Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 54-5 (2021) 247–252

Switched Systems as Hybrid Programs [⋆]

Yong Kiam Tan André Platzer

Computer Science Department, Carnegie Mellon University, Pittsburgh, USA (e-mail: {yongkiat,aplatzer}@cs.cmu.edu)

Abstract: Real world systems of interest often feature interactions between discrete and continuous dynamics. Various hybrid system formalisms have been used to model and analyze this combination of dynamics, ranging from mathematical descriptions, e.g., using impulsive differential equations and switching, to automata-theoretic and language-based approaches. This paper bridges two such formalisms by showing how various classes of switched systems can be modeled using the language of hybrid programs from differential dynamic logic (dL). The resulting models enable the formal specification and verification of switched systems using dL and its existing deductive verification tools such as KeYmaera X. Switched systems also provide a natural avenue for the generalization of dL's deductive proof theory for differential equations. The completeness results for switched system invariants proved in this paper enable effective safety verification of those systems in dL.

Keywords: Hybrid and switched systems modeling · reachability analysis, verification and abstraction of hybrid systems · hybrid programs · differential dynamic logic

1. INTRODUCTION

The study of *hybrid systems*, i.e., mathematical models that combine discrete and continuous dynamics, is motivated by the need to understand the hybrid dynamics present in many real world systems (Liberzon, 2003; Platzer, 2018). Various formalisms can be used to describe hybrid systems, for example, impulsive differential equations (Haddad et al., 2006); switched systems (Liberzon, 2003; Sun and Ge, 2011); hybrid time combinations of discrete and continuous dynamics (Goebel et al., 2009, 2012); hybrid automata (Henzinger, 1996); and language-based models (Rönkkö et al., 2003; Liu et al., 2010; Platzer, 2010, 2018). These formalisms differ in their generality and in how the discrete-continuous dynamical combination is modeled, e.g., ranging from differential equations with discontinuous right-hand sides, to combinators that piece together discrete and continuous programs. Consequently, different formalisms may be better suited for different hybrid system applications and it is worthwhile to explore connections between different formalisms in order to exploit their various strengths for a given application.

A *switched system* consists of a family of continuous ordinary differential equations (ODEs) together with a discrete switching signal that prescribes the active ODE the system follows at each time. These models are commonly found in control designs where appropriately designed switching can be used to achieve control goals that cannot be achieved by purely continuous means (Liberzon, 2003).

Differential dynamic logic (dL) (Platzer, 2010, 2018) provides the language of *hybrid programs*, whose hybrid dynamics arise from combining discrete programming constructs with continuous ODEs. This combination yields a rich and flexible language for describing hybrid systems, e.g., with event- or time-triggered design paradigms.

This paper shows how various classes of switched systems can be fruitfully modeled in the language of hybrid programs. The contributions are as follows:

(1) Important classes of switched systems are modeled as hybrid programs in Sections 3–4. Subtleties associated with those models are investigated, along with methods for detecting and avoiding those pitfalls.
(2) Completeness results for differential equation invariants in dL (Platzer and Tan, 2020) are extended to invariants of switched systems, yielding an effective technique for proving switched system safety.

These contributions enable sound deductive verification of switched systems in dL and they lay the groundwork for further development of proof automation for switched systems, such as in the KeYmaera X (Fulton et al., 2015) hybrid systems prover based on dL. To demonstrate the versatility of the proposed hybrid program models, Section 5 uses KeYmaera X to formally verify stability for several switched system examples using standard Lyapunov function techniques (Liberzon, 2003). All proofs are available in the supplement (Tan and Platzer, 2021b).

2. BACKGROUND

This section informally recalls differential dynamic logic (dL) and the language of hybrid programs used to model switched systems in Sections 3 and 4. Formal presentations of dL are available elsewhere (Platzer, 2010, 2017, 2018).

2.1 Hybrid Programs

The language of *hybrid programs* is generated by the following grammar, where x is a variable, e is a dL term,

⋆ This research was sponsored by the AFOSR under grant number FA9550-16-1-0288. The first author was also supported by A*STAR, Singapore.

e.g., a polynomial over x, and Q is a dL formula.

$$\alpha, \beta \ ::= \ x := e \mid ?Q \mid x' = f(x) \,\&\, Q \mid \alpha;\beta \mid \alpha \cup \beta \mid \alpha^*$$

Discrete assignment $x := e$ sets the value of variable x to that of term e in the current state. Test $?Q$ checks that formula Q is true in the current state and aborts the run otherwise. The continuous program $x' = f(x) \,\&\, Q$ continuously evolves the system state by following the ODE $x' = f(x)$ for a nondeterministically chosen duration $t \geq 0$, as long as the system remains in the domain constraint Q for all times $0 \leq \tau \leq t$. The sequence program $\alpha;\beta$ runs program β after α, the choice program $\alpha \cup \beta$ nondeterministically chooses to run either α or β, and the loop program α^* repeats α for $n \in \mathbb{N}$ iterations where n is chosen nondeterministically. The nondeterminism inherent in hybrid programs is useful for abstractly modeling real world behaviors (Platzer, 2018). The evolution of various hybrid programs is illustrated in parts A–C and G of Fig. 1.

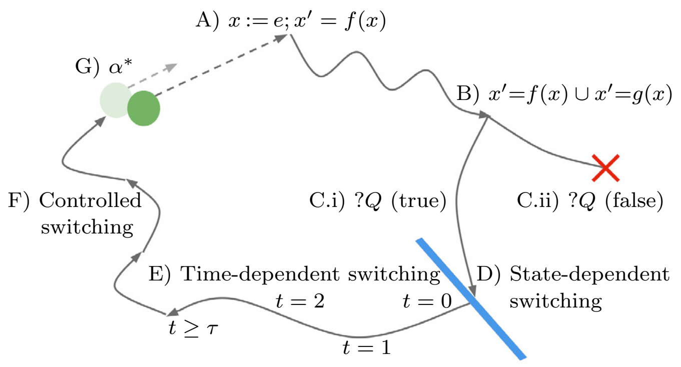


Fig. 1. The green initial state evolving according to a hybrid program featuring (clockwise from top):
A a discrete assignment (dashed line) followed sequentially by continuous ODE evolution (solid line),
B a choice between two ODEs (Section 3.1),
C a test that aborts (red $\times$) system evolutions leaving Q,
D switching when the system state crosses the thick blue switching surface (Section 3.2),
E switching after time $t \geq \tau$ has elapsed (Section 4.1),
F switching control that is designed to drive the system state close to its initial position (Section 4.2), and
G a loop that repeats system evolution (in lighter colors).

Notationally, $x = (x_1, \dots, x_n)$ are the state variables of an n-dimensional system, so $x' = f(x) \,\&\, Q$ is an autonomous n-dimensional system of ordinary differential equations over x; the ODE is written as $x' = f(x)$ when there is no domain constraint, i.e., $Q \equiv \mathit{true}$. For simplicity, all ODEs have polynomial right-hand sides, dL terms e are polynomial over x, and P, Q are formulas of first-order real arithmetic over x; extensions of the term language to Noetherian functions are described in Platzer and Tan (2020). The single-sided conditional `if` is defined as $\texttt{if}(P)\{\alpha\} \equiv (?P;\alpha) \cup (?\neg P)$. Nondeterministic choice over a finite family of hybrid programs α_p for $p \in \mathcal{P}$, $\mathcal{P} \equiv \{1, \dots, m\}$ is denoted $\bigcup_{p \in \mathcal{P}} \alpha_p \equiv \alpha_1 \cup \alpha_2 \cup \ldots \cup \alpha_m$.

The formula language of dL extends first-order logic formulas with dynamic modalities for specifying properties of a hybrid program α (Platzer, 2017, 2018). The box modality formula $[\alpha]P$ says that formula P is true for *all* states reachable by following the nondeterministic evolutions of hybrid program α, while the diamond modality formula $\langle\alpha\rangle P$ says that formula P is true for *some* reachable state of α. This paper focuses on using box modality formulas for specifying safety properties of hybrid programs. For example, formula $R \rightarrow [\alpha^*]P$ says that initial states satisfying precondition R remain in the safe region P after any number of runs of the loop α^*. A key technique for proving safety properties of such a loop is to identify an *invariant* I of α such that formula $I \rightarrow [\alpha]I$ is *valid*, i.e., true in all states (Platzer, 2018). To enable effective proofs of safety, invariance, and various other properties of interest, dL provides compositional reasoning principles for hybrid programs (Platzer, 2017, 2018) and a complete axiomatization for ODE invariants (Platzer and Tan, 2020).

2.2 Switched Systems

A *switched system* is described by the following data:

(1) an open, connected set $D \subseteq \mathbb{R}^n$ which is the *state space* of interest for the system,
(2) a finite (non-empty) family $\mathcal{P}$ of ODEs $x' = f_p(x)$ for $p \in \mathcal{P}$, and,
(3) for each initial state $\omega \in D$, a set of *switching signals* $\sigma : [0, \infty) \rightarrow \mathcal{P}$ prescribing the ODE $x' = f_{\sigma(t)}(x)$ to follow at time t for the system's evolution from ω. [1]

Switching phenomena can either be described explicitly as a function of time, or implicitly, e.g., as a state predicate, depending on the real world switching mechanism being modeled. Several standard classes of switching mechanisms are studied in Sections 3 and 4, following the nomenclature from Liberzon (2003). These switching mechanisms are illustrated in parts D–F of Fig. 1.

For simplicity, this paper assumes that the state space is $D = \mathbb{R}^n$. More general definitions of switched systems are possible but are left out of scope, see Liberzon (2003). For example, $\mathcal{P}$ can more generally be an (uncountably) infinite family and some switched systems may have *impulse effects* where the system state is allowed to make instantaneous, discontinuous jumps during the system's evolution, such as the dashed jump in part A of Fig. 1.

3. ARBITRARY AND STATE-DEPENDENT SWITCHING

3.1 Arbitrary Switching

Real world systems can exhibit switching mechanisms that are uncontrolled, *a priori* unknown, or too complicated to describe succinctly in a model. For example, a driving vehicle may encounter several different road conditions depending on the time of day, weather, and other unpredictable factors—given the multitude of combinations to consider, it is desirable to have a single model that exhibits and switches between all of those road conditions. *Arbitrary switching* is a useful paradigm for such systems because it considers *all* possible switching signals and their corresponding system evolutions. The arbitrary switching

[1] A more precise definition is given in the supplement (Tan and Platzer, 2021b), where the switching signals σ are also required to be well-defined (Liberzon, 2003; Sun and Ge, 2011) so that they model physically realizable switching.

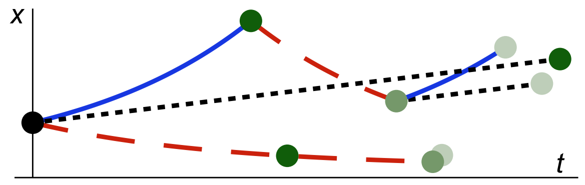


Fig. 2. Evolution of $\alpha_{\texttt{arb}}$ for $x' = x$ (solid blue), $x' = 1$ (dotted black), and $x' = -x$ (dashed red) from the initial state (black circle). Switching steps are marked by green circles and faded colors illustrate progression in loop iterations for the loop operator in $\alpha_{\texttt{arb}}$.

mechanism is modeled by the following hybrid program and illustrated in Fig. 2.

$$\alpha_{\texttt{arb}} \equiv \Big(\bigcup_{p\in\mathcal{P}} x' = f_p(x)\Big)^*$$

Observe that *i)* the system nondeterministically chooses which ODE to follow at each loop iteration; *ii)* it follows the chosen ODE for a nondeterministic duration; *iii)* each loop iteration corresponds to a switching step and the loop repeats for a finite, nondeterministically chosen number of iterations. Two subtle behaviors are illustrated by the bottom trajectory in Fig. 2: $\alpha_{\texttt{arb}}$ can switch to the same ODE across a loop iteration or it can *chatter* by making several discrete switches without continuously evolving its state between those switches (Sogokon et al., 2017). These behaviors are harmless for safety verification because they do not change the set of reachable states of the switched system. Formally, the adequacy of $\alpha_{\texttt{arb}}$ as a model of arbitrary switching is shown in the following proposition.

Proposition 1. A state is reachable by hybrid program $\alpha_{\texttt{arb}}$ iff it is reachable in finite time by a switched system $x' = f_p(x)$ for $p \in \mathcal{P}$ following a switching signal σ.

By Proposition 1, the dL formula $[\alpha_{\texttt{arb}}]P$ specifies safety for arbitrary switching, i.e., for any switching signal σ, the system states reached at all times by switching according to σ satisfy the safety postcondition P.

3.2 State-Dependent Switching

Arbitrary switching can be constrained by enabling switching to the ODE $x' = f_p(x)$ only when the system state belongs to a corresponding domain specified by formula Q_p. This yields the *state-dependent switching* paradigm, which is useful for modeling real systems that are either known or designed to have particular switching surfaces. For the finite family of ODEs with domains $x' = f_p(x) \,\&\, Q_p$, $p \in \mathcal{P}$, state-dependent switching is modeled as follows:

$$\alpha_{\texttt{state}} \equiv \Big(\bigcup_{p\in\mathcal{P}} x' = f_p(x) \,\&\, Q_p\Big)^*$$

Operationally, if the system is currently evolving in domain Q_i and is about to leave the domain, it must switch to another ODE with domain Q_j that is true in the current state to continue its evolution. Arbitrary switching $\alpha_{\texttt{arb}}$ is the special case of $\alpha_{\texttt{state}}$ with no domain restrictions. The following result generalizes Proposition 1 to consider only states reached while obeying the specified domains.

Proposition 2. A state is reachable by hybrid program $\alpha_{\texttt{state}}$ iff it is reachable in finite time by a switched system $x' = f_p(x)$ for $p \in \mathcal{P}$ following a switching signal σ while obeying the domains Q_p.

The next two results are syntactically provable in dL and they provide sound and complete invariance reasoning principles for state-dependent (and arbitrary) switching. Formula ϕ is *computable* from a set of inputs iff there is an algorithm that outputs ϕ when given those inputs.

Lemma 3. Formula I is an invariant for $\alpha_{\texttt{state}}$ iff I is invariant for all constituent ODEs $x' = f_p(x) \,\&\, Q_p$, $p \in \mathcal{P}$.

Theorem 4. From input ODEs $x' = f_p(x) \,\&\, Q_p$, $p \in \mathcal{P}$ and formula I, there is a computable formula of real arithmetic ϕ such that formula I is invariant for $\alpha_{\texttt{state}}$ iff ϕ is valid. In particular, invariance for $\alpha_{\texttt{state}}$ is decidable.

Lemma 3 shows that when searching for an invariant of $\alpha_{\texttt{state}}$, it suffices to search for a *common* invariant of every constituent ODE. Theorem 4 enables sound and complete invariance proofs for systems with state-dependent switching in dL, relying on dL's complete axiomatization for ODE invariance and decidability of first-order real arithmetic over polynomial terms (Tarski, 1951). These results also extend to Noetherian functions, e.g., exponentials and trigonometric functions, at the cost of losing decidability of the resulting arithmetic (Platzer and Tan, 2020).

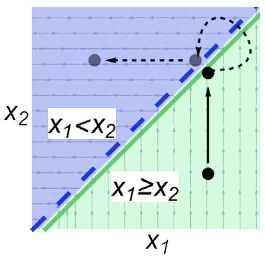
3.3 Modeling Subtleties

The model $\alpha_{\texttt{state}}$ as defined above makes no *a priori* assumptions about how the ODEs and their domains $x' = f_p(x) \,\&\, Q_p$ are designed, so results like Theorem 4 apply generally to all state-dependent switching designs. However, state-dependent switching can exhibit some well-known subtleties (Liberzon, 2003; Sogokon et al., 2017) and it becomes the onus of modelers to appropriately account for these subtleties. This section examines various subtleties that can arise in $\alpha_{\texttt{state}}$ and prescribes sufficient arithmetical criteria for avoiding them; like Theorem 4, these arithmetical criteria are decidable for systems with polynomial terms (Tarski, 1951). As a running example, let the line $x_1 = x_2$ be a *switching surface*, i.e., the example systems described below are intended to exhibit switching when their system state reaches this line.

Well-defined switching. First, observe that the domains Q_p must cover the entire state space; otherwise, there would be system states of interest where no continuous dynamics is active. This can be formally guaranteed by deciding validity of the formula ①: $\bigvee_{p\in\mathcal{P}} Q_p$. Next, consider the following ODEs:

$$\underbrace{x_1' = 0, x_2' = 1 \,\&\, x_1 \geq x_2}_{x' = f_A(x) \,\&\, Q_A}$$

$$\underbrace{x_1' = -1, x_2' = 0 \,\&\, x_1 < x_2}_{x' = f_B(x) \,\&\, Q_B}$$



Consider the system evolution starting in $Q_A \equiv x_1 \geq x_2$ illustrated above on the right. When the system reaches $x_1 = x_2$ (the illustration is offset for clarity), it is about to *locally progress* into $Q_B \equiv x_1 < x_2$ by switching to ODE $x' = f_B(x)$ but it gets stuck because it cannot make the infinitesimal jump from Q_A to enter Q_B; augmenting domain Q_B to $x_1 \leq x_2$ enables the switch. More generally, to avoid the need for infinitesimal jumps, domains Q_p should be augmented to include states that locally progress into Q_p under the ODE $x' = f_p(x)$ and, symmetrically, states that locally exit Q_p (Sogokon et al., 2017). Local progress (and exit) for ODEs is characterized as follows.

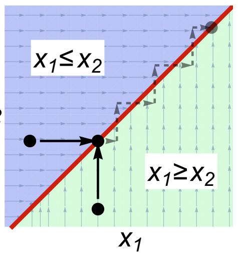
Theorem 5. (Platzer and Tan (2020)). From input ODE $x' = f(x) \,\&\, Q$, there are computable formulas of real arithmetic $(\dot{Q})^{(*)}_f$, $(\dot{Q})^{(*)}_{-f}$ that respectively characterize the states from which $x' = f(x)$ locally progresses into Q and those from which it locally exits Q.

By Theorem 5, to avoid the stuck states exemplified above for ODEs $x' = f_p(x) \,\&\, Q_p$, $p \in \mathcal{P}$ in $\alpha_{\mathtt{state}}$, it suffices to decide validity of the formula ②: $(\dot{Q}_p)^{(*)}_{f_p} \lor (\dot{Q}_p)^{(*)}_{-f_p} \to Q_p$ for each $p \in \mathcal{P}$. Condition ② is syntactically significantly simpler but equivalent to the domain augmentation presented in Sogokon et al. (2017) for piecewise continuous models, a form of state-dependent switching.

Sliding modes. The preceding subtlety arose from incomplete domain constraint specifications. Another subtlety that can arise because of incomplete specification of ODE dynamics is exemplified by the following ODEs:

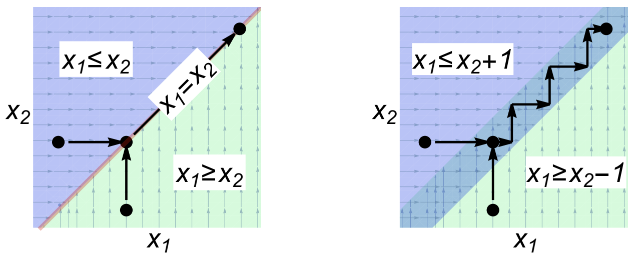
$$\underbrace{x_1' = 0, x_2' = 1 \,\&\, x_1 \geq x_2}_{x'=f_A(x) \,\&\, Q_A}$$

$$\underbrace{x_1' = 1, x_2' = 0 \,\&\, x_1 \leq x_2}_{x'=f_B(x) \,\&\, Q_B}$$



Systems starting in $Q_A \equiv x_1 \geq x_2$ or $Q_B \equiv x_1 \leq x_2$ eventually reach the line $x_1 = x_2$ but they then get stuck because the ODEs on either side of $x_1 = x_2$ drive system evolution onto the line. Mathematically, the system enters a *sliding mode* (Liberzon, 2003) along $x_1 = x_2$; as illustrated above, this can be thought of as infinitely fast switching between the ODEs that results in a new sliding dynamics *along* the switching surface $x_1 = x_2$.

When the sliding dynamics can be calculated exactly, it suffices to add those dynamics to the switched system, e.g., adding the sliding dynamics $x_1' = \frac{1}{2}, x_2' = \frac{1}{2} \,\&\, x_1 = x_2$ to the example above allows stuck system states on $x_1 = x_2$ to continuously progress along the line (illustrated below, left). An alternative is *hysteresis switching* (Liberzon, 2003) which enlarges domains adjacent to the sliding mode so that a system that reaches the sliding surface is allowed to briefly continue following its current dynamics before switching. For example, for a fixed $\varepsilon > 0$, the enlarged domains $Q_A \equiv x_1 \geq x_2 - \varepsilon$ and $Q_B \equiv x_1 \leq x_2 + \varepsilon$ allows the stuck states to evolve off the line for a short distance. This yields arbitrary switching in the overlapped part of both domains (illustrated below, right). For a family of domains Q_p, $p \in \mathcal{P}$ meeting conditions ① and ②, hysteresis switching can be introduced by replacing each Q_p with its closed ε-neighborhood for some chosen $\varepsilon > 0$.



To guarantee the absence of stuck states, by Theorem 5, it suffices to decide validity of the formula ③: $\bigvee_{p \in \mathcal{P}} (\dot{Q}_p)^{(*)}_{f_p}$, i.e., every point in the state space can switch to an ODE which locally progresses in its associated domain. Models meeting conditions ② and ③ also meet condition ①.

Zeno behavior. Hybrid and switched system models can also exhibit *Zeno behavior*, where the model makes infinitely many discrete transitions in a finite time interval (Zhang et al., 2001). Such behaviors are an artifact of the model and are not reflective of the real world. Zeno traces are typically excluded when reasoning about hybrid system models (Zhang et al., 2001), e.g., Proposition 2 specifies safety for all *finite* (thus non-Zeno) executions of state-dependent switching. The detection of Zeno behavior in switched systems is left out of scope for this paper.

4. TIME-DEPENDENT AND CONTROLLED SWITCHING

4.1 Time-Dependent Switching

The *time-dependent switching* paradigm imposes timing constraints on switching signals. To specify such constraints syntactically, each ODE in the family $p \in \mathcal{P}$ is extended with a common, fresh clock variable t with $t' = 1$ yielding ODEs of the form $x' = f_p(x), t' = 1$, and a fresh (discrete) flag variable u is used to select and track the ODE to follow at each time. One form of timing constraint is *slow switching*, where the system switches arbitrarily between ODEs but must spend a minimum *dwell time* $\tau > 0$ between each switch. Sufficiently large dwell times can be used to stabilize some systems (see Section 5). Slow switching is modeled by the following hybrid program:

$$\alpha_{\mathtt{slow}} \equiv \alpha_r; \Big(\mathtt{if}(t \geq \tau)\{\alpha_r\}; \bigcup_{p \in \mathcal{P}} (?u{=}p; x'{=}f_p(x), t'{=}1)\Big)^*$$

$$\alpha_r \equiv \; t := 0; \bigcup_{p \in \mathcal{P}} u := p$$

The program α_r resets the clock t to 0 and nondeterministically chooses a new value for the flag u. For each loop iteration of $\alpha_{\mathtt{slow}}$, the guard $t \geq \tau$ checks if the current ODE has executed for at least time τ before running α_r to pick a new value for u. The subsequent choice selects the ODE to follow based on the value of flag u.

Proposition 6. A state is reachable by hybrid program $\alpha_{\mathtt{slow}}$ iff it is reachable in finite time by a switched system $x' = f_p(x)$ for $p \in \mathcal{P}$ following a switching signal σ that spends at least time τ between its switching times.

Theorem 7. From input ODEs $x' = f_p(x)$, $p \in \mathcal{P}$ and formula I, there is a computable formula of real arithmetic ϕ such that formula I is invariant for $\alpha_{\mathtt{slow}}$ iff ϕ is valid. In particular, invariance for $\alpha_{\mathtt{slow}}$ is decidable.

4.2 Controlled Switching

The discrete fragment of hybrid programs can be used to flexibly model (computable) *controlled switching* mechanisms, e.g., those that combine state-dependent and time-dependent switching constraints, or make complex switching decisions based on the state of the system. An abstract controlled switching model is shown below, where program α_i initializes the system state (e.g., of the clock or flag) and α_u models a controller that assigns a decision $u := p$.

$$\alpha_{\texttt{ctrl}} \equiv \alpha_i; \Big(\alpha_u; \bigcup_{p\in\mathcal{P}} (?u = p; x' = f_p(x), t' = 1 \,\&\, Q_p)\Big)^*$$

Hybrid program $\alpha_{\texttt{ctrl}}$ resembles the shape of standard models of event-triggered and time-triggered systems in dL (Platzer, 2018) but is adapted for controlled switching. The controller program α_u inspects the current state variables x and the clock t. It can modify the clock, e.g., by resetting it with $t := 0$, but α_u must not discretely change the state variables x. The subsequent choice selects the ODE to follow based on the value of flag u assigned in α_u.

The slow switching model $\alpha_{\texttt{slow}}$ is an instance of $\alpha_{\texttt{ctrl}}$ where the controller program switches only after the dwell time is exceeded. Another example is *periodic switching*, where the controller periodically cycles through a family of ODEs. Switching with sufficiently fast period can be used to stabilize a family of unstable ODEs, e.g., for linear ODEs whose system matrices have a stable convex combination (Tokarzewski, 1987). Without loss of generality, assume that $\mathcal{P} \equiv \{1, \dots, m\}$, the desired switching order is $1, \dots, m$, and the periodic signal is required to follow the i-th ODE for exactly time $\zeta_i > 0$. Periodic fast switching is modeled as an instance of $\alpha_{\texttt{ctrl}}$ as follows:

$$\alpha_{\texttt{fast}} \equiv \alpha_{\texttt{ctrl}} \text{ where } \alpha_i \equiv \ t := 0; u := 1, Q_p \equiv t \le \zeta_p, \text{ and}$$

$$\alpha_u \equiv \bigcup_{p\in\mathcal{P}} \texttt{if}(u = p \land t = \zeta_p) \left\{ \begin{array}{l} t := 0; u := u + 1; \\ \texttt{if}(u > m)\{u := 1\} \end{array} \right\}$$

The system is initialized with $t = 0$, $u = 1$ at the start of the cycle. The controller program α_u then deterministically cycles through $u = 1, \dots, m$ by discretely incrementing the flag variable whenever the time limit ζ_p for the currently chosen ODE is reached. The domain constraints Q_p respectively limit each ODE to run for at most time ζ_p as prescribed for the switched system.

Proposition 8. A state is reachable by hybrid program $\alpha_{\texttt{fast}}$ iff it is reachable in finite time by a switched system $x' = f_p(x)$ for $p \in \{1, \dots, m\}$ following the switching signal σ that periodically switches in the order $1, \dots, m$ according to the times $\zeta_1, \zeta_2, \dots, \zeta_m$ respectively.

A subtlety occurs in $\alpha_{\texttt{fast}}$ and Proposition 8 when one of the constituent ODEs exhibits *finite time blowup* before reaching its switching time, e.g., consider switching between ODEs $x' = 1$ and $x' = x^2$ with times $\zeta_1 = \zeta_2 = 1$ starting from a state where $x = 0$; the latter ODE blows up in the first cycle. Mathematically, the switching signal σ is simply ignored after the blowup time, but such blowup phenomena may not accurately reflect real world behavior. Global existence of solutions for all ODEs in the switched system can be verified in dL (Tan and Platzer, 2021a).

5. STABILITY VERIFICATION IN KEYMAERA X

This section shows how stability can be formally verified in dL using the KeYmaera X theorem prover[2] (Fulton et al., 2015) for the switched systems modeled by $\alpha \in \{\alpha_{\texttt{arb}}, \alpha_{\texttt{state}}, \alpha_{\texttt{slow}}\}$. For these systems, the origin $0 \in \mathbb{R}^n$ is *stable* iff the following formula is valid:

$$\forall \varepsilon > 0\, \exists \delta > 0\, \forall x\, (\|x\|^2 < \delta^2 \to [\alpha]\, \|x\|^2 < \varepsilon^2)$$

This formula expresses that, for initial states sufficiently close to the origin ($\|x\|^2 < \delta^2$ for $\delta > 0$), all states reached by hybrid program α from those states remain close to the origin ($\|x\|^2 < \varepsilon^2$ for $\varepsilon > 0$). By Propositions 1, 2, and 6, the formula specifies stability for the switched systems modeled by $\alpha \in \{\alpha_{\texttt{arb}}, \alpha_{\texttt{state}}, \alpha_{\texttt{slow}}\}$ uniformly in their respective sets of switching signals (Liberzon, 2003).

Unlike invariance, a switched system can be stable (resp. unstable) even if all of its constituent ODEs are unstable (resp. stable), depending on the switching mechanism (Liberzon, 2003). Stability verification for such systems is important because it provides formal guarantees that specific switching designs correctly eliminate potential instabilities in systems of interest. An important technique for proving stability for ODEs and switched systems is to design an appropriate *Lyapunov function*, i.e., an auxiliary energy measure that is non-increasing along all system trajectories (Liapounoff, 1907; Liberzon, 2003).

Example 9. Consider arbitrary switching $\alpha_{\texttt{arb}}$ with ODEs:

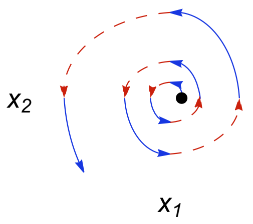
$$x_1' = -x_1 + x_2^3, x_2' = -x_1 - x_2$$
$$x_1' = -x_1, x_2' = -x_2$$

Both ODEs are stable and share the common Lyapunov function $v = \frac{x_1^2}{2} + \frac{x_2^4}{4}$. To prove stability for this example, the key idea is to show that $v < k \land x_1^2 + x_2^2 < \varepsilon$ is a loop invariant of $\alpha_{\texttt{arb}}$, where k is an upper bound on the initial value of v close to the origin.

Example 10. The following ODEs Ⓐ and Ⓑ are individually stable (Liberzon, 2003, Example 3.1). However, as illustrated below on the right, there is a switching signal that causes the system to diverge from the origin, i.e., these ODEs are *not* stable under arbitrary switching.

$$\underbrace{x_1' = -\frac{x_1}{8} - x_2, x_2' = 2x_1 - \frac{x_2}{8}}_{Ⓐ \text{ (solid blue)}}$$

$$\underbrace{x_1' = -\frac{x_1}{8} - 2x_2, x_2' = x_1 - \frac{x_2}{8}}_{Ⓑ \text{ (dashed red)}}$$



Stability can be achieved by a state-dependent switching design with domains: Ⓐ $x_1x_2 \le 0$ and Ⓑ $x_1x_2 \ge 0$. The resulting system modeled by $\alpha_{\texttt{state}}$ has the common Lyapunov function $v = x_1^2 + x_2^2$. The proof uses a loop invariant similar to Example 9 and, crucially, checks the arithmetical Lyapunov function conditions for the derivative of v only on the respective domains for each ODE.

[2] All examples are formalized in KeYmaera X 4.9.2 at: `https://github.com/LS-Lab/KeYmaeraX-projects/blob/master/stability/switchedsystems.kyx`.

Example 11. The example ODEs Ⓐ, Ⓑ can also be stabilized by sufficiently slow switching in $\alpha_{\mathtt{slow}}$ with minimum dwell time $\tau = 3$ (the value of τ can be further optimized). Here, two different Lyapunov functions are used: Ⓐ $2x_1^2+x_2^2$ and Ⓑ $x_1^2+2x_2^2$. The key proof idea is to bound both Lyapunov functions by decaying exponentials, and show that the dwell time τ is sufficiently large to ensure that both Lyapunov functions have decayed by an appropriate fraction when a switch occurs at time $t \geq \tau$.

The minimum dwell time principle can be used more generally to stabilize any family of stable linear ODEs (Liberzon, 2003). For example, the ODE Ⓒ $x_1' = -x_1, x_2' = -x_2$ is also stable and has the Lyapunov function $x_1^2 + x_2^2$. All three ODEs Ⓐ, Ⓑ, Ⓒ can be stabilized with the same dwell time $\tau = 3$. The KeYmaera X proof required minimal changes, e.g., the loop invariants were updated to account for the new ODE Ⓒ and its Lyapunov function.

6. RELATED WORK

There are numerous hybrid system formalisms in the literature (Haddad et al., 2006; Liberzon, 2003; Sun and Ge, 2011; Goebel et al., 2009, 2012; Henzinger, 1996; Rönkkö et al., 2003; Liu et al., 2010; Platzer, 2010, 2018); see the cited articles and textbooks for further references.

Connections between several formalisms have been examined in prior work. Platzer (2010) shows how hybrid automata can be embedded into hybrid programs for their safety verification; the book also generalizes dL with (disjunctive) differential-algebraic constraints that can be used to model and verify continuous dynamics with state-dependent switching (Platzer, 2010, Chapter 3). This paper instead models switching with discrete program operators which enables compositional reasoning for the hybrid dynamics in switched systems. Sogokon et al. (2017) study hybrid automata models for ODEs with piecewise continuous right-hand sides and highlight various subtleties in the resulting models; similar subtleties for state-dependent switching models are presented in Section 3.3. Goebel et al. (2009, 2012) show how impulsive differential equations, hybrid automata, and switched systems can all be understood as hybrid time models, and derive their properties using this connection; Theorems 4 and 7 are proved for switched systems using their hybrid program models.

7. CONCLUSION

This paper provides a blueprint for developing and verifying hybrid program models of switched systems. These contributions enable several future directions, including: *i)* formalizing *asymptotic stability* for switched systems (Liberzon, 2003; Sun and Ge, 2011), i.e., the systems are stable (Section 5) *and* their trajectories tend to the origin over time; *ii)* modeling switched systems under more general continuous dynamics, e.g., differential inclusions (Goebel et al., 2012) or differential-algebraic constraints (Platzer, 2010); *iii)* developing practical proof automation for switched systems in KeYmaera X, e.g., automated synthesis and verification of invariants and Lyapunov functions for various switching mechanisms.

Acknowledgments. We thank the ADHS'21 anonymous reviewers for their helpful feedback on this paper.

REFERENCES

Fulton, N., Mitsch, S., Quesel, J., Völp, M., and Platzer, A. (2015). KeYmaera X: an axiomatic tactical theorem prover for hybrid systems. In A.P. Felty and A. Middeldorp (eds.), *CADE*, volume 9195 of *LNCS*, 527–538. Springer, Cham.

Goebel, R., Sanfelice, R.G., and Teel, A.R. (2009). Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2), 28–93.

Goebel, R., Sanfelice, R.G., and Teel, A.R. (2012). *Hybrid Dynamical Systems: Modeling, Stability, and Robustness.* Princeton University Press.

Haddad, W.M., Chellaboina, V., and Nersesov, S.G. (2006). *Impulsive and Hybrid Dynamical Systems: Stability, Dissipativity, and Control.* Princeton University Press.

Henzinger, T.A. (1996). The theory of hybrid automata. In *LICS*, 278–292. IEEE Computer Society.

Liapounoff, A. (1907). Probléme général de la stabilité du mouvement. *Annales de la Faculté des sciences de Toulouse : Mathématiques*, 9, 203–474.

Liberzon, D. (2003). *Switching in Systems and Control.* Systems & Control: Foundations & Applications. Birkhäuser.

Liu, J., Lv, J., Quan, Z., Zhan, N., Zhao, H., Zhou, C., and Zou, L. (2010). A calculus for hybrid CSP. In K. Ueda (ed.), *APLAS*, volume 6461 of *LNCS*, 1–15. Springer.

Platzer, A. (2010). *Logical Analysis of Hybrid Systems - Proving Theorems for Complex Dynamics.* Springer.

Platzer, A. (2017). A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reasoning*, 59(2), 219–265.

Platzer, A. (2018). *Logical Foundations of Cyber-Physical Systems.* Springer.

Platzer, A. and Tan, Y.K. (2020). Differential equation invariance axiomatization. *J. ACM*, 67(1), 6:1–6:66.

Rönkkö, M., Ravn, A.P., and Sere, K. (2003). Hybrid action systems. *Theor. Comput. Sci.*, 290(1), 937–973.

Sogokon, A., Ghorbal, K., and Johnson, T.T. (2017). Operational models for piecewise-smooth systems. *ACM Trans. Embed. Comput. Syst.*, 16(5s), 185:1–185:19.

Sun, Z. and Ge, S.S. (2011). *Stability Theory of Switched Dynamical Systems.* Communications and Control Engineering. Springer.

Tan, Y.K. and Platzer, A. (2021a). An axiomatic approach to existence and liveness for differential equations. *Formal Aspects Comput.*

Tan, Y.K. and Platzer, A. (2021b). Switched systems as hybrid programs. *CoRR*, abs/2101.06195. URL `https://arxiv.org/abs/2101.06195`.

Tarski, A. (1951). *A Decision Method for Elementary Algebra and Geometry.* RAND Corporation, Santa Monica, CA.

Tokarzewski, J. (1987). Stability of periodically switched linear systems and the switching frequency. *International Journal of Systems Science*, 18(4), 697–726.

Zhang, J., Johansson, K.H., Lygeros, J., and Sastry, S. (2001). Zeno hybrid systems. *Int. J. Robust Nonlinear Control.*, 11(5), 435–451.