Cambridge Large One

www.cambridge.org

Results

Keywords:
cyber-physical system, controller synthesis, reset controller, transverse set, reach-avoid set, differential invariant, semi-definite programming

Reset Controller Synthesis: A Correct-by-Construction Way to the Design of CPS

Naijun Zhan ,[1,2,3] Han Su,[2,3] Mengfei Yang,[4] and Bin Gu[*,5]

[1]School of Computer Science, Peking University, Beijing, China
[2]SKLCS, Institute of Software, CAS, Beijing, China
[3]University of Chinese Academy of Sciences, CAS, Beijing, China
[4]China Academy of Space Technology, Beijing, China
[5]Beijing Institute of Control Engineering, Beijing, China

*Author for correspondence. Email: gubin@ios.ac.cn

Abstract

Controller synthesis offers a correct-by-construction methodology to ensure the correctness and reliability of safety-critical cyber-physical systems (CPS). Controllers are classified based on the types of controls they employ, which include reset controllers, feedback controllers, and switching logic controllers. Reset controllers steer the behavior of a CPS to achieve system objectives by restricting its initial set and redefining its reset map associated with discrete jumps. Although the synthesis of feedback controllers and switching logic controllers has received considerable attention, research on reset controller synthesis is still in its early stages, despite its theoretical and practical significance. This paper outlines our recent efforts to address this gap. Our approach reduces the problem to computing differential invariants and reach-avoid sets. For polynomial CPS, the resulting problems can be solved by further reduction to convex optimizations. Moreover, considering the inevitable presence of time delays in CPS design, we further consider synthesizing reset controllers for CPS that incorporate delays.

Introduction

As defined by Baheti and Gill in (Baheti and Gill 2011), *cyber-physical systems (CPS)* refers to a new generation of systems integrating computational and physical capabilities, capable of interacting with humans through various modalities. The ability to interact with, and expand the capabilities of, the physical world through *computation*, *communication*, and *control* serves as an enabler for future technology developments. CPS is pervasive in our daily life, examples include spacecrafts, high speed train control systems, automated plants and factories, and so on. Many of these systems are entrusted with safety-critical tasks, necessitating the development of formally verifiable CPS that are both safe and reliable. However, efficiently developing such CPS remains a longstanding challenge.

Controller synthesis provides a correct-by-construction mechanism to guarantee the correctness and reliability of CPS. In essence, controller synthesis endeavors to create an operational behavior model for a component, based on a model of assumed environmental behaviors and a system goal. This process ensures the system reliably achieves the specified objective when the environment aligns with the provided assumptions. Controller synthesis has attracted increasing attention from computer science and control theory in the past decades. In the case of CPS, an operation (i.e., control) could be either an input to dynamics, a switch condition from one mode to another, an initial condition for each mode, or a reset map when conducting discrete jumps. Depending on the types of controls, controllers can be naturally classified into feedback controllers, switching logic controllers, and reset controllers. In the literature, there is a huge bulk of work on the synthesis of controllers of the first two types, please refer to (Tomlin, Lygeros, and Sastry 2000; Asarin et al. 2000; Coogan and Arcak 2012; Jha et al. 2010; Taly, Gulwani, and Tiwari 2011; Girard 2012; Gulwani and Tiwari 2008; Taly, Gulwani, and Tiwari 2011; Zhao, Zhan, and Kapur 2013) and the references therein. However, the synthesis of controllers of the third type is still in the infant stage, despite its theoretical and practical significance. Many important practical problems can be reduced to reset controller synthesis, e.g., the substantial instantaneous change in velocity of a spacecraft induced by impulsive controls in satellite rendezvous (Brentari et al. 2018), re-configuring safety-critical devices such as spacecraft when an exception happens, etc. Furthermore, as indicated by the following motivating example, in some cases, the system goal cannot be achieved only with feedback controllers and/or switching logic controllers.

CAMBRIDGE UNIVERSITY PRESS

Example 1 (A motivating example (Liu et al. 2023)). *Consider a CPS given in Fig. 1. Suppose the safe sets in mode q_1 and q_2*

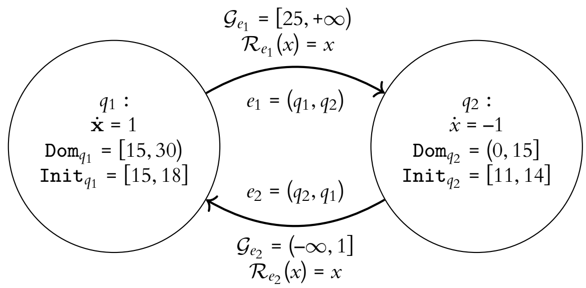

Figure 1. Hybrid Automaton for Example 1

are $\mathcal{S}_1 = [15, 31)$, $\mathcal{S}_2 = (0, 14]$, respectively. As the dynamics in the two modes both are autonomous without inputs, it is impossible to find feedback controllers for them to maintain safety. Moreover, one may easily observe that once a discrete jump happens, the system will not be safe anymore. This means only strengthening the domain constraints and guards for discrete jumps to maintain safety is trivially impossible. However, it is possible to synthesize a reset controller to maintain safety.

Related Work Numerous studies have delved into verifying hybrid systems, which can broadly be categorized into model-checking and theorem proving. the former is essentially based on reachable set computation, currently can only handle bounded time. For example, tools such as SpaceEx (Frehse et al. 2011), iSAT-ODE (Eggers, Fränzle, and Herde 2008), dReach (Kong et al. 2015), and Flow* (Chen, Ábrahám, and Sankaranarayanan 2013) fall within this category. In contrast, the latter can provide unbounded verification of HSs with scalability based on specification logics and invariant generation, e.g., *differential dynamic Logic* (dL) (Platzer 2012) and *hyrid Hoare logic* (HHL) (Liu et al. 2010; Zhan et al. 2023). dL demonstrates significant capability in deducing verification for HSs, proving effective across various verification challenges, including the verification of liveness properties (Tan and Platzer 2019) and switched systems (Tan and Platzer 2021) with the help of the tool KeYmaera X (Platzer 2010). While, HHL can handle more complicated behaviors of HSs such as communication, concurrency, and so on, with the help of the tool HHLProver (Wang, Zhan, and Zou 2015). Event-B (Richard 2024; Richard et al. 2017; Richard et al. 2015; Butler, Abrial, and Banach 2016; Dupont et al. 2021; Dupont et al. 2022) also stands as a useful method for formal modeling and verifying HSs.

Verification of HSs can also be pursued in a correct-by-construction manner through refinement syntactically (Back and Wright 2012) or controller synthesis semantically (Bozga and Sifakis 2022). Refinement plays a key role in classical programming theories, however, the counterparts for HSs are really few in the literature, although model-based design has become dominant in the design of HSs. (Loos and Platzer 2016) proposed differential refinement logic to cope with refinement relation among different levels of abstraction for a given HS. (Yan et al. 2020) defined a set of refinement rules for transforming HCSP to SystemC, and (S. Wang et al. 2024) proposed a set of refinement rules for transforming HCSP to ANSI-C, both with the correctness guarantee based on approximate bisimulation.

Extensive work has been dedicated to controller synthesis for HSs. One category of research focuses on feedback controllers, with various methods addressing this type of synthesis problem, including moment-based methods (Zhao, Mohan, and Vasudevan 2019), Hamilton-Jacobi-based methods (Tomlin, Lygeros, and Sastry 2000), barrier certificates-based methods (Ames et al. 2016), abstraction-based methods (Girard 2012), and counter-example-guided inductive synthesis methods (Abate et al. 2017). Another category addresses the synthesis problem of switching controllers, which can be classified into abstraction-based methods (Girard 2012; Tabuada 2009; Belta, Yordanov, and Gol 2017), and constraint-solving-based methods (Taly, Gulwani, and Tiwari 2011; Zhao, Zhan, and Kapur 2013; Taly and Tiwari 2010). However, since its initial exploration in (Clegg 1958), there has been limited research on reset controllers, which is the focus of our paper.

Synopsis of Reset Controller Synthesis

In this paper, we summarize our recent work on the reset controller synthesis for CPS, details can be found in (Liu et al. 2023; Su et al. 2023).

Reset Controller Synthesis Without Time-delay

Firstly, we investigate reset controller synthesis with ideal mathematical models, i.e., *hybrid automata* (HA), which is a popular model for CPS. Formally,

Definition 1. *An HA $\mathcal{H}$ is a tuple $(\mathcal{Q}, \mathcal{X}, \mathbf{f}, \mathtt{Init}, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R})$, where*

- $\mathcal{Q} = \{q_1, q_2, \dots\}$ *is a finite set of* modes*;*
- $X = \{x_1, \dots, x_n\}$ *is a set of* continuous state variables*, also written as a vector of variables* $\mathbf{x}$*, which is interpreted over* $\mathbb{R}^n$*. Normally, we use* $\mathcal{X} \subseteq \mathbb{R}^n$ *to denote the continuous state space, and a (hybrid) state of the system is represented as* $(q, \mathbf{x}) \in \mathcal{Q} \times \mathcal{X}$*;*
- $\mathtt{Init} \subseteq \mathcal{Q} \times \mathcal{X}$ *is a set of* initial states*;*
- $\mathtt{Dom} : \mathcal{Q} \to P(\mathcal{X})$ *assigns to each* $q \in \mathcal{Q}$ *a domain, written as* $\mathtt{Dom}_q \subseteq \mathcal{X}$*. The system can reside in a mode only if the domain constrain of the mode is satisfied;*
- $\mathbf{f} : \mathcal{Q} \to (\mathcal{X} \to \mathbb{R}^n)$ *assigns to each* $q \in \mathcal{Q}$ *a* locally Lipschitz continuous vector field $\mathbf{f}_q$ *defined over* $\mathtt{Dom}_q$*;*
- $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ *is a set of* edges (jumps)*;*
- $\mathcal{G} : \mathcal{E} \to P(\mathcal{X})$ *assigns a* guard condition $\mathcal{G}_e$ *to each edge e, s.t. the discrete jump can happen only if its guard is satisfied;*
- $\mathcal{R}(\cdot, \cdot) : \mathcal{E} \times \mathcal{X} \to P(\mathcal{X})$ *assigns a* reset map $\mathcal{R}_e$ *to each edge* $e \in \mathcal{E}$ *with* $\mathcal{R}_e : \mathcal{X} \to P(\mathcal{X})$*, that relates a state in the pre-mode to a set of states in the post-mode*[1].

1. For an edge $e = (q, p)$, we refer to q as the pre-mode of e, and p as the post-mode of e

The semantics of HA in terms of *(hybrid) trajectories* is defined in a standard way, please refer to (Zhan, Shuling, and Zhao 2017) for a comprehensive introduction to HA.

Now, we can formulate the problems of interest as follows.

Problem 1 (Reset Controller Synthesis). *Given an HA $\mathcal{H}$ as Definition 1, we consider*

- ***Problem 1.1:*** *for a given safe set $\mathcal{S} \subseteq \mathcal{Q} \times \mathcal{X}$, whether one can redefine* $\mathtt{Init}$ *and $\mathcal{R}$, and obtain a redesigned HA $\mathcal{H}' = (\mathcal{Q}, X, f, \mathtt{Init}^r, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R}^r)$, which is safe w.r.t. $\mathcal{S}$.*
- ***Problem 1.2:*** *for a given safe set $\mathcal{S} \subseteq \mathcal{Q} \times \mathcal{X}$ and a target set $\mathtt{TR} \subseteq \mathcal{Q} \times \mathcal{X}$, whether one can redefine* $\mathtt{Init}$ *and $\mathcal{R}$, and obtain a redesigned HA $\mathcal{H}' = (\mathcal{Q}, X, f, \mathtt{Init}^r, \mathtt{Dom}, \mathcal{E}, \mathcal{G}, \mathcal{R}^r)$, s.t. for any $(q, \mathbf{x}) \in \mathtt{Init}^r$, any trajectory starting from $(q, \mathbf{x})$ must reach $\mathcal{T}$, and $\mathcal{H}'$ is safe w.r.t. $\mathcal{S}$ before reaching into $\mathcal{T}$.*

To address the above two problems, the following notions are needed.

Definition 2 (Transverse Set). *Given a vector field* $\mathbf{f}$ *and a set $S \subseteq \mathbb{R}^n$,* the transverse set *of S w.r.t.* $\mathbf{f}$*, denoted by $\mathtt{trans}_{\mathbf{f}\uparrow S}$ of* $\mathbf{f}$ *over S, is defined by*

$$\mathtt{trans}_{\mathbf{f}\uparrow S} \widehat{=} \{\mathbf{x} \in \partial S \mid \forall \epsilon > 0,\ \exists t \in [0, \epsilon), \phi(\mathbf{x}, t) \notin S\}$$

where ∂S is the boundary of S.

Intuitively, any trajectory starting from *the transverse set* of S w.r.t. $\mathbf{f}$ will leave S immediately. For example, in Fig. 2, $\mathbf{x}_2 \in \mathtt{trans}_{\mathbf{f}\uparrow S}$, $\mathbf{x}_3 \in \mathtt{trans}_{\mathbf{f}\uparrow S}$, $\mathbf{x}_4 \in \mathtt{trans}_{\mathbf{f}\uparrow S}$, but $\mathbf{x}_1 \notin \mathtt{trans}_{\mathbf{f}\uparrow S}$. Clearly, if $\mathtt{trans}_{\mathbf{f}\uparrow S}$ is empty, then any trajectory starting from S stays within S forever, which implies S is a differential invariant (see Definition 3).

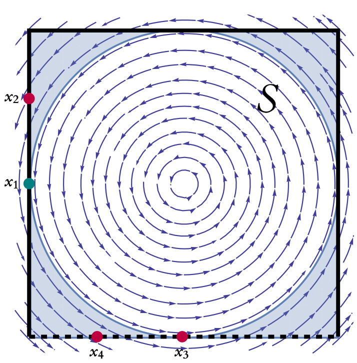

Figure 2. An example of transverse set. The arrows indicate the vector field of $\mathbf{f}$. The area within the black square is a safe area S. The dotted line on the lower border of the square indicates that this part of the boundary is not within the safe area.

Definition 3 (Differential Invariant (DI)). *A set C is a **differential invariant** of vector field* $\mathbf{f}$ *w.r.t. a set S if for all $\mathbf{x} \in C$ and $T \geq 0$*

$$\begin{pmatrix}\forall t \in [0, T]. \\ \phi(\mathbf{x}, t) \in S\end{pmatrix} \implies \begin{pmatrix}\forall t \in [0, T]. \\ \phi(\mathbf{x}, t) \in C\end{pmatrix}$$

In other words, $\mathtt{trans}_{\mathbf{f}\uparrow S \cap C} = \emptyset$. Clearly, if $S \subseteq C$, then C is a DI of $\mathbf{f}$ w.r.t. S. Normally, we are only interested in such DIs that are subsets of the domain constraint S.

Definition 4 (Reach-Avoid Set). *Given a vector field* $\mathbf{f}$*, an initial set $\mathcal{X}_0$, a safe set $\mathcal{S}$ and a target set $\mathcal{T}$, the (maximal) reach-avoid set $\mathtt{RA}(\mathcal{X}_0 \xrightarrow[\mathbf{f}]{S} \mathcal{T})$ is defined by*

$$\mathtt{RA}(\mathcal{X}_0 \xrightarrow[\mathbf{f}]{S} \mathcal{T}) \widehat{=} \left\{ \mathbf{x} \in \mathcal{X}_0 \cap S \;\middle|\; \begin{array}{l} \exists T \geq 0, \\ \quad \forall t \in [0, T), \phi(\mathbf{x}, t) \in S \wedge \\ \quad \forall \epsilon > 0, \exists t \in [T, T + \epsilon), \phi(\mathbf{x}, t) \in \mathcal{T} \end{array} \right\}$$

For example, in Fig.2, the blue shaded area (including the border) is $\mathtt{RA}(\mathcal{S} \xrightarrow[f]{S} \mathtt{trans}_{\mathbf{f}\uparrow S})$.

Problem 1.1 can be solved by requiring that in each mode $q \in \mathcal{Q}$ any continuous flow from the initial set $\mathtt{Init}_q$ either

i) safely reaches the must-jump part of a jump eventually, that is $\bigcup_{p \in \mathtt{Post}(q)} \mathtt{RA}(\mathtt{SD}_q \xrightarrow[f_q]{\mathtt{SD}_q} \mathtt{Dom}_q^c \cap \mathcal{G}_{e=(q,p)})$, where $\mathtt{SD}_q = \mathtt{Dom}_q \cap S_q$, $\mathtt{Post}(q)$ stands for the set of modes to which there is a jump from q, and $\mathtt{Dom}_q^c$ for the complement of $\mathtt{Dom}_q$; or
ii) stays inside the mode forever and subject to the safety constraint, that is $\mathtt{SD}_q \setminus \mathtt{RA}(\mathtt{SD}_q \xrightarrow[f_q]{\mathtt{SD}_q} \mathtt{trans}_{f_q \uparrow \mathtt{SD}_q})$.

Obviously, i) corresponds to a reach-avoid problem, which considers how to compute the maximal set of initial states s.t. flows starting from them reach the target eventually while remaining inside the safe set before the reach. As showed in (Liu et al. 2023), by introducing a template, whose 0-sublevel set is an inner-approximation of the reach-avoid set, the maximal reach-avoid set of polynomial hybrid automata can be inner-approximated by solving a certain convex programming problem, which can be done using off-the-shell SDP solvers. After that, new reset maps corresponding to the jump are also synthesized to guarantee safety in the post-mode. While, ii) corresponds to a differential invariant generation problem, which can be solved relatively well by exploiting existing methods, e.g., (Liu, Zhan, and Zhao 2011; Ghorbal and Platzer 2014; Xue et al. 2019; Q. Wang et al. 2022).

For example, consider a given HA and a safe set as in Fig. 3. In the first step, we compute the must-jump parts respectively in q_1 and q_2 by computing the corresponding reach-avoid sets, and obtain

$$\mathtt{RA}_1 = \mathtt{RA}(\mathtt{SD}_{q_1} \xrightarrow[\mathbf{f}_{q_1}]{\mathtt{SD}_{q_1}} \mathtt{trans}_{\mathbf{f}_{q_1} \uparrow \mathtt{SD}_{q_1}^c \cap \mathcal{G}_{e_1}}),$$

$$\mathtt{RA}_2 = \mathtt{RA}(\mathtt{SD}_{q_2} \xrightarrow[\mathbf{f}_{q_2}]{\mathtt{SD}_{q_2}} \mathtt{trans}_{\mathbf{f}_{q_2} \uparrow \mathtt{SD}_{q_2}^c \cap \mathcal{G}_{e_2}})$$

In the second step, we can compute DIs respectively in q_1 and q_2 by computing the corresponding transverse set, and obtain

$$DI_1 = \mathtt{SD}_{q_1} \setminus \mathtt{RA}(\mathtt{SD}_{q_1} \xrightarrow[\mathbf{f}_{q_1}]{\mathtt{SD}_{q_1}} \mathtt{trans}_{\mathbf{f}_{q_1}\uparrow \mathtt{SD}_{q_1}}),$$
$$DI_2 = \mathtt{SD}_{q_2} \setminus \mathtt{RA}(\mathtt{SD}_{q_2} \xrightarrow[\mathbf{f}_{q_2}]{\mathtt{SD}_{q_2}} \mathtt{trans}_{\mathbf{f}_{q_2}\uparrow \mathtt{SD}_{q_2}})$$

Finally, we can redefine the initial set and reset map as follows:

$$\mathtt{Init}^r_{q_1} = \mathtt{Init}_{q_1} \cap (\mathtt{DI}_1 \cup \mathtt{RA}_1),$$
$$\mathtt{Init}^r_{q_2} = \mathtt{Init}_{q_2} \cap (\mathtt{DI}_2 \cup \mathtt{RA}_2),$$
$$\mathcal{R}^r_{e_1}(x) \subseteq \mathtt{DI}_2 \cup \mathtt{RA}_2 \ \forall x \in \mathcal{G}_{e_1},$$
$$\mathcal{R}^r_{e_2}(x) \subseteq \mathtt{DI}_1 \cup \mathtt{RA}_1 \ \forall x \in \mathcal{G}_{e_2}$$

The redefined HA is also shown in Fig. 3.

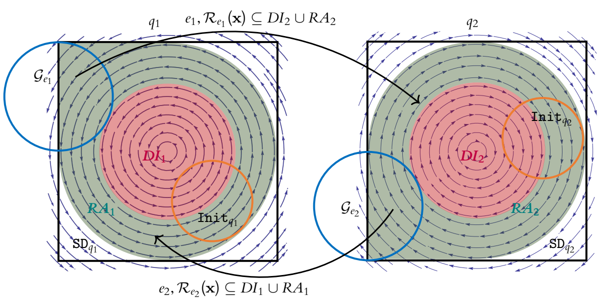

Figure 3. An example for solving **Problem 1.1**. The areas enclosed by black squares represent the intersection of the domain and the safe set, denoted as $\mathtt{SD}_q$. The regions enclosed by orange circles indicate the initial sets, while those enclosed by blue circles represent the guard conditions. The red regions denote the differential invariants of the respective modes, while the green regions signify the reach-avoid sets.

Problem 1.2 In this case, step i) becomes more involved, as a flow may also reach the target set of the current mode, but it is still a reach-avoid problem and can thus be treated similarly. Furthermore, a non-trivial liveness constraint rules out the case of ii). However, an additional problem must be addressed, i.e., how to avoid the unreachability caused by infinite loops among the modes. This problem can be solved by searching and blocking all simple loops among the modes.

For example, to synthesize a reset controller for the HA given in Fig. 4 with the given safe and target set, we have to

- block all trajectories that can reach q_3, as $\mathcal{T}_{q_3} = \emptyset$, which implies the liveness cannot be satisfied along these trajectories;
- block all trajectories with a simple loop containing q_0, q_1 and q_2, as such trajectories could evolve infinitely along the loop, and never reach the target.

We omit the technical details of how to implement the above idea by redefining the initial set and reset map, and technical details can be found in (Liu et al. 2023).

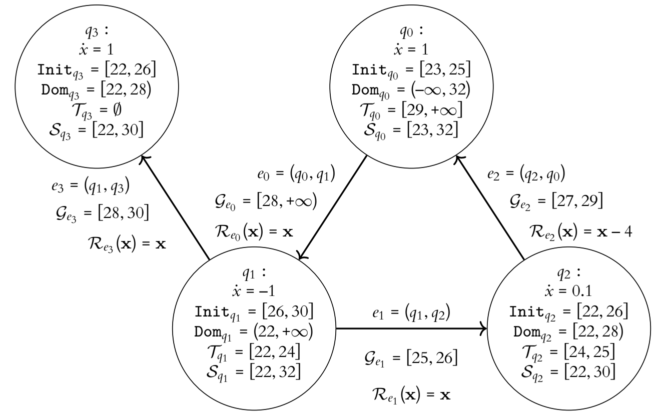

Figure 4. An example of solving **Problem 1.2**

Reset controller synthesis with time-delay

Time-delay is inevitable in the design of CPS, because of

- conversions between analog and digital signal domains,
- complex digital signal-processing chains enhancing,
- filtering and fusing sensory signals before they enter control,
- sensor networks harvesting multiple sensor sources before feeding them to control,
- network delays in networked control applications physically removing the controller(s) from the control path, and just name a few.

The delay-free assumption makes the problem mathematically simple, but physically impossible, even impractical, as it may lead to deteriorated control performance and invalid verification certificates obtained by abstracting away time-delay in practice. So, realistically, we should consider this issue in the context of time-delay like delay hybrid automata (Bai et al. 2021) so that the time spent by the reset controller can be modeled as time delay and thus it can be taken into account. Thus, we investigate the reset controller synthesis problem for delay hybrid systems (dHS), which contains delay in both continuous evolution and discrete transitions, and propose a novel reach-avoid analysis based method.

Reach-avoid for delay differential equations (DDE) Consider a DDE of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{x}(t-\tau)), \quad \mathbf{f} \in \mathbb{R}\left[\mathbf{x}(t), \mathbf{x}(t-\tau)\right]^n \qquad (1)$$

a safe set $\mathcal{S} \in \mathbb{R}^n$ and a target set $\mathcal{T} \in \mathbb{R}^n$, a (the maximal) reach-avoid set $\mathcal{RA}(\mathbf{f}, \mathcal{S}, \mathcal{T})$ is defined as

$$\mathcal{RA}(\mathbf{f}, \mathcal{S}, \mathcal{T}) \,\widehat{=}\, \left\{ \phi \in \mathcal{C}([-\tau, 0], \mathcal{S}) \;\middle|\; \begin{array}{l} \exists\, t' \in \mathbb{R}, \\ \quad x^{\phi}(t') \in T \,\wedge \\ \quad \forall\, t \in [-\tau, t'),\ x^{\phi}(t) \in \mathcal{S} \end{array} \right\}$$

where $\mathcal{C}([-\tau, 0], \mathcal{S})$ stands for the set of all continuous functions from $[-\tau, 0]$ to $\mathcal{S}$, x^{ϕ} denote the trajectory of (1) with initial function ϕ.

Definition 5 (Reach-Avoid Barrier Functional (RABFal))**.** *Given a DDE of the form (1) with domain $D \subseteq \mathbb{R}^n$, safe set $\mathcal{S}$ and target set $\mathcal{T}$ represented by*

$$\mathcal{S} \,\widehat{=}\, \{\mathbf{x} \in D \mid s(\mathbf{x}) \leq 0\}, \quad \mathcal{T} \,\widehat{=}\, \{\mathbf{x} \in D \mid g(\mathbf{x}) \leq 0\},$$

we call $H : \mathcal{C}([-\tau, 0], D) \to \mathbb{R}$ a reach-avoid barrier functional *if we can find a bounded function $w : D \to \mathbb{R}$ such that the following conditions are satisfied:*

$$-\frac{dH(\mathbf{x}_t)}{dt} \geq 0,\ \forall \mathbf{x}_t \in \mathcal{C}([-\tau, 0], \mathcal{S}) \tag{2}$$

$$H(\mathbf{x}_t) \geq 0,\ \forall \mathbf{x}_t \in \mathcal{C}([-\tau, 0], \mathcal{S}),\ s.t.\ \mathbf{x}_t(0) \in \partial\mathcal{S} \tag{3}$$

$$H(\mathbf{x}_t) - \frac{dw(\mathbf{x}_t(0))}{dt} \geq g(\mathbf{x}_t(0)),\ \forall \mathbf{x}_t \in \mathcal{C}([-\tau, 0], \mathcal{S}) \tag{4}$$

Theorem 1 (Su et al. 2023)**.** *Given a DDE of the form* (1)*, safe set $\mathcal{S}$ and target set $\mathcal{T}$, the set $\mathcal{RA}_{in}$ defined by the 0-sublevel set of H, i.e.,*

$$\mathcal{RA}_{in} \,\widehat{=}\, \{\boldsymbol{\phi} \in \mathcal{C}([-\tau, 0], \mathcal{S}) \mid H(\boldsymbol{\phi}) < 0\} \tag{5}$$

is an inner-approximation of $\mathcal{RA}(\mathbf{f}, \mathcal{S}, \mathcal{T})$, i.e., $\mathcal{RA}_{in} \subseteq \mathcal{RA}(\mathbf{f}, Safe, \mathcal{T})$.

In (Su et al. 2023), it is proved that synthesizing such RAB-Fal can be reduced to solving SDP.

Definition 6 (Delay Hybrid Automata (dHA) (Bai et al. 2021))**.** *A dHA $\mathcal{H}$ is a tuple $(\mathcal{Q}, X, \mathtt{Init}, \mathtt{Dom}, \mathbf{f}, \mathcal{E}, \mathcal{G}, \mathcal{R}, ST)$, where*

- $\mathcal{Q} = \{q_1, \ldots, q_m\}$ *is a finite set of modes;*
- $X = \{x_1, \ldots, x_n\}$ *is a set of continuous state; variables, written as $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$;*
- $\mathtt{Init} \subseteq \mathcal{Q} \times \mathcal{C}([-\tau, 0], \mathbb{R}^n)$ *assigns a set of initial states to each mode;*
- $\mathtt{Dom} : \mathcal{Q} \to 2^{\mathbb{R}^n}$ *defines a domain constraint for each mode $q \in \mathcal{Q}$, denoted by $\mathtt{Dom}_q \subseteq \mathbb{R}^n$*
- $\mathbf{f} : \mathcal{Q} \to (\mathcal{C}([-\tau, 0], \mathbb{R}^n) \to \mathbb{R}^n)$ *defines the continuous dynamics with delay for each mode q, denoted by $\mathbf{f}_q$ with the type $\mathcal{C}([-\tau, 0], \mathbb{R}^n) \to \mathbb{R}^n$;*
- $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ *is a set of discrete transitions;*
- $\mathcal{G} : \mathcal{E} \to 2^{\mathbb{R}^n}$ *assigns a switching guard $\mathcal{G}_e \subseteq \mathbb{R}^n$ to each discrete transition $e \in \mathcal{E}$;*
- $\mathcal{R} : \mathcal{E} \to (\mathbb{R}^n \to \mathcal{C}([-\tau, 0], \mathbb{R}^n))$ *assigns a reset function $\mathcal{R}_e$ to each discrete transition $e \in \mathcal{E}$ with $\mathcal{R}_e : \mathbb{R}^n \to \mathcal{C}([-\tau, 0], \mathbb{R}^n)$;*
- $ST \subseteq \mathcal{E} \times \mathbb{R}$ *assigns a switching time to each discrete transition $e \in \mathcal{E}$.*

Problem 2 (Reset Controller Synthesis for dHA)**.** *Given a dHA $\mathcal{H}$ as Definition 6, for a given compact safe set $\mathcal{S} \subseteq Q \times X$ and a compact target set $\mathcal{T} \subseteq Q \times X$, whether we can find a new $\mathtt{Init}^r$ and $\mathcal{R}^r$ such that all executions of the redesigned dHA $\mathcal{H}^r = (\mathcal{Q}, X, \mathtt{Init}, \mathtt{Dom}, \mathbf{f}, \mathcal{E}, \mathcal{G}, \mathcal{R}^r, \mathtt{Init}^r, ST)$ will reach $\mathcal{T}$ while stay in $\mathcal{S}$ before that.*

With the above notions and notations, **Problem 2** can be solved quite similarly to **Problem 1.2**, we will use the following example to illustrate the procedure, the details can be found in Su et al. 2023.

As an illustrative example, consider a dHA given by Fig. 5. The synthesis procedure can be sketched by the following four steps:

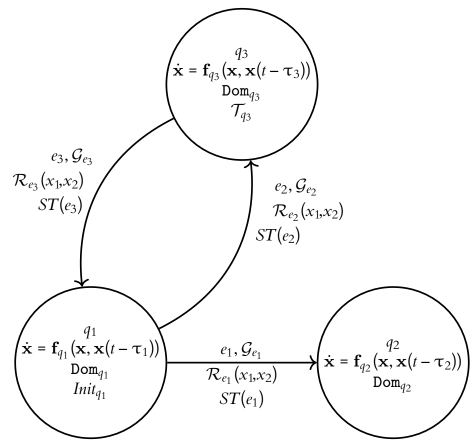

Figure 5. A Running Example of dHA

Step 1: First, compute the reach-avoid set for each mode w.r.t. the target and the guards of the outgoing jumps from it, and then partition a mode into several sub-modes so that their reach-avoid sets are mutually disjoint. For instance, for the running example, as shown in Fig 6, q_1 is split into three sub-modes q_{11}, q_{12} and q_{13}, their reach-avoid sets are computed as below:

$$\mathcal{RA}_{in}(1,1) \,\widehat{=}\, \mathtt{RA}(\mathtt{SD}_{q_1} \xrightarrow[\mathbf{f}_{q_1}]{\mathtt{SD}_{q_1}} g_{11} \cap \mathtt{Dom}^c_{q_1})$$

$$\mathcal{RA}_{in}(1,2) \,\widehat{=}\, \mathtt{RA}(\mathtt{SD}_{q_1} \xrightarrow[\mathbf{f}_{q_1}]{\mathtt{SD}_{q_1}} g_{12} \cap \mathtt{Dom}^c_{q_1})$$

$$\mathcal{RA}_{in}(1,3) \,\widehat{=}\, \mathtt{RA}(\mathtt{SD}_{q_1} \xrightarrow[\mathbf{f}_{q_1}]{\mathtt{SD}_{q_1}} g_{13} \cap \mathtt{Dom}^c_{q_1})$$

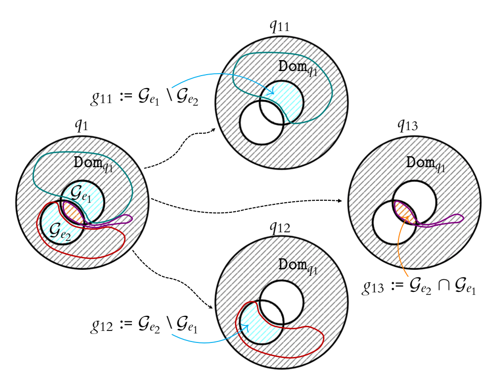

Figure 6. Mode partition of q_1. On the left side, we have mode q_1 with guard conditions $\mathcal{G}(e_1)$ and $\mathcal{G}(e_2)$ represented by blue slashes, and their intersection is depicted by orange slashes. The reach-avoid set to $\mathcal{G}(e_1) \cup \mathcal{G}(e_2)$ can be partitioned into three disjoint regions: g_{11}, g_{12}, and g_{13}, as shown above. Accordingly, mode q_1 is partitioned into three sub-modes: q_{11}, q_{12}, and q_{13}.

With the same manner, the partition of mode q_3 is shown in Fig 7. Their reach-avoid set are computed as below:

$$\mathcal{RA}_{in}(3,0) \mathrel{\widehat{=}} \mathtt{RA}(\mathtt{SD}_{q_3} \xrightarrow[\mathbf{f}_{q_3}]{\mathtt{SD}_{q_3}} g_{30})$$

$$\mathcal{RA}_{in}(3,1) \mathrel{\widehat{=}} \mathtt{RA}(\mathtt{SD}_{q_3} \xrightarrow[\mathbf{f}_{q_3}]{\mathtt{SD}_{q_3}} g_{31} \cap \mathtt{Dom}^c_{q_3})$$

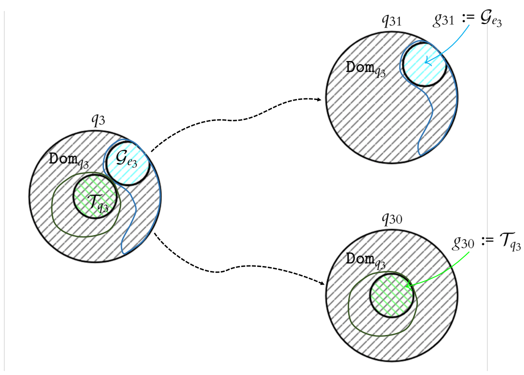

Figure 7. Mode partition of q_3. The left side is mode q_3 with the guard condition $\mathcal{G}_{e_3}$ (blue slashes) and the target set $\mathcal{T}_{q_3}$ (green slashes). Correspondingly, q_3 is partitioned into two sub-modes: q_{30} with $g_{30} = \mathcal{T}_{q_3}$, and q_{31} with $g_{31} = \mathcal{G}_{e_3}$.

Second, introduce necessary jumps between these sub-modes. Let's consider q_{31} in the running example, edges from q_{31} to the sub-modes of q_1 are introduced, i.e., including (q_{31}, q_{11}), (q_{31}, q_{12}) and (q_{31}, q_{13}).
Third, define a reset map for each introduced edge. Continue the above example, we have

$$\begin{aligned}
&\mathcal{R}^m_{(q_{31},q_{11})}(\mathbf{x}) \subseteq \mathcal{RA}_{in}(1,1), && \forall \mathbf{x} \in g_{31}\\
&\mathcal{R}^m_{(q_{31},q_{12})}(\mathbf{x}) \subseteq \mathcal{RA}_{in}(1,2), && \forall \mathbf{x} \in g_{31}\\
&\mathcal{R}^m_{(q_{31},q_{13})}(\mathbf{x}) \subseteq \mathcal{RA}_{in}(1,3), && \forall \mathbf{x} \in g_{31}\\
&\mathcal{R}^m_{(q_{13},q_{30})}(\mathbf{x}) \subseteq \mathcal{RA}_{in}(3,0), && \forall \mathbf{x} \in g_{13}\\
&\mathcal{R}^m_{(q_{13},q_{31})}(\mathbf{x}) \subseteq \mathcal{RA}_{in}(3,1), && \forall \mathbf{x} \in g_{13}\\
&\quad\quad\cdots
\end{aligned}$$

Step 2: Abstract away continuous dynamics in each resulted mode, and obtain a discrete directed graph (DDG). For the running example, it results a DDG given in Fig 8.

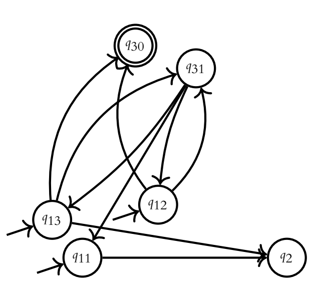

Figure 8. The resulting discrete directed graph

Step 3: Prune unsatisfied paths in the DDG, which are either unreachable like $\langle q_{14}, q_2\rangle$ and $\langle q_{31}, q_{12}, q_2\rangle$ or simple loops like $\langle q_{14}, q_{31}, q_{14}\rangle$. The DDG after pruning is depicted in Fig. 9, where only two edges (e_{11}, e_{12}) are left.

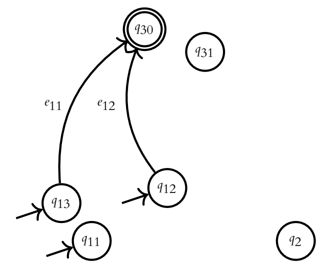

Figure 9. The discrete directed graph after edges prunning

Step 4: Synthesize a reset controller from the resulted DDG. Continue the running example, we obtain

$$\begin{aligned}
&\mathcal{R}^r_{e_2}(x) = R^m_{e_{12}}(\mathbf{x}), && \forall x \in \mathcal{G}^m_{e_{12}} = \mathcal{G}_{e_2} \setminus \mathcal{G}_{e_1}\\
&\mathcal{R}^r_{e_2}(x) = R^m_{e_{11}}(\mathbf{x}), && \forall x \in \mathcal{G}^m_{e_{11}} = \mathcal{G}_{e_2} \cap \mathcal{G}_{e_1}\\
&\mathtt{Init}^r_{q_1} = \mathtt{Init}^m_{q_{12}} \cup \mathtt{Init}^m_{q_{13}}.
\end{aligned}$$

Conclusion

In summary, we sketched our recent work on reset controller synthesis, including

- how to reduce the problem of synthesizing reset controllers w.r.t. safety and liveness constraints to reach-avoid set computation and differential invariant generation problems;
- how to inner-approximate reach-avoid sets by solving certain convex programming problems, which can be efficiently conducted using off-the-shell SDP solvers;
- how to synthesize reset controller for dHSs by reducing it into reach-avoid analysis for DDE and depth-first-search with block for discrete-event dynamics.

Regarding future work, we emphasize the following topics along this research line:

- To extend our approach to more general hybrid systems with more complicated vector fields, e.g., probabilistic and stochastic behavior, the combination of time-delay and stochasticity, and so on.
- To investigate potential correct-by-construction frameworks for HSs by taking feedback controller synthesis, switching logic controller synthesis, and reset controller synthesis into account uniformly.
- To integrate recent advances on differential invariant generation by reduction non-convex programming to SDP e.g. in (Q. Wang et al. 2021, 2022) into our synthesis framework.
- To conduct more complicated and practical case studies.

Acknowledgement

I would like to thank all collaborators involved in this research, including Jiang Liu, Yunjun Bai, Bai Xue, Jiyu Zhu, etc.

Funding Statement This work is partly funded by the NSFC under grant No. 62192730&62192732&62192735, and the CAS Project for Young Scientists in Basic Research under grant No. YSBR-040.

Conflicts of Interest None

Code & Data Availability All code can be found in GitHub: https://github.com/Han-SU/Reset-Controller-Synthesis.git.

Ethics Statement Ethical approval was obtained from the Ethics Committee of Peking University and University of Chinese Academy of Sciences. Study participants gave written informed consent to take part in the study.

Author Contribution Naijun Zhan, Mengfei Yang, and Bin Gu contributed the theoretical part, and Han Su contributed the experimental part of this paper.

Connections References Broman D, Woodcock J. What are the fundamental software abstractions for designing reliable cyber-physical systems operating in uncertain environments? Research Directions: Cyber-Physical Systems. 2023;1:e4. doi: 10.1017/cbp.2023.4

References

Abate, Alessandro, Iury Bessa, Dario Cattaruzza, Lucas Cordeiro, Cristina David, Pascal Kesseli, Daniel Kroening, and Elizabeth Polgreen. 2017. Automated formal synthesis of digital controllers for state-space physical plants. In *CAV'17,* 462–482. Springer.

Ames, Aaron, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. 2016. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control* 62 (8): 3861–3876.

Asarin, Eugene, Olivier Bournez, Thao Dang, Oded Maler, and Amir Pnueli. 2000. Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE* 88 (7): 1011–1025.

Back, Ralph-Johan, and Joakim Wright. 2012. *Refinement calculus: a systematic introduction.* Springer Science & Business Media.

Baheti, Radhakisan, and Helen Gill. 2011. Cyber-physical systems. *The Impact of Control Technology* 12 (1): 161–166.

Bai, Yunjun, Ting Gan, Li Jiao, Bican Xia, Bai Xue, and Naijun Zhan. 2021. Switching controller synthesis for delay hybrid systems under perturbations. In *HSCC'21,* 1–11.

Belta, Calin, Boyan Yordanov, and Ebru Aydin Gol. 2017. *Formal methods for discrete-time dynamical systems.* Vol. 89. Springer.

Bozga, Marius, and Joseph Sifakis. 2022. Correct by design coordination of autonomous driving systems. In *ISoLA'22,* 13–29. Springer.

Brentari, Mirko, Sofia Urbina, Denis Arzelier, Christophe Louembet, and Luca Zaccarian. 2018. A hybrid control framework for impulsive control of satellite rendezvous. *IEEE Transactions on Control Systems Technology* 27 (4): 1537–1551.

Butler, Michael J, Jean-Raymond Abrial, and Richard Banach. 2016. *Modelling and refining hybrid systems in event-b and rodin.*

Chen, Xin, Erika Ábrahám, and Sriram Sankaranarayanan. 2013. Flow*: an analyzer for non-linear hybrid systems. In *CAV'13,* 258–263. Springer.

Clegg, John C. 1958. A nonlinear integrator for servomechanisms. *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry* 77 (1): 41–42.

Coogan, Samuel, and Murat Arcak. 2012. Guard synthesis for safety of hybrid systems using sum of squares programming. In *CDC'12,* 6138–6143. IEEE.

Dupont, Guillaume, Yamine Ait-Ameur, Neeraj Kumar Singh, and Marc Pantel. 2021. Event-B hybridation: a proof and refinement-based framework for modelling hybrid systems. *ACM Transactions on Embedded Computing Systems* 20 (4): 1–37.

Dupont, Guillaume, Yamine Aït-Ameur, Neeraj Kumar Singh, and Marc Pantel. 2022. Formally verified architectural patterns of hybrid systems using proof and refinement with Event-B. *Science of Computer Programming* 216:102765.

Eggers, Andreas, Martin Fränzle, and Christian Herde. 2008. SAT modulo ODE: a direct SAT approach to hybrid systems. In *ATVA'08,* 171–185. Springer.

Frehse, Goran, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. SpaceEx: Scalable verification of hybrid systems. In *CAV'11,* 379–395. Springer.

Ghorbal, Khalil, and André Platzer. 2014. Characterizing algebraic invariants by differential radical invariants. In *TACAS'14,* 279–294. Springer.

Girard, Antoine. 2012. Controller synthesis for safety and reachability via approximate bisimulation. *Automatica* 48 (5): 947–953.

Gulwani, Sumit, and Ashish Tiwari. 2008. Constraint-based approach for analysis of hybrid systems. In *CAV'08',* 190–203. Springer.

Jha, Susmit, Sumit Gulwani, Sanjit A Seshia, and Ashish Tiwari. 2010. Synthesizing switching logic for safety and dwell-time requirements. In *ICCPS'10,* 22–31.

Kong, Soonho, Sicun Gao, Wei Chen, and Edmund Clarke. 2015. Dreach: δ-reachability analysis for hybrid systems. In *TACAS'15,* 200–205. Springer.

Liu, Jiang, Jidong Lv, Zhao Quan, Naijun Zhan, Hengjun Zhao, Chaochen Zhou, and Liang Zou. 2010. A calculus for hybrid csp. In *ASPLS'10,* 1–15. Springer.

Liu, Jiang, Han Su, Yunjun Bai, Bin Gu, Bai Xue, Mengfei Yang, and Naijun Zhan. 2023. *Correct-by-construction for hybrid systems by synthesizing reset controller.* arXiv: 2309.05906 `[eess.SY]`.

Liu, Jiang, Naijun Zhan, and Hengjun Zhao. 2011. Computing semi-algebraic invariants for polynomial dynamical systems. In *EMSOFT'11,* 97–106.

Loos, Sarah M, and André Platzer. 2016. Differential refinement logic. In *LICS'16,* 505–514.

Platzer, André. 2010. Differential-algebraic dynamic logic for differential-algebraic programs. *Journal of Logic and Computation* 20 (1): 309–352.

———. 2012. Logics of dynamical systems. In *LICS'12,* 13–24. IEEE.

Richard, Banach. 2024. Core hybrid Event-B III: fundamentals of a reasoning framework. *Science of Computer Programming* 231:103002.

Richard, Banach, Butler Michael, Shengchao Qin, Nitika Verma, and Huibiao Zhu. 2015. Core hybrid Event-B I: single hybrid Event-B machines. *Science of Computer Programming* 105:92–123.

Richard, Banach, Butler Michael, Shengchao Qin, and Huibiao Zhu. 2017. Core hybrid Event-B II: multiple cooperating hybrid Event-B machines. *Science of Computer Programming* 139:1–35.

Su, Han, Jiyu Zhu, Shenghua Feng, Yunjun Bai, Bin Gu, Jiang Liu, Mengfei Yang, and Naijun Zhan. 2023. *Reset controller synthesis by reach-avoid analysis for delay hybrid systems.* arXiv: 2309.05908 `[eess.SY]`.

Tabuada, Paulo. 2009. *Verification and control of hybrid systems: a symbolic approach.* Springer Science & Business Media.

Taly, Ankur, Sumit Gulwani, and Ashish Tiwari. 2011. Synthesizing switching logic using constraint solving. *International Journal on Software Tools for Technology Transfer* 13 (6): 519–535.

Taly, Ankur, and Ashish Tiwari. 2010. Switching logic synthesis for reachability. In *EMSOFT'10,* 19–28.

Tan, Yong Kiam, and André Platzer. 2019. An axiomatic approach to liveness for differential equations. In *FM'19,* 371–388. Springer.

———. 2021. Switched systems as hybrid programs. *IFAC-PapersOnLine* 54 (5): 247–252.

Tomlin, Claire J, John Lygeros, and S Shankar Sastry. 2000. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE* 88 (7): 949–970.

Wang, Qiuye, Mingshuai Chen, Bai Xue, Naijun Zhan, and Joost-Pieter Katoen. 2021. Synthesizing invariant barrier certificates via difference-of-convex programming. In *CAV'21,* 443–466. Springer.

———. 2022. Encoding inductive invariants as barrier certificates: synthesis via difference-of-convex programming. *Inf. and Comput.* 289 (Part): 104965.

Wang, Shuling, Zekun Ji, Xiong Xu, Bohua Zhan, Qiang Gao, and Naijun Zhan. 2024. Formally verified c code generation from hybrid communicating sequential processes. In *ICCPS'24,* 123–134. IEEE.

Wang, Shuling, Naijun Zhan, and Liang Zou. 2015. An improved HHL prover: an interactive theorem prover for hybrid systems. In *ICFEM'15,* 382–399. Springer.

Xue, Bai, Qiuye Wang, Naijun Zhan, and Martin Fränzle. 2019. Robust invariant sets generation for state-constrained perturbed polynomial systems. In *HSCC'19,* 128–137.

Yan, Gaogao, Li Jiao, Shuling Wang, Lingtai Wang, and Naijun Zhan. 2020. Automatically generating SystemC code from HCSP formal models. *ACM Transactions on Software Engineering and Methodology* 29 (1): 1–39.

Zhan, Naijun, Wang Shuling, and Hengjun Zhao. 2017. *Formal Verification of Simulink/Stateflow Diagrams: A deductive approach.* Springer, Cham.

Zhan, Naijun, Bohua Zhan, Shuling Wang, Dimitar Guelev, and Xiangyu Jin. 2023. *A generalized hybrid Hoare logic.* arXiv: 2303.15020 `[cs.LO]`.

Zhao, Hengjun, Naijun Zhan, and Deepak Kapur. 2013. Synthesizing switching controllers for hybrid systems by generating invariants. In *Theories of programming and formal methods - essays dedicated to jifeng he on the occasion of his 70th birthday,* 8051:354–373. LNCS.

Zhao, Pengcheng, Shankar Mohan, and Ram Vasudevan. 2019. Optimal control of polynomial hybrid systems via convex relaxations. *IEEE Transactions on Automatic Control* 65 (5): 2062–2077.