition of Controller and Plant). *Let* $\boldsymbol{RCtrl}(ctrl, \delta)$ *be a reactive controller satisfying its contract* (A_{ctrl}, G_{ctrl}) *and* $\boldsymbol{CPlant}(\{x' = \theta \ \& \ H\}, \Delta)$ *be a controllable plant satisfying its contract* (A_{plant}, G_{plant})*. Further let the components* $\boldsymbol{RCtrl}(ctrl, \delta)$ *and* $\boldsymbol{CPlant}(\{x' = \theta \ \& \ H\}, \Delta)$ *be non-interfering, the contracts* (A_{ctrl}, G_{ctrl}) *and* (A_{plant}, G_{plant}) *be compatible, and* J_{cmp} *be a composition invariant. Then, the parallel composition* $\boldsymbol{CCS}$ *is safe, i.e.,* $(\mathcal{E} \wedge A_{ctrl} \wedge Init_{ctrl} \wedge A_{plant} \wedge Init_{plant}) \rightarrow [\boldsymbol{CCS}](G_{ctrl} \wedge G_{plant})$ *is valid.*

Proof. Adapts [7, Thm. 2] to a syntactic $d\mathcal{L}$ proof with loop invariant $A_{ctrl} \wedge G_{ctrl} \wedge A_{plant} \wedge G_{plant} \wedge J_{\text{cmp}}$ with differential refinement to replace δ with Δ, see long version [8] for details. □

Example 6 (Water-tank contract). The controller and the water-level are non-interfering, their contracts compatible, and the controller is fast enough to keep the plant safe ($\delta_{wlctrl} \leqslant \Delta_{wl}$). We apply Thm. 1 with the composition invariant $J_{\text{cmp}} \doteq wl = (fin - fout)(t - \tau) + wlm$ to obtain that the composition is safe, *i.e.* formula $\mathcal{E} \wedge Init_{wl} \wedge Init_{wlctrl} \wedge A_{wl} \wedge A_{wlctrl} \rightarrow [\textbf{Water-tank}](G_{wl} \wedge G_{wlctrl})$ is valid. The composition invariant says how the true value wl deviates from the last measured value wlm according to the flow $fin - fout$ as time $t - \tau$ passes.

Outlook We adapted parallel composition of [7] to model and prove computer-controlled systems composed of two components, a reactive controller and a controllable plant. Next, we extend this concept to arbitrarily nested combinations of controllers and plants with a systematic integration of timed constraints.

4 Parallel Composition

We want to extend the integration of temporal considerations for every component in a timed framework. The previous section shows the importance of temporal considerations in CCS. Industrial systems combine CCS in parallel and it is necessary to have a framework to handle temporal properties.

In order to reason about parallel execution of control software sharing computation resources, models of different costs (controllability, performance, latency, etc) become important. For example, when two programs execute quasi-parallel on a single CPU core, their computation resources are shared and execution may mutually preempt. As a result, the worst-case execution times of the programs sum up to the total worst-case execution time of the composed system. This requires designing plants with sufficiently longer controllability periods, and controllers that react further in advance.

Based on the parallel composition pattern in [7] and the concepts of reactive controller and controllable plant introduced above, here we present parallel compositions of component hierarchies, including composition of multiple reactive controllers, multiple controllable plants, and mixed compositions. We retain the algebraic properties of [7], commutativity and associativity, and present theorems guaranteeing that the conjunction of contracts is preserved through composition.

Controllable plants are already hierarchically compositional per Def. 3. The particular structure to enclose control programs with temporal guards in reactive controllers, however, makes it necessary to extend the definition of reactive controller (Def. 2) to a *multi-choice reactive controller* that combines choices of each of its constituting atomic reactive controllers non-deterministically. We associate a *fresh variable* τ_i with each atomic reactive controller $ctrl_i$. It is used to specify the time stamp of the controller in an execution cycle.

Definition 7 (Multi-Choice Reactive Controller). *A* multi-choice reactive controller $\boldsymbol{MRCtrl}\left(\bigcup_{1\leqslant i\leqslant n} ctrl_i, \delta\right)$ *with* n *control choices and overall* reactivity bound δ *has the program shape*

$$\boldsymbol{MRCtrl}\left(\bigcup_{1\leqslant i\leqslant n} ctrl_i, \delta\right) \doteq \left(\bigcup_{1\leqslant i\leqslant n} \boldsymbol{RCtrl}(ctrl_i, \delta)\right) .$$

The parallel composition follows cases for purely discrete components, purely continuous components or a mix of both, which we detail in the subsections below. We illustrate each case with an example with two connected water-tanks, one where the inlet flow of one is the outlet flow of the other, with respective reactive controllers to ensure that they remain within a pre-defined range. The first controller actuates on the inlet flow of the first tank, whereas the second actuates on the outlet valve of the second tank.

4.1 Parallel Composition of Multi-Choice Reactive Controllers

We refine the parallel composition operator for multi-choice reactive controllers to consider the controllability and reactivity bounds Δ and δ of its components. By definition, the controllability bound of composed components α and β is always $\min(\Delta_\alpha, \Delta_\beta)$ of their individual bounds Δ_α, Δ_β. The reactivity bound depends on the physical architecture that composes α and β. It is overapproximated by a max+ cost function $\mathcal{C} : \mathbb{R}^2 \to \mathbb{R}$ such that $\mathcal{C}(\delta_\alpha, \delta_\beta) = \max(\delta_\alpha, \delta_\beta)$ if α and β have controllers running independently (e.g. two ECUs or PLCs), or else $\delta_\alpha + \delta_\beta$, if both controllers execute on one resource. Notice that such a definition is associative and commutative with respect to composition.

Modeling. We first define the parallel composition of discrete components, which are multi-choice reactive controllers $\mathbf{MRCtrl}(\bigcup_{1\leqslant i\leqslant n} ctrl_i, \delta)$. To the definition in [7], we add the cost model $\mathcal{C}$ to combine individual bounds δ as that of the composed system. The parallel composition is the non-deterministic choice between all control choices in multi-choice reactive controllers $\mathbf{MRCtrl}(\bigcup_{1\leqslant i\leqslant n_\alpha} \alpha_i, \delta_\alpha)$ and $\mathbf{MRCtrl}(\bigcup_{1\leqslant j\leqslant n_\beta} \beta_j, \delta_\beta)$, but with the individual δ_α and δ_β replaced by the cost model $\mathcal{C}(\delta_\alpha, \delta_\beta)$. Interleaving of controller executions occurs through embedding the non-deterministic choice in the loop of a full system, see Thm. 2.

Definition 8 (Parallel Composition of Multi-Choice Controllers). *Let* α *and* β *be multi-choice reactive controllers ofshapes* $\boldsymbol{MRCtrl}(\bigcup_{1\leqslant i\leqslant n_\alpha} \alpha_i, \delta_\alpha)$

and $\boldsymbol{MRCtrl}(\bigcup_{1\leqslant j\leqslant n_\beta}\beta_j,\delta_\beta)$*. Their parallel composition* $\alpha\otimes\beta$ *has shape:*

$$\boldsymbol{MRCtrl}\left(\bigcup_{1\leqslant i\leqslant n_\alpha}\alpha_i\cup\bigcup_{1\leqslant j\leqslant n_\beta}\beta_j,\mathcal{C}(\delta_\alpha,\delta_\beta)\right).$$

Example 7 (Composition of two water-level controllers). We compose two reactive water-level controllers $wlctrl_1$ (reactivity $\delta_{wlctrl_1}=0.05$s) and $wlctrl_2$ (reactivity $\delta_{wlctrl_2}=0.02$s) on one CPU. The multi-choice reactive controller resulting from cost model $\mathcal{C}(\delta_{wlctrl_1},\delta_{wlctrl_2})=\delta_{wlctrl_1}+\delta_{wlctrl_2}$ is:

$$\begin{aligned}&\mathbf{MRCtrl}\,(wlctrl_1\cup wlctrl_2,\delta_{wlctrl_1}+\delta_{wlctrl_2}\ )\\ =\ &\mathbf{RCtrl}(wlctrl_1,\delta_{wlctrl_1}+\delta_{wlctrl_2})\cup\mathbf{RCtrl}(wlctrl_2,\delta_{wlctrl_1}+\delta_{wlctrl_2})\\ =\ &\ (?t\leqslant\tau_1+\delta_{wlctrl_1}+\delta_{wlctrl_2};\ wlctrl_1;\ \tau_1:=t)\\ &\cup(?t\leqslant\tau_2+\delta_{wlctrl_1}+\delta_{wlctrl_2};\ wlctrl_2;\ \tau_2:=t)\end{aligned}$$

where $wlctrl_1$ follows Example 1 and
$wlctrl_2\doteq wlm_2:=wl;\big((?wlm_2\geqslant 9.7;fout_2:=1)\cup(?wlm_2\leqslant 2.3;fout_2:=0)\big)$.

Algebraic properties. We retain commutativity and associativity of the parallel composition operator defined in [7]. Commutativity implies that we are able to decompose a system and associativity ensures that we can build it step-by-step. The proof, detailed in the long version [8], relies on the commutativity and associativity of both non-deterministic choice and cost model $\mathcal{C}$.

Modular verification. We adapt [7, Thm. 2] by adding the condition that the individual reactivity bound δ_α of a controller α must neither occur in its functional behavior $\bigcup_{1\leqslant i\leqslant n_\alpha}\alpha_i$ nor in its guarantees. Failing to do so may prevent to re-use a component proof.

Definition 9 (Non-interfering Controllers). *Two controllers* α *and* β *are* non-interfering *if they do not modify the same variables, i.e. the outputs are separated* ($BV(\alpha)\cap BV(\beta)=\varnothing$)*, and if they do not influence the guarantees of the other component* ($FV(G_\alpha)\cap BV(\alpha)=\varnothing$ *and* $FV(G_\beta)\cap BV(\beta)=\varnothing$)*.*

Theorem 2 (Composition of Multi-Choice Reactive Controllers). *Let* α *and* β *be non-interfering multi-choice reactive controllers with program shape* $\boldsymbol{MRCtrl}(\bigcup_{1\leqslant i\leqslant n_\alpha}\alpha_i,\delta_\alpha)$ *and* $\boldsymbol{MRCtrl}(\bigcup_{1\leqslant j\leqslant n_\beta}\beta_j,\delta_\beta)$ *satisfying their compatible contracts* (A_α,G_α) *and* (A_β,G_β) *and let* J_{cmp} *be a composition invariant. Then the parallel composition* $\alpha\otimes\beta$ *is safe, i.e.,* $(\mathcal{E}\wedge A_\alpha\wedge Init_\alpha\wedge A_\beta\wedge Init_\beta)\rightarrow[(\alpha\otimes\beta)^*](G_\alpha\wedge G_\beta)$ *is valid.*

Proof. Similar to Thm. 1 using the additional condition that δ_α (resp. δ_β) does not appear in the functional behavior $\bigcup_{1\leqslant i\leqslant n_\alpha}\alpha_i$ (resp. $\bigcup_{1\leqslant j\leqslant n_\beta}\beta_j$) of the controller, nor in its guarantee G_α (resp. G_β). See long version [8]. □

Non-interference of controllers and compatibility of contracts are standard requirements when modeling a system compositionally and safely.

Example 8 (Safe composition of two water-level controllers). The contract of the first reactive controller $wlctrl_1$ is the same as in Example 5 with necessary changes. The contract for the second controller is :

$$\begin{cases} A_{wlctrl_2} : \top \\ G_{wlctrl_2} : wlm_2 \leqslant 2.3 \rightarrow fout_2 = 0 \\ \qquad\qquad 9.7 \leqslant wlm_2 \rightarrow fout_2 = 1 \\ \qquad\qquad (2.3 \leqslant wlm_2 \leqslant 9.7) \rightarrow (fout_2 = 0 \vee fout_2 = 1) \end{cases}$$

The controller actuates the outlet valve of the system ($fout_2$). It is open if the real water-level of the second tank is too close to the maximum threshold (10 here) to drain the tank and closed in order to fill the tank if too close to the minimum threshold (2). The two controllers are non-interfering, the contracts are compatible and they both satisfy their contracts (verified using the proof calculus of $d\mathcal{L}$). Hence, Thm. 2 guarantees that the parallel composition is safe, *i.e.* that the contract $(A_{wlctrl_1} \wedge A_{wlctrl_2}, G_{wlctrl_1} \wedge G_{wlctrl_2})$ is valid.

4.2 Parallel Composition of Controllable Plants

When composing two continuous components in parallel, the controllability of the resulting system is the minimum of their individual controllability bounds (which is obvious from the semantics of ODEs listed in Tab. 1: safety proofs hold for any non-negative duration, so also for smaller durations).

Modeling Non-interference of controllable plants ensures that their combined continuous dynamics stays true to the isolated dynamics, and that they do not interfere with the guarantees of the respective other component.

Definition 10 (Non-interfering Plants). *Two controllable plants α and β with* ***CPlant***$(\{x' = \theta \ \& \ H\}, \Delta_\alpha)$ *and* ***CPlant***$(\{y' = \eta \ \& \ Q\}, \Delta_\beta)$ *and contracts (A_α, G_α) and (A_β, G_β) are* non-interfering *if $BV(\{x' = \theta \ \& \ H\}) \cap FV(\eta) = \varnothing$ and $BV(\{y' = \eta \ \& \ Q\}) \cap FV(\theta) = \varnothing$, and if $BV(\{x' = \theta \ \& \ H\}) \cap FV(G_\beta) = \varnothing$ and $BV(\{y' = \eta \ \& \ Q\}) \cap FV(G_\alpha) = \varnothing$.*

Note that non-interference implies $\mathrm{BV}(\{x' = \theta \ \& \ H\}) \cap \mathrm{BV}(\{y' = \eta \ \& \ Q\}) = \varnothing$.

Definition 11 (Parallel Composition of Controllable Plants). *Let α and β be non-interfering controllable plants* ***CPlant***$(\{x' = \theta \ \& \ H\}, \Delta_\alpha)$ *and* ***CPlant***$(\{y' = \eta \ \& \ Q\}, \Delta_\beta)$*. The parallel composition $\alpha \otimes \beta$ is an ODE system of the shape* ***CPlant***$(\{x' = \theta, y' = \eta \ \& \ H \wedge Q\}, min(\Delta_\alpha, \Delta_\beta))$.

Example 9 (Composition of two water-level). Here, we compose the water level dynamics of two tanks ($\{wl_1' = fin - fout_1, t' = 1 \ \& \ wl_1 \geqslant 0 \wedge t \leqslant \Delta_{wl_1}\}$ and $\{wl_2' = fout_1 - fout_2, t' = 1 \ \& \ wl_2 \geqslant 0 \wedge \Delta_{wl_2}\}$) to obtain a controllable plant modeling the evolution of both water levels simultaneously. Their respective controllability bounds are $\Delta_{wl_1} = 0.2s$ and $\Delta_{wl_2} = 0.15s$. The controllable plant resulting from the parallel composition expands to $\{wl_1' = fin - fout_1, wl_2' = fout_1 - fout_2, t' = 1 \ \& \ wl_1 \geqslant 0 \wedge wl_2 \geqslant 0 \wedge t \leqslant min(\Delta_{wl_1}, \Delta_{wl_2})\}$.

Algebraic properties. Commutativity and associativity of the parallel composition pattern defined in [7] are preserved. The proof, detailed in the long version [8], follows from commutativity and associativity of "," in ODEs and of operator min.

Modular verification. The conjunction of contracts is retained for parallel composition of continuous components, similar to parallel composition of controllers.

Theorem 3 (Composition of Controllable Plants). *Let α and β be two non-interfering controllable plants* ***CPlant***$(\{x' = \theta \ \& \ H\}, \Delta_\alpha)$, ***CPlant***$(\{y' = \eta \ \& \ Q\}, \Delta_\beta)$ *satisfying their respective compatible contracts* (A_α, G_α), (A_β, G_β), *and let J_{cmp} be a composition invariant. Then the parallel composition $\alpha \otimes \beta$ is safe, i.e.,* $(\mathcal{E} \wedge A_\alpha \wedge Init_\alpha \wedge A_\beta \wedge Init_\beta) \rightarrow [(\alpha \otimes \beta)^*](G_\alpha \wedge G_\beta)$ *is valid.*

Proof. Similar to Thm. 1 after separating the non-interfering plants using the inverse direction of the differential ghost axiom [13], see long version [8]. □

Example 10 (Safe composition of two water-level). The contract for the first water level is the same as in Example 5 with necessary changes. We guarantee that the water level of the second tank is within 2 and 10, provided that there is a controller which reacts appropriately. Its contract is:

$$\begin{cases} A_{wl_2} : G_{wlctrl_2} \\ G_{wl_2} : 2 \leqslant wl_2 \leqslant 10 \end{cases}$$

We apply Thm. 3 to guarantee that the controllable plant modeling the evolution of water levels in distinct connected tanks is safe, *i.e.* it satisfies the contract $(Awl_1 \wedge A_{wl_2}, G_{wl_1} \wedge G_{wl_2})$.

4.3 Parallel Composition of Multi-Choice Reactive Controllers and Controllable Plants

We present the composition of a multi-choice reactive controller with a controllable plant that may result from the composition of several atomic controllable plants. We lift the definition of **CCS** (Sec. 3) to a general integration of controllability and reactivity.

Modeling We define a multi computer-controlled system **MCCS** as the parallel composition of a multi-choice reactive controller with a controllable plant.

Definition 12 (Multi Computer-Controlled System). *A* multi computer-controlled system *is a parallel composition of a multi-choice reactive controller*

MRCtrl$(\bigcup_{1\leqslant i\leqslant n} ctrl_i, \delta)$ *and a controllable plant* ***CPlant***$(\{y' = \theta \ \ \& \ \ H\}, \Delta)$. *The parallel composition* ***MCCS*** *has the hybrid program shape:*

$$\left(\{y' = \theta, t' = 1 \ \ \& \ \ H \wedge \underbrace{\bigwedge_{1\leqslant i\leqslant n} t \leqslant \tau_i + \delta}_{\delta\leqslant\Delta}\} \cup \textbf{\textit{MRCtrl}}\left(\bigcup_{1\leqslant i\leqslant n} ctrl_i, \delta\right)\right)^*$$

The formula $\bigwedge_{1\leqslant i\leqslant n} t \leqslant \tau_i + \delta$ is the conjunction of the reactivity bounds of all the n sub-controllers $ctrl_i$.

Modular verification Cor. 1 lifts Thm. 1 (for a single controller and a single plant) to multi computer-controlled systems of possibly many controllers with a controllable plant representing multiple simultaneous evolutions.

Corollary 1 (Composition of Multi-Choice Reactive Controller and Controllable Plant). *Let* ***MRCtrl***$(\bigcup_{1\leqslant i\leqslant n} ctrl_i, \delta)$ *be a multi-choice reactive controller non-interfering with the controllable plant* ***CPlant***$(\{x' = \theta \ \ \& \ \ H\}, \Delta)$ *satisfying their compatible contracts* (A_{ctrl}, G_{ctrl}) *and* (A_{plant}, G_{plant}). *Further let* J_{cmp} *be a composition invariant. Then,* ***MCCS*** *is safe, i.e.,* $(\mathcal{E} \wedge A_{ctrl} \wedge Init_{ctrl} \wedge A_{plant} \wedge Init_{plant}) \rightarrow [\textbf{\textit{MCCS}}](G_{ctrl} \wedge G_{plant})$ *is valid.*

Proof. Similar to the proof of Thm. 1, but with multi-choice reactive controller instead of a single reactive controller. □

4.4 Parallel Composition of Multi Computer-Controlled Systems

When composing multi computer-controlled systems, the combined reactivity of all controllers must not exceed the combined (minimum) controllability bounds of the plants. Otherwise, safety cannot be guaranteed, as elaborated next.

Modeling The parallel composition of two multi computer-controlled systems is similar to the composition of a multi-choice reactive controller with a controllable plant to obtain a multi computer-controlled system **MCCS**, but with extra care for the combined reactivity bounds obtained from the physical cost model $\mathcal{C}$.

Definition 13 (Parallel Composition of Multi Computer-Controlled Systems). *Let* α *and* β *be two multi computer-controlled systems with shapes* $\alpha \doteq \left(\{x' = \theta, t' = 1 \ \ \& \ \ H \wedge \bigwedge_{1\leqslant i\leqslant n} t \leqslant \tau_i + \delta_\alpha\} \cup \textbf{\textit{MRCtrl}}(\bigcup_{1\leqslant i\leqslant n} \alpha_i, \delta_\alpha)\right)^*$, $\beta \doteq \left(\{y' = \eta, t' = 1 \ \ \& \ \ Q \wedge \bigwedge_{1\leqslant j\leqslant m} t \leqslant \tau_j + \delta_\beta\} \cup \textbf{\textit{MRCtrl}}(\bigcup_{1\leqslant j\leqslant m} \beta_j, \delta_\beta)\right)^*$. *The parallel composition* $\alpha \otimes \beta$ *has the hybrid program shape:*

$$\left(\begin{array}{l} \textbf{\textit{MRCtrl}}\left(\bigcup_{1\leqslant i\leqslant n} \alpha_i, \mathcal{C}(\delta_\alpha, \delta_\beta)\right) \cup \textbf{\textit{MRCtrl}}\left(\bigcup_{1\leqslant j\leqslant m} \beta_j, \mathcal{C}(\delta_\alpha, \delta_\beta)\right) \\ \cup\{x' = \theta, y' = \eta, t' = 1 \ \ \& \ \ H \wedge Q \\ \qquad\qquad \wedge \underbrace{\bigwedge_{1\leqslant i\leqslant n} t \leqslant \tau_i + \mathcal{C}(\delta_\alpha, \delta_\beta) \ \wedge \bigwedge_{1\leqslant j\leqslant m} t \leqslant \tau_j + \mathcal{C}(\delta_\alpha, \delta_\beta)}_{\mathcal{C}(\delta_\alpha,\delta_\beta)\leqslant min(\Delta_\alpha,\Delta_\beta)}\} \end{array}\right)^*$$

Algebraic properties We retain the commutativity and associativity properties (under the condition that the provided max+ cost function $\mathcal{C}$ is commutative and associative), essential for a modular component-based approach.

Proposition 1 (Commutativity and Associativity). *Let α, β and γ be multi computer-controlled systems. Then:*

$$\begin{array}{lll} \alpha \otimes \beta & = \beta \otimes \alpha & \textit{(Commutativity)} \\ (\alpha \otimes \beta) \otimes \gamma & = \alpha \otimes (\beta \otimes \gamma) & \textit{(Associativity)} \end{array}$$

Proof. Follows from Def. 13, see long version [8] for details. □

Modular verification We retain also the respective contracts through the parallel composition. We assume that the individual reactivity bound δ_α of the controller does not occur in its functional behavior, nor in its guarantees.

Theorem 4 (Composition of Multi Computer-Controlled Systems). *Let α and β be non-interfering multi computer-controlled systems (with program shape $\boldsymbol{MCCS}_\alpha$ and $\boldsymbol{MCCS}_\beta$ per Def. 12) satisfying their respective compatible contracts (A_α, G_α) and (A_β, G_β), and let J_{cmp} be a composition invariant. Then the parallel composition $\alpha \otimes \beta$ is safe, i.e., $(\mathcal{E} \wedge A_\alpha \wedge Init_\alpha \wedge A_\beta \wedge Init_\beta) \rightarrow [\alpha \otimes \beta](G_\alpha \wedge G_\beta)$ is valid.*

Proof. We use the commutativity and associativity of operator $\otimes$ to group the multi-choice reactive controllers into a single discrete fragment and the controllable plants into a single continuous fragment. We prove contracts are retained for the discrete fragment by Thm. 2 and for the continuous fragment by Thm. 3. Finally, contracts are retained for the composition of discrete fragment to the continuous fragment using Cor. 1, see long version [8]. □

Outlook In this section, we presented how to extend our previous component-based approach to take into account the timing constraints inherent in the design of a Computer-Controlled System. We have proved that we retain the commutativity and associativity, essential to scale up to realistic systems. Finally, we state and prove theorems to retain contracts through the parallel composition. Theses results give us confidence in the ability of our approach to be adapted to new challenges that will arise when applied to realistic industrial systems.

5 Related Work

Recent component-based verification techniques [10,11] proposed a composition operator in $d\mathcal{L}$ based on the modeling pattern $(ctrl; plant)^*$ to split verification of systems into more manageable pieces. It focuses on separating self-contained components (a controller monitoring its own plant) instead of separating discrete and continuous fragments. This paper extends previous work [7] with capabilities to handle timing relations of CCS upon composition (using max+ cost functions)

and syntactic proofs to facilitate implementation of the proposed techniques as tactics in the theorem prover KeYmaera X.

Hybrid automata [1] are a popular formalism to model hybrid systems, but composition of automata results in an exponential product automaton which is intractable to analyze in practice. *I/O hybrid automata* [9] is an extension of hybrid automata with explicit inputs and outputs. Assume-guarantee reasoning [4] on such automata tackles composability to prevent state-space explosion. Yet, use is in practice restricted to linear hybrid automata. Differential Dynamic Logic handles systems with ODEs (and not just linear ODEs), thus our approach is more expressive.

Hybrid Communicating Sequential Processes (HCSP) [5] is a hybrid extension of the CSP framework. It features a native parallel composition operator and communicating primitives in addition to standard constructs for hybrid systems (sequences, loops, ODEs) and a proof calculus has been proposed in [6]. In contrast, our parallel composition operator is not native and relies on usual constructs of $d\mathcal{L}$. The benefit is that we do not have to extend $d\mathcal{L}$ and check the soundness of such extension, but it requires additional effort to mechanize it into the theorem prover KeYmaera X. Also, our approach provides engineering support for timing aspects and modular verification principles.

6 Conclusion

We presented a component-based verification technique for modularly designing and verifyinf computer-controlled systems with special focus on timing constraints (reactivity and controllability) and modular verification. Our concepts enable systematic modeling of CCS in a modular way while maintaining algebraic properties of composition patterns and preserving contract proofs through composition. We additionally support reasoning on non-functional properties (reactivity, controllability) through multiple compositions of reactive controllers and plants. This paves the way to ultimately model complex cyber-physical systems (several controllers running in parallel according to a generic max+ cost function that monitor different plants) from only simple, atomic components. Verification of safety properties for the global system reduces to component safety proofs with only mild assumptions on the reactivity of controllers (does not exceed the controllability of plants) and compatibility between contracts.

As future work, we intend to allow more aggressive compositions to lift restrictions of the techniques presented here: allow some interference in the parallel composition of controllable plants and reactive controllers with additional compatibility proofs (lift non-interference restriction of Cor. 1); allow time and reactivity in the predictions and guarantees of controllers with refactoring techniques to strengthen control choices upon composition (lift restriction of Thm. 4 that does not grant controllers to exploit their reactivity bounds δ for control decisions); and support fine-grained communication going beyond shared variables with communication channels as in Hybrid Communicating Sequential Processes. For proof automation, we intend to implement the theorems of this paper as tactics in the KeYmaera X theorem prover.

References

1. Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, pages 209–229. Springer, 1993.
2. Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto Sangiovanni-Vincentelli, Werner Damm, Thomas Henzinger, and Kim Guldstrand Larsen. Contracts for system design. Technical report, 2012.
3. Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. Keymaera X: an axiomatic tactical theorem prover for hybrid systems. In *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, pages 527–538, 2015.
4. Thomas A Henzinger, Marius Minea, and Vinayak Prabhu. Assume-guarantee reasoning for hierarchical hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 275–290. Springer, 2001.
5. He Jifeng. From CSP to hybrid systems. In *A classical mind*, pages 171–189. Prentice Hall International (UK) Ltd., 1994.
6. Jiang Liu, Jidong Lv, Zhao Quan, Naijun Zhan, Hengjun Zhao, Chaochen Zhou, and Liang Zou. A calculus for hybrid CSP. In *Asian Symposium on Programming Languages and Systems*, pages 1–15. Springer, 2010.
7. Simon Lunel, Benoît Boyer, and Jean-Pierre Talpin. Compositional proofs in differential dynamic logic. In Axel Legay and Klaus Schneider, editors, *ACSD*, 2017.
8. Simon Lunel, Stefan Mitsch, Benoît Boyer, and Jean-Pierre Talpin. Parallel composition and modular verification of computer controlled systems in differential dynamic logic. *CoRR*, abs/1907.02881, July 2019.
9. Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. Hybrid I/O automata. *Inf. Comput.*, 185(1):105–157, 2003.
10. Andreas Müller, Stefan Mitsch, Werner Retschitzegger, Wieland Schwinger, and André Platzer. A component-based approach to hybrid systems safety verification. In Erika Abraham and Marieke Huisman, editors, *IFM*, volume 9681 of *LNCS*, pages 441–456. Springer, 2016.
11. Andreas Müller, Stefan Mitsch, Werner Retschitzegger, Wieland Schwinger, and André Platzer. Tactical contract composition for hybrid system component verification. *STTT*, 20(6):615–643, 2018. Special issue for selected papers from FASE'17.
12. André Platzer. The complete proof theory of hybrid systems. In *LICS*, pages 541–550. IEEE, 2012.
13. André Platzer. A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reas.*, 59(2):219–265, 2017.
14. André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, Cham, 2018.
15. André Platzer and Yong Kiam Tan. Differential equation axiomatization: The impressive power of differential ghosts. In Anuj Dawar and Erich Grädel, editors, *LICS*, pages 819–828, New York, 2018. ACM.
16. Julien Signoles, Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, and Boris Yakobowski. Frama-C: a software analysis perspective. *Formal Aspects of Computing*, 27, 10 2012.