Synthesis of P-stable Abstractions

Anna Becchi[1], Alessandro Cimatti[1], and Enea Zaffanella[2]

[1] Fondazione Bruno Kessler, Italy {abecchi,cimatti}@fbk.eu
[2] University of Parma, Italy enea.zaffanella@unipr.it

Abstract. Stability is a fundamental requirement of dynamical systems. Most of the works concentrate on verifying stability for a given stability region. In this paper we tackle the problem of *synthesizing* $\mathbb{P}$*-stable abstractions*. Intuitively, the $\mathbb{P}$-stable abstraction of an open dynamical system characterizes the transitions between stability regions in response to external inputs. The stability regions are not given - rather, they are synthesized as the tightest representation with respect to a given set of relevant predicates $\mathbb{P}$. A $\mathbb{P}$-stable abstraction is enriched by timing information derived from the duration of stabilization.
We implement a synthesis algorithm in the framework of Abstract Interpretation, that allows different degrees of approximation. We show the representational power of $\mathbb{P}$-stable abstractions, that provide a high-level account of the behavior of the system with respect to stability, and we experimentally evaluate the effectiveness of a compositional approach, that allows synthesizing $\mathbb{P}$-stable abstractions for significant systems.

1 Introduction

Context. Reactive systems are often designed to operate in some stable condition (in absence of external stimuli), and to reach a possibly different stable condition (in response to some external stimulus). Stability may be reached after variable amounts of time, possibly depending on the physical dynamics being controlled. Notable examples are HVAC systems and relay-based circuits, built out of electromechanical components, pervasively adopted in the railways domain for the control of stations.

System stability is hard to assess. Stability is not to be confused with a completely still situation (i.e. a zero-derivative point) and partly oscillating or limit behaviors may be considered stable. Furthermore, a system may exhibit a large number of stable conditions, which may be difficult to characterize by inspection, especially for legacy systems. Finally, the intended discrete logical behavior depends crucially on the physical status: a light may be on or off depending on the current in a lamp resistor; an engine may be powered or not depending on whether the magnetic field induced by a coil is sufficient to close a switch.

In the context of analyzing legacy systems, one is interested in characterizing the specification of a set of controlling actions in terms of their effects on the system state. These inputs may trigger a sequence of complex internal changes, both *discrete* (like relay interactions) and *continuous* (like the charging process

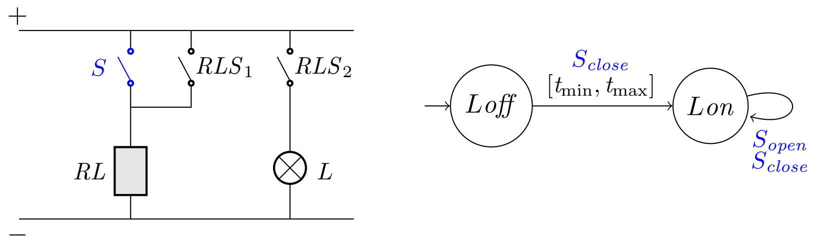

Fig. 1. A circuit regulating a lamp L, based on a switch S and a relay RL, and an automaton that describes the effects of the external actions on the lamp state.

of a capacitor) before reaching the next stable state. The duration of these evolutions is also important: after an action, it may be necessary to wait some time before evaluating the state of the system or before accepting a new input.

As an example, consider the simple circuit represented on the left-hand side of Figure 1. It receives external inputs corresponding to actions on the switch S: when the delayed relay RL receives enough current (i.e., if the switch stays closed for a sufficient time for its charging process) it closes the corresponding switches RLS_1 and RLS_2. We consider the problem of extracting the relation between some evaluations of interest, e.g., the lamp L being on or off, and the sequence of performed actions, in a formalism that allows formal verification.

Contribution. In this paper we investigate the problem of characterizing the effects of events on a hybrid system by analyzing where the triggered behaviors stabilize. We define the notion of $\mathbb{P}$-*stable abstraction* as the automaton that captures the essence of stabilization following each external input.

The granularity of the abstraction is induced by a given set of relevant predicates $\mathbb{P}$. Intuitively, an (abstract) state is associated to predicate valuations, and identifies the (concrete) states that are stable in the corresponding region. The transitions between abstract states describe the stabilization process of the concrete system when responding to an external stimulus. The abstraction is made accurate by requiring the stability regions to be *minimal*: the stability of a trajectory is defined in terms of the smallest $\mathbb{P}$-representable region in which the trajectory eventually converges. The synthesis of $\mathbb{P}$-stable abstractions directly results in a set of temporal properties that are satisfied by the concrete system, and can therefore be used in reverse-engineering and migrating to new technology. In order to capture the duration of stabilizations, a $\mathbb{P}$-stable abstraction is enriched with timing information characterizing the time spent in unstable states. This information can be used to synthesize the correct value to impose a slow-switching hypothesis on the external environment of the system [17].

Second, we prove that the problem can be recast in the framework of Abstract Interpretation (AI) [14] and propose a synthesis algorithm based on the exploration of the abstract state space. At the core of the algorithm is the computation of sufficient conditions for stability. The AI framework is fundamental to seamlessly approximate $\mathbb{P}$-stable abstractions by reducing the precision of the abstract domain. A compositional approach is possible where the analysis algorithm is applied to the individual components of a complex network, later combining their $\mathbb{P}$-stable abstractions. This approach is sound and, in the practically relevant case of discrete interaction of components, it may obtain the same result as obtained by the $\mathbb{P}$-stable abstraction to the complex network.

The proposed algorithm has been implemented in a symbolic analysis tool leveraging the PPLite library [5] for convex polyhedra. An experimental evaluation, focusing on a set of parametric benchmarks representing circuits with run-to-completion behaviors, shows that the proposed techniques can obtain abstractions of rather complex hybrid systems, also providing evidence for the need of adopting the compositional analysis approach in order to improve on scalability.

Structure of the paper. In Section 2 we discuss related work. In Section 3 we introduce some background. In Section 4 we introduce $\mathbb{P}$-stable abstraction. In Section 5 we cast the problem in the AI framework. In Section 6 we discuss the implementation and we experimentally evaluate the approach. In Section 7 we draw some conclusions and outline directions for future work.

2 Related works

State of the art. Stability is an important property of dynamical and hybrid systems which has been widely studied from different perspectives. Classic stability is defined by requiring that all the trajectories are asymptotically attracted by an equilibrium point x_e [9,17]. Since classical asymptotic stability excludes oscillating behaviors, *region stability* [21,22] requires that the trajectories eventually remain inside a given invariant region, intuitively corresponding to the temporal property $\mathrm{AFAG}R$, for a given region R, even if no single equilibrium point exists. The alternative notion of *strong attractor* also requires that all the trajectories of the system never leave the region once entered, i.e $\mathrm{A}\neg R\mathrm{UAG}R$.

The problem is typically to verify the global stability of a given system, i.e. proving that every trajectory satisfies the required stability criterion (be it asymptotic or region stability). When global stability does not hold, an additional task is to compute the region of attraction, i.e. the set of states whose outgoing trajectories are stable.

Asymptotic stability can be proven by providing a Lyapunov function as a certificate that the energy of the system is decreasing (in its domain) until the equilibrium point is reached. Several methods have been proposed to this aim, with different levels of automation, soundness and scalability [10,16,20,24]. Region stability verification cannot be directly tackled as a reachability problem. It is proved by reduction to liveness checking with combinations of reachability and SMT solving, or based on the use of (cartesian) predicate abstraction [21,22].

Interestingly, in the case of switching systems, stability of the whole system is not implied by the stability of each modality. Some works (e.g. [6,23]) aim at finding conditions on switching sequences in order to ensure the stability of the composed system. Another approach to achieve global stability is to impose a *slow-switching* condition, i.e. there must be a sufficiently high time interval between subsequent inputs. [8,19] prove the adequacy of such a time interval by analyzing the *average dwell time* of the system.

Novelty. This work differs from the works mentioned above in several ways. First, in contrast to verifying stability with respect to a given region, we *synthesize* a $\mathbb{P}$-stable abstraction that characterizes all the system behaviors with respect to stabilization. Notice that we do not rely on a single convergence region being given. We explore the space of possible convergence regions induced by the set $\mathbb{P}$ of predicates, and find the tightest representations. Second, the synthesized region is not a simple invariant of the system: rather, it is *possibly, eventually* invariant only for the trajectories triggered by the event under consideration. Hence we simulate hybrid evolutions with a relation of possible attraction between two stable conditions: we want to express the existence of an eventually convergent trajectory (intuitively corresponding to an EFAGR property, for a given region R), rather than requiring stabilization for all paths (as in AFAGR). Another key difference is that the aforementioned approaches are mainly related to purely dynamical or closed hybrid systems. We adopt a more expressive framework, considering switched systems, open to autonomous events. Specifically, our aim is to analyze the stabilization effects for external inputs, by considering the "closed" dynamic of the system. Finally, we take into account timing information.

This work is also quite distinct from predicate abstraction for hybrid systems [1,2]: the main difference is that predicates are not evaluated in transient states, i.e., "abstract" transitions will connect predicates evaluated only in stable conditions. Consider, for example, that the length of the traces is not retained.

In terms of techniques, this is the first work exploiting the Abstract Interpretation framework [14,15] in the field of stability analysis. We trade precision for efficiency and propose an approximated analysis that can be implemented using known techniques.

3 Background

We write $\mathbb{R}_{\geq}$ for the set of non-negative reals. Given a sequence σ and an index $i \in \mathbb{N}$, let $\sigma[i]$ denote the i-th element of σ. We adopt a logic notation derived by SMT, using a fragment of first-order logic and the theory of Linear Real Arithmetic (LRA). Given a set of Boolean variables L, let $\Psi(L)$ define the set of boolean combinations over L. Given a set of real-valued variables V, let LPred_V define finite conjunctions of LRA predicates with free variables in V. We write $\Psi(L, V)$ to denote SMT(LRA) formulae obtained by boolean combinations of Boolean variables in L and linear predicates over V.

Finite and timed automata. A *finite state automaton* is a tuple $\langle Q, Q_0, A, R\rangle$ where Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, A is a finite set of labels and $R \subseteq (Q \times A \times Q)$ is the labeled transition relation between states. A *timed automaton* [3] $\langle Q, Q_0, C, A, \text{inv}, R\rangle$ is an automaton equipped with a finite set of clocks C, with state invariants $\text{inv}\colon Q \to \text{LPred}_C$ associating each state $q \in Q$ with its clock constraints $\text{inv}(q)$[3] and with $R \subseteq (Q \times A \times \text{LPred}_C \times \wp(C) \times A)$, where: edge $(q, a, g, r, q') \in R$ represents the transition from state q to q', labeled with a and guarded by clock constraints g; the set $r \subseteq C$ gives the set of clocks to be reset with this transition. We adopt notation $q \xrightarrow{a,g,r} q'$.

In a timed automaton with a single clock variable c, whenever a state q is involved only in the untimed transitions $q_i \xrightarrow{a,c:=0} q \xrightarrow{\varepsilon,c\geq m} q_j$ and $\text{inv}(q) = (c \leq M)$, we omit it from the set of states in the tuple and write $q_i \xrightarrow{a,[m,M]} q_j$, meaning that q_i reaches q_j with a transition labeled with a in a time between m and M. When clear from context we also omit the 'inv' component from the tuple.

Hybrid Systems. Let $\dot{v}$ denote the time derivative dv/dt. A *linear hybrid system* with piecewise affine dynamics is a tuple $\mathcal{H} = \langle \mathit{Loc}, \mathit{Var}, A, \text{inv}, \text{init}, \text{flow}, \text{disc}\rangle$ where [25]: Loc is a finite set of locations; $\mathit{Var} = \{v_1, \ldots, v_n\}$ is a finite set of continuous state variables; A is a finite set of synchronization labels; $\text{init}\colon \mathit{Loc} \to \text{LPred}_{\mathit{Var}}$ defines initial conditions for each location; $\text{inv}\colon \mathit{Loc} \to \text{LPred}_{\mathit{Var}}$ defines invariant conditions for each location; $\text{flow}\colon \mathit{Loc} \to \text{LPred}_{\mathit{Var} \cup \dot{\mathit{Var}}}$ defines the continuous transition relation; $\text{disc} \subseteq (\mathit{Loc} \times A \times \mathit{Loc} \times \text{LPred}_{\mathit{Var} \cup \mathit{Var}'})$ defines the labeled discrete transition relation. A *state* of a hybrid system $\mathcal{H}$ is a tuple $\langle \ell, \boldsymbol{x}\rangle$ where $\ell \in \mathit{Loc}$ and $\boldsymbol{x} \in \mathbb{R}^n$. Let Σ denote the state space of $\mathcal{H}$ and $\text{init}(\mathcal{H})$ denote the set of states $s = \langle \ell, \boldsymbol{x}\rangle$ such that $\boldsymbol{x} \models (\text{inv}(\ell) \wedge \text{init}(\ell))$. A *run* of hybrid system $\mathcal{H}$ is a path $\rho = (s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{\delta_3} s_3 \xrightarrow{a_4} \ldots)$ where $\delta_i \in \mathbb{R}_{\geq}$, $a_i \in A$, $s_i = \langle \ell_i, \boldsymbol{x}_i\rangle \in \Sigma$, $s_0 \in \text{init}(\mathcal{H})$ and each step corresponds to either a continuous transition

$$\frac{\begin{array}{c}\delta \in \mathbb{R}_{\geq} \qquad f\colon [0,\delta] \to \mathbb{R}^n \qquad \dot{f}\colon (0,\delta) \to \mathbb{R}^n \qquad f(0) = \boldsymbol{x} \qquad f(\delta) = \boldsymbol{x'} \\ \forall \epsilon \in [0,\delta] : f(\epsilon) \models \text{inv}(\ell) \qquad \forall \epsilon \in (0,\delta) : (f(\epsilon), \dot{f}(\epsilon)) \models \text{flow}(\ell)\end{array}}{\langle \ell, \boldsymbol{x}\rangle \xrightarrow{\delta} \langle \ell, \boldsymbol{x'}\rangle}$$

or a discrete transition

$$\frac{(\ell, a, \ell', \mu) \in \text{disc} \qquad (\boldsymbol{x}, \boldsymbol{x'}) \models \mu \qquad \boldsymbol{x'} \models \text{inv}(\ell')}{\langle \ell, \boldsymbol{x}\rangle \xrightarrow{a} \langle \ell', \boldsymbol{x'}\rangle}.$$

We write $\text{Run}(\mathcal{H})$ for the set of all runs of $\mathcal{H}$. A run ρ diverges if ρ is infinite and the sum $\sum_{i\geq 0} \delta_i$ diverges.[4] We consider systems without Zeno behaviors, i.e., every finite run of $\mathcal{H}$ is a prefix of some divergent run of $\mathcal{H}$. If $\text{Run}(\mathcal{H}) \subseteq \text{Run}(\mathcal{H}')$, then $\mathcal{H}'$ is a *relaxation* of $\mathcal{H}$ and $\mathcal{H}$ is a *refinement* of $\mathcal{H}'$ [8].

[3] A clock predicate is a linear predicate over a clock variable c of the form $c \bowtie k$, where k is constant. A clock constraint is a finite conjunction of clock predicates.

[4] We let $\delta_i = 0$ whenever the i-th transition is a discrete one.

In hybrid automata, as well as in a switched systems [17], discrete behaviors can be distinguished between controlled (internal) events, for logic-based mechanism, and autonomous (external) events, modeling unpredictable environmental influences. The set of labels for discrete interaction of $\mathcal{H}$ is therefore partitioned in $A = I \cup E$, denoting internal and external events respectively. Given a hybrid automaton $\mathcal{H} = \langle \mathit{Loc}, \mathit{Var}, A, \mathrm{inv}, \mathrm{init}, \mathrm{flow}, \mathrm{disc}\rangle$, we denote with $\mathcal{H}^c$ the corresponding *closed system*, i.e., $\mathcal{H}^c \doteq \langle \mathit{Loc}, \mathit{Var}, I, \mathrm{inv}, \mathrm{init}, \mathrm{flow}, \mathrm{disc}^c\rangle$ with $\mathrm{disc}^c = \{(\ell, i, \ell', \mu) \in \mathrm{disc} \mid i \in I\}$. Let '$\overset{c}{\rightsquigarrow}$' denote the reflexive and transitive closure of the transition relation of $\mathcal{H}^c$. When clear from the context, we also use '$s \overset{c}{\rightsquigarrow} s'$' to denote the corresponding run from s to s' in the closed system.

Time of a run and slow switching. For each run $\rho = (s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{a_2} s_2 \dots)$ and index m, the time spent to complete the prefix of length m of ρ is $\tau_m(\rho) \doteq \sum_{i=1}^{m} \delta_i$. For a finite run of length n, $\tau(\rho)$ is a shortcut for $\tau_n(\rho)$. Let ε be the ordered sequence of indices for the external events of ρ. The sequence of *external switching time points* of ρ is a sequence t of elements in $\mathbb{R}_{\geq}$ such that $t[0] = 0$ and $t[i] = \tau_{\varepsilon[i]}(\rho)$ for all i. For each hybrid automaton $\mathcal{H}$ and time $d \in \mathbb{R}_{\geq}$, $\mathcal{H}_d$ is a refinement of $\mathcal{H}$ such that every run of $\mathcal{H}_d$ is associated with a sequence t of external switching time points satisfying $(t[i+1] - t[i]) > d$ for all i.

Abstract Interpretation. We assume some familiarity with the basic notions of lattice theory [7] and Abstract Interpretation [14,15]. Given a poset $(L, \preceq)$ and a set $S \subseteq L$, its *downward closure* is $\downarrow S \doteq \{x \in L \mid \exists s \in S \,.\, x \preceq s\}$; the set $S \subseteq L$ is downward closed if $S = \downarrow S$. The notation for *upward closure* is similar. Given two posets $(L, \preceq)$ and $(L^\sharp, \preceq^\sharp)$ and two monotonic functions $\alpha \colon L \to L^\sharp$ and $\gamma \colon L^\sharp \to L$, the pair (α, γ) is a *Galois connection* [14], denoted $L \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} L^\sharp$, if for all $x \in L$, $x^\sharp \in L^\sharp$ it holds that $\alpha(x) \preceq^\sharp x^\sharp$ if and only if $x \preceq \gamma(x^\sharp)$. For each concrete function $F \colon L \to L$, its most precise sound abstraction in $L^\sharp$ according to this Galois connection is $F^\sharp \doteq \alpha \circ F \circ \gamma$.

Temporal Logic. In the rest of this paper, we adopt a notation inspired to model checking for specific patterns of computation-tree logic formulae. In particular, $\mathcal{H}, s \models \mathrm{AG}\phi$ means that for all $\rho \in \mathrm{Run}(\mathcal{H})$ outgoing from s (i.e. such that $\rho[0] = s$), for all $i \in \mathbb{N}$, $\rho[i] \models \phi$. Similarly $\mathcal{H}, s \models \mathrm{EFAG}\phi$ means that there exists a run $\rho \in \mathrm{Run}(\mathcal{H})$ outgoing from s, and a $j \in \mathbb{N}$ such that $\rho[j] \models \mathrm{AG}\phi$.

4 $\mathbb{P}$-stable Abstraction

In this section we characterize the stabilizing executions of a closed hybrid system with respect to the truth values of a given set of predicates. Namely, we build an abstract timed automaton whose transitions simulate the stabilizing runs of the closed system following the occurrence of an external event.

4.1 Region stability

Given a hybrid system $\mathcal{H} = \langle Loc, Var, A, \mathrm{inv}, \mathrm{init}, \mathrm{flow}, \mathrm{disc}\rangle$, where the discrete interactions A are partitioned into the sets of the internal and external events, we call *closed evolution* (for short evolution) of a state $s \in \Sigma$ a hybrid run of the closed system $\mathcal{H}^c$ starting from s. Given a region $R \subseteq \Sigma$ of the state space, we say that a state $s \in \Sigma$ is *internally stable in* R (for short, stable in R) if and only if R is invariant for all closed evolutions of s; state s is said to be *possibly attracted by* R (for short, attracted by R) if there exists a closed evolution of s reaching a state internally stable in R. More formally:

Definition 1 (Stability and attraction). *For each $s \in \Sigma$ and $R \subseteq \Sigma$,*

$$\begin{aligned} \textsc{stable}(s, R) &\iff (s \models^c \mathrm{AG}R); \\ \textsc{attr}(s, R) &\implies (s \models^c \mathrm{EFAG}R). \end{aligned}$$

Different definitions can be provided for relation '$\textsc{attr}$': for instance, one could impose the stronger constraint that the trajectory does not oscillate inside and outside R before stabilizing, hence $\textsc{attr}(s, R) \doteq (s \models^c \mathrm{E}(\neg R \mathrm{U} \mathrm{AG}R))$.[5] Nonetheless, the results stated in the following will also hold for the weaker definition $\textsc{attr}(s, R) \doteq (s \models^c \mathrm{EFAG}R)$.

The "run-to-completion" closed evolutions of a state $s \in \Sigma$ are described by the regions that possibly attract it: due to the non-determinism of the closed system, taken into account by the existential path quantification, this can be a set of different regions, each of them representing a possible future stable condition. The most precise characterization of these behaviors could be given by the set of *minimal* regions for which $\textsc{attr}(s, R)$ holds. The synthesis of such a set presents us with the problem of minimality. In fact, it is common to find trajectories that exhibit asymptotic behaviors to an equilibrium condition: for instance, the discharging process of a capacitor is described by an exponential flow that, ideally, never reaches a null charge, so that a minimal region of convergence does not exists. Nonetheless, under a certain threshold the capacitor can be assumed to be discharged and the following decay of voltage has no impact on its behavior in the circuit. In other words, within a certain stable region of interest, the exact trajectories may not be relevant for the analysis of the system. This suggests to fix a priori a finite set of predicates $\mathbb{P}$ representing the properties we want to observe: hence, the run-to-completion behaviors will be described as the minimal attracting areas chosen from the (finite) set of regions induced by $\mathbb{P}$.

4.2 Untimed $\mathbb{P}$-stable abstraction

Given a finite set of predicates $\mathbb{P} \subseteq \Psi(Loc, Var)$ we denote with $\Phi_{\mathbb{P}}$ the set of their (finite) boolean combinations, omitting the subscript when clear from context. We say that $\mathbb{P}$ induces a grid in the state space, since every formula $\phi \in \Phi$ defines a $\mathbb{P}$-expressible region as the set of its models in Σ. Both relations

[5] This can be seen as the "existential" version of the *strong attractor* definition [21].

'STABLE' and 'ATTR' defined in Section 4.1 can be evaluated on such regions. Note that, given a state s, the set of formulae in Φ such that $\textsc{stable}(s, \phi)$ holds is upward closed (i.e., for all $\phi, \phi' \in \Phi$ such that $\phi \Rightarrow \phi'$, if $\textsc{stable}(s, \phi)$ then $\textsc{stable}(s, \phi')$). Having fixed a finite set of $\mathbb{P}$-representable regions, we can now define the minimal version of relations STABLE and ATTR.

Definition 2 (Minimal $\mathbb{P}$-stability). *For each $s \in \Sigma$ and $\phi \in \Phi$, let*

$$\textsc{no\_stronger\_attr}(s, \phi) \doteq \nexists \phi' \in \Phi \,.\, ((\phi' \Rightarrow \phi) \wedge (\phi' \neq \phi) \wedge \textsc{attr}(s, \phi'));$$
$$\textsc{stable}_{\min}(s, \phi) \doteq \textsc{stable}(s, \phi) \wedge \textsc{no\_stronger\_attr}(s, \phi);$$
$$\textsc{attr}_{\min}(s, \phi) \doteq \textsc{attr}(s, \phi) \wedge \textsc{no\_stronger\_attr}(s, \phi).$$

Namely, relation '$\textsc{stable}_{\min}$' links a state $s \in \Sigma$ with a formula $\phi \in \Phi$ if ϕ is invariant for the evolutions of s in the closed system and if there are no smaller $\mathbb{P}$-representable regions in which s can converge. If such a ϕ exists, it is unique (modulo logical equivalence) and state s is said to be *minimally* $\mathbb{P}$*-stable* (or simply stable). If a state has no formula in Φ such that '$\textsc{stable}_{\min}$' holds, then it is said to be *transient*, since there exists an evolution attracted by a smaller $\mathbb{P}$-region that does not contain it. In the following we use the notation $\textsc{transient}(s)$. Observe that, by definition, the relation '$\textsc{attr}_{\min}$' is a subset of relation 'ATTR' and, like the latter, it is non-deterministic: '$\textsc{attr}_{\min}$' links a state with the minimal regions in which one of its closed evolution stabilizes.

We say that a system $\mathcal{H}$ has a *well-defined run-to-completion semantics* for the set of predicates $\mathbb{P}$, written $\textsc{wd-rtc}(\mathcal{H}, \mathbb{P})$ for short, if a state of $\mathcal{H}$ cannot delay indefinitely its reaching a $\mathbb{P}$-stable condition. In other words, if $\textsc{attr}_{\min}(s, \phi)$, not only there exists a path $s \overset{c}{\rightsquigarrow} s'$ with $\textsc{stable}_{\min}(s', \phi)$, but also, there is no path that satisfies G TRANSIENT. This requirement expresses that the system reaction to an event must be reliable and there must exist a finite time after which the system has completed the process.

As an example, consider an automaton with states $Q = \{q_0, q_1, q_2\}$ and transition relation $R(q_0, q_1)$, $R(q_1, q_1)$, $R(q_1, q_2)$, $R(q_2, q_2)$. The runs starting from q_0 are attracted by predicate $\{q_2\}$, but are allowed to stay in the transient state q_1 for an unbounded number of steps before reaching q_2. Hence, the stabilization process with respect to $\{q_2\}$ is not well-defined. Nevertheless, for the same system WD-RTC holds with predicate $\{q_1 \vee q_2\}$.

Definition 3 ($\mathbb{P}$-stable abstraction). *Let $\mathcal{H} = \langle Loc, Var, A, \mathrm{inv}, \mathrm{init}, \mathrm{flow}, \mathrm{disc}\rangle$ be a hybrid automaton, and $E \subseteq A$ a set of external events. Let $\mathbb{P} \subseteq \Psi(Loc, Var)$ be a set of predicates. The (untimed) $\mathbb{P}$-stable abstraction of $\mathcal{H}$ is the finite state automaton $\mathcal{A} \doteq \langle \Phi, \Phi_0, E, \hookrightarrow\rangle$, where*

$$\Phi_0 = \{\, \phi \in \Phi \mid \exists s_0 \in \mathrm{init}(\mathcal{H}), s \in \Sigma \,.\, s_0 \overset{c}{\rightsquigarrow} s \wedge \textsc{stable}_{\min}(s, \phi) \,\}$$

and $\phi \overset{e}{\hookrightarrow} \phi'$ if and only if there exist $s, s_1, s' \in \Sigma$ such that

$$\textsc{stable}_{\min}(s, \phi), \quad s \xrightarrow{e} s_1 \overset{c}{\rightsquigarrow} s', \quad \textsc{stable}_{\min}(s', \phi').$$

Intuitively, the runs of $\mathcal{A}$ represent the evolution of the truth values of the predicates in $\mathbb{P}$ in response to external events once stabilization is reached, upon "absorption" of the transient states. Notice that initial states Φ_0 are defined following the same intuition, and represent the possible initial $\mathbb{P}$-stable regions, since $\mathcal{H}$ may start in a transient condition.

The definition of $\mathbb{P}$-stable abstraction captures the fact that we are studying the effects of the external inputs when $\mathcal{H}$ is *in a stable condition*: we disregard runs where external inputs are received in transient states. This "stable-switching" paradigm can be intuitively thought of as the qualitative counterpart of the slow-switching hypothesis used as sufficient condition for reasoning about global stability in switched systems [17]. We informally define the restriction of $\mathcal{H}$ under stable switching as the hybrid automaton $\mathcal{H}_{ss}$ obtained by applying the guard $\neg$TRANSIENT to every external transition of $\mathcal{H}$.

The $\mathbb{P}$-stable abstraction of $\mathcal{H}$ is a weak simulation [18] of $\mathcal{H}_{ss}$. The precision depends on the choice of $\mathbb{P}$. In fact, two states that are stable in the same region ϕ are not necessarily connected by a concrete run: when distinct areas of attraction are represented with the same formula, spurious behaviors may be introduced.

We compare the definition of $\mathbb{P}$-stable abstraction with "classic" predicate abstraction [1,2]. In our setting, the concretization of an abstract state ϕ is the set of states in Σ that are stable in ϕ. Similarly, the abstract transitions represent concrete transitions of the form $s \xrightarrow{e} s_1 \overset{c}{\rightsquigarrow} s'$, i.e., a single external event possibly followed by internal transitions. Thus, differently from predicate abstraction, a path in the abstraction may be significantly shorter than the corresponding concrete ones.

For each $\phi \overset{e}{\hookrightarrow} \phi'$ in $\mathcal{A}$, let $\Gamma(\phi \overset{e}{\hookrightarrow} \phi')$ be the set of the simulated hybrid runs:

$$\Gamma(\phi \overset{e}{\hookrightarrow} \phi') \doteq \left\{ s \xrightarrow{e} s_1 \overset{c}{\rightsquigarrow} s' \;\middle|\; \text{STABLE}_{\min}(s,\phi),\ \text{STABLE}_{\min}(s',\phi') \right\}.$$

The Γ operator is naturally extended to runs of $\mathcal{A}$ [6] by concatenation of its applications to single transitions, and to set of runs.

Proposition 1. *If* WD-RTC$(\mathcal{H},\mathbb{P})$ *holds, the stable-switching runs of* $\mathcal{H}$ *are simulated by the runs of* $\mathcal{A}$, *i.e.,* $\mathrm{Run}(\mathcal{H}_{ss}) \subseteq \Gamma(\mathrm{Run}(\mathcal{A}))$.

Example 1. Consider the circuit on the left-hand side of Figure 1, where external events include the opening/closing of switch S and the property of interest is the condition of lamp L: letting i_L (resp., i_{RL}) denote the intensity of the current passing through the lamp (resp., the relay), we choose $\mathbb{P} = \{(-c \leq i_L \leq c)\}$, so that the state space is partitioned into $\mathit{Loff} \doteq (-c \leq i_L \leq c)$ and $\mathit{Lon} \doteq \neg \mathit{Loff}$. Initially all the switches are open and neither the relay coil RL nor the lamp L receives current. Hence, the system is (internally) stable in Loff.

Starting from a condition stable in Loff, if the external event S_{close} is received, i_{RL} increases with a continuous dynamics implementing the delay of its

[6] For each $\phi_0 \in \Phi_0$ operator Γ applies to initial transitions as $\Gamma(\hookrightarrow \phi_0) \doteq \{s \overset{c}{\rightsquigarrow} s' \mid s \in \mathrm{init}(\mathcal{H}), \text{STABLE}_{\min}(s',\phi_0)\}$.

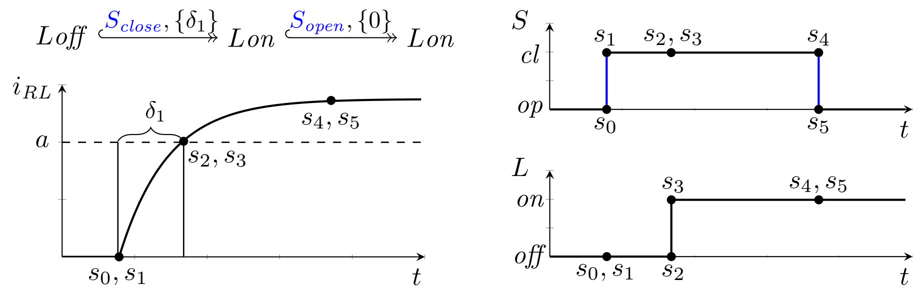

Fig. 2. States of trace $s_0 \xrightarrow{S_{close}} s_1 \xrightarrow{\delta_1} s_2 \xrightarrow{i} s_3 \xrightarrow{\delta_3} s_4 \xrightarrow{S_{open}} s_5$ projected on i_{RL}, the state of the switch S and of lamp L with respect to time t.

activation. After δ_1 time, i_{RL} reaches the activation threshold a (state s_2) and it closes the corresponding switches with an *internal* discrete transition (labeled i). The closing of RLS_2 turns on the lamp L. The system is now stable in Lon. The hybrid system run $s_0 \xrightarrow{S_{close}} s_1 \xrightarrow{\delta_1} s_2 \xrightarrow{i} s_3$ (with $\textsc{stable}_{\min}(s_0, Loff)$, $\textsc{attr}_{\min}(s_1, Lon)$, $\textsc{attr}_{\min}(s_2, Lon)$ and $\textsc{stable}_{\min}(s_3, Lon)$) corresponds to a single transition $Loff \xhookrightarrow{S_{close}} Lon$ in the $\mathbb{P}$-stable abstraction.

Since RLS_1 has been closed, RL receives current even if the switch S gets opened. It follows that when later receiving the external event S_{open} in state s_4 (with $\textsc{stable}_{\min}(s_4, Lon)$), the system switches to state s_5 and the lamp stays on (namely, $\textsc{stable}_{\min}(s_5, Lon)$). In the $\mathbb{P}$-stable abstraction we will have the transition $Lon \xhookrightarrow{S_{open}} Lon$.

4.3 Timed $\mathbb{P}$-stable abstraction

We now characterize the time needed to reach a stable condition after receiving an external input.

Definition 4 (Convergence time of $\phi \overset{e}{\hookrightarrow} \phi'$). *For each abstract transition $\phi \overset{e}{\hookrightarrow} \phi'$ its* convergence time *is the interval in $\mathbb{R}_{\geq}$ $ct(\phi \overset{e}{\hookrightarrow} \phi') \doteq [lb, ub]$, where*

$$lb = \inf\left\{ \tau(\rho) \mid \rho \in \Gamma(\phi \overset{e}{\hookrightarrow} \phi') \right\},$$
$$ub = \sup\left\{ \tau_m(\rho) \mid \rho \in \Gamma(\phi \overset{e}{\hookrightarrow} \phi'), \neg\textsc{stable}_{\min}(\rho[m], \phi') \right\}.$$

The convergence time represents the time spent in the transient states. If the system is stable in ϕ and an external event e is received, after $\max ct(\phi \overset{e}{\hookrightarrow} \phi')$ time the system will certainly be stable in ϕ'; before $\min ct(\phi \overset{e}{\hookrightarrow} \phi')$, the system is certainly still in a transient state so that it is not safe to evaluate the truth value of the predicates. Similar considerations apply to initial conditions: each $\phi \in \Phi_0$ is associated with a convergence time $ct(\hookrightarrow \phi)$, that represents the time needed to stabilize at start up.

Note that, since we are assuming $\textsc{wd-rtc}(\mathcal{H}, \mathbb{P})$, the convergence time of Definition 4 is always bounded from above (i.e., $ub < +\infty$). An unbounded convergence time would imply the existence of a path starting from ϕ that can indefinitely postpone its stabilization in ϕ', which is probably an undesirable system behavior. Hence, the computation of convergence time is a practical way to detect the violation of the $\textsc{wd-rtc}(\mathcal{H}, \mathbb{P})$ hypothesis, and obtain diagnostic information to either debug the model or change the set of predicates $\mathbb{P}$.

On the basis of these considerations, we can define a timed automaton that simulates the $\mathbb{P}$-stable runs of the hybrid system $\mathcal{H}$.

Definition 5 (Timed $\mathbb{P}$-stable abstraction). *Given a $\mathbb{P}$-stable abstraction $\mathcal{A} = \langle \Phi, \Phi_0, E, \hookrightarrow \rangle$ the corresponding* timed *abstraction is a timed automaton having as initial state an additional state $\star$ and having*

- *transition $\star \xhookrightarrow{\varepsilon,[m,M]} \phi$, for each $\phi \in \Phi_0$, with $[m, M] = ct(\hookrightarrow \phi)$;*
- *transition $\phi \xhookrightarrow{e,[m,M]} \phi'$, for each $\phi \overset{e}{\hookrightarrow} \phi'$, with $[m, M] = ct(\phi \overset{e}{\hookrightarrow} \phi')$.*

Starting from the additional state $\star$, each path reaches the first stable condition ϕ in the corresponding initialization time $ct(\hookrightarrow \phi)$. Then, after an external event e, it non-deterministically jumps to the next stable condition within the interval imposed by the associated convergence time.

The convergence time information can be used to define the runs we are abstracting with a *slow*-switching characterization, rather than a (possibly uncomputable) stable-switching one. Let $ct = \max\{\, M \mid \phi \xhookrightarrow{e,[m,M]} \phi' \text{ in } \mathcal{A} \,\}$; then the refinement $\mathcal{H}_{ct}$ (see Section 3), allowing external inputs with a delay of (at least) ct, ensures that the system has always sufficient time to reach stability. It follows that $\mathrm{Run}(\mathcal{H}_{ct}) \subseteq \mathrm{Run}(\mathcal{H}_{ss})$, hence, $\mathcal{H}_{ct}$ defines a concrete semantics compliant with $\mathcal{A}$.

Example 2. Reconsider the circuit of Figure 1, whose $\mathbb{P}$-stable abstraction has been analyzed in Example 1. We can compute the timing information of the stabilization processes, obtaining $ct(\mathit{Loff} \xhookrightarrow{S_{close}} \mathit{Lon}) = [\delta_1, \delta_1]$; the other abstract transitions are instantaneous (i.e., their convergence time is 0, as they have no transient states). By waiting δ_1 after each external event, the system $\mathcal{H}_{\delta_1}$ follows the behaviors described by the $\mathbb{P}$-stable abstraction: namely, we know how long switch S must stay closed in order to turn on (and keep on) the lamp.

5 $\mathbb{P}$-stable Abstraction via Abstract Interpretation

The simulation presented in Section 4 may not be computable when dealing with complex hybrid systems. In this section we formulate the same concepts in an Abstract Interpretation framework: this provides a formal setting to search for a balance between precision and efficiency.

In order to abstract the semantics of $\mathcal{H}_{ss}$ (i.e, the stable switching semantics of $\mathcal{H}$), we consider as concrete domain the powerset of hybrid states $(\wp(\Sigma), \subseteq)$ and as concrete function the post-image operator of an external event subject to a stability constraint: $\mathrm{post}_{ss}(S, e) \doteq \{s' \in \Sigma \mid \exists s \in S \,.\, \neg\,\textsc{transient}(s) \wedge s \xrightarrow{e} s'\}$.

$\mathbb{P}$-stable abstraction as Galois connection. The simulation of Definition 3 can be seen as the abstraction of function post_{ss}. Consider functions (α_1, γ_1), with $\alpha_1(S) \doteq \{\phi \in \Phi \mid \exists s \in S \,.\, \textsc{attr}_{\min}(s, \phi)\}$ for each $S \subseteq \Sigma$, and $\gamma_1(F) \doteq \{s \in \Sigma \mid \forall \phi \in \Phi : \textsc{attr}_{\min}(s, \phi) \implies \phi \in F\}$ for each $F \subseteq \Phi$: this couple forms the Galois connection $(\wp(\Sigma), \subseteq) \overset{\gamma_1}{\underset{\alpha_1}{\leftrightarrows}} (\wp(\Phi), \subseteq)$. The use of the powerset of formulae Φ as abstract domain allows for representing a disjunction of regions and describe precisely the non deterministic behavior of the concrete function. The following proposition states that this abstraction defines exactly the same relation introduced in Definition 3.

Proposition 2. *Let $\mathcal{A}$ be the $\mathbb{P}$-stable abstraction of $\mathcal{H}$. For $\phi, \phi' \in \Phi$ and $e \in E$, $\phi \overset{e}{\hookrightarrow} \phi'$ in $\mathcal{A}$ if and only if $\phi' \in \alpha_1(\text{post}_{ss}(\gamma_1(\{\phi\}), e))$; also, $\Phi_0 = \alpha_1(\text{init}(\mathcal{H}))$.*

Approximating the $\mathbb{P}$-stable abstraction. A first simplification step can be to overapproximate disjunctive stable regions with their join: this lets us obtain a deterministic abstract system based on a conservative abstraction of all the possible behaviors of the concrete system. To this aim, we can consider Φ instead of its powerset. In addition, we can further approximate this domain using its cartesian relaxation.

We denote with $(\mathbb{K}, \sqsubseteq)$ the cartesian abstraction [4] of $(\Phi, \Rightarrow)$. This can be formally defined considering the lattice of knowledge values for each predicate $P_i \in \mathbb{P}$, namely $\mathcal{P}_i = (\{\bot_i, p_i, \overline{p}_i, \top_i\}, \sqsubseteq_i)$ with $\bot_i \sqsubset_i p_i \sqsubset_i \top_i$ and $\bot_i \sqsubset_i \overline{p}_i \sqsubset_i \top_i$; then, $(\mathbb{K}, \sqsubseteq) = \left(\otimes_{P_i \in \mathbb{P}} \mathcal{P}_i\right)$, where '$\otimes$' denotes the smash product operator. Hence, every $k \in \mathbb{K}$, with $k \neq \bot$, is a vector $(k_1, \ldots, k_p)$, with $k_i \in \{p_i, \overline{p}_i, \top_i\}$. Every $k \in \mathbb{K}$ represents a formula in Φ: while $\bot$ is 'false', $(k_1, \ldots, k_p)$ is $(\bigwedge_{k_i = p_i} P_i \wedge \bigwedge_{k_i = \overline{p}_i} \neg P_i)$. We will use $k \in \mathbb{K}$ meaning the corresponding formula in Φ. Note that $(\mathbb{K}, \sqsubseteq)$ is a meet sublattice of $(\Phi, \Rightarrow)$: joins are not preserved since the cartesian abstraction cannot express precisely disjunctions between predicates.

We build an abstraction as composition of Galois connections:

$$(\wp(\Sigma), \subseteq) \overset{\gamma_1}{\underset{\alpha_1}{\leftrightarrows}} (\wp(\Phi), \subseteq) \overset{\gamma_2}{\underset{\alpha_2}{\leftrightarrows}} (\Phi, \Rightarrow) \overset{\gamma_3}{\underset{\alpha_3}{\leftrightarrows}} (\mathbb{K}, \sqsubseteq),$$

where α_2 is the join operator on Φ, i.e., $\alpha_2(F) \doteq \vee(F)$, γ_2 is the downward closure, i.e., $\gamma_2(\phi) \doteq \downarrow\{\phi\}$, and (α_3, γ_3) is the cartesian abstraction, i.e., $\alpha_3(\phi) \doteq \sqcap\{k \in \mathbb{K} \mid \phi \Rightarrow k\}$ and $\gamma_3(k) = k$. Let $\alpha \doteq \alpha_3 \circ \alpha_2 \circ \alpha_1$, and $\gamma \doteq \gamma_1 \circ \gamma_2 \circ \gamma_3$.

Definition 6 (Approximated $\mathbb{P}$-stable abstraction). *The approximated $\mathbb{P}$-stable abstraction of hybrid system $\mathcal{H}$ with external events E is the finite state automaton $\mathcal{A}^\sharp \doteq \langle \mathbb{K}, \{k_0\}, E, \hookrightarrow \rangle$, with initial state $k_0 \doteq \alpha(\text{init}(\mathcal{H}))$ and $k \overset{e}{\hookrightarrow} k'$ if and only if $k' = \alpha(\text{post}_{ss}(\gamma(k), e))$.*

In other words, in the *approximated* $\mathbb{P}$-stable abstraction we have transition $k \overset{e}{\hookrightarrow} k'$ if *every* trajectory starting from a state that is stable in k with the external event e will eventually have k' as invariant, provided that no other external events are accepted meanwhile.

By definition, $\gamma \circ \alpha$ is an approximation of $\gamma_1 \circ \alpha_1$: since all the concrete functions we are dealing with are monotone, $\mathcal{A}^\sharp$ of Definition 6 is a sound overapproximation of the $\mathbb{P}$-stable abstraction $\mathcal{A}$ of Definition 3. Spurious behaviors can be introduced mainly because non deterministic trajectories are conservatively abstracted with a single transition. Function 'α' loses the ability to distinguish between states that are stable in a region with the ones that are stable in a greater one, therefore losing part of the minimality of the stable predicates. As an example, given two formulae $\phi_1, \phi_2 \in \Phi$, for Definition 3 the set of states that are considered stable in $\phi_1 \vee \phi_2$, are disjoint from the states that are stable in ϕ_1 or in ϕ_2, because minimality is required. Instead, using 'α_2', we consider stable in $\phi_1 \vee \phi_2$ also all the states that are stable in smaller regions, therefore recovering the monotonicity of the concretization function.

Finally, the use of the cartesian structure $\mathbb{K}$ overapproximates disjunctions.

Extending the same reasoning done in Section 4.2, for each $k \overset{e}{\hookrightarrow} k'$ in $\mathcal{A}^\sharp$, let $\mathit{\Gamma}^\sharp(k \overset{e}{\hookrightarrow} k')$ be the set of runs abstracted by it:

$$\mathit{\Gamma}^\sharp(k \overset{e}{\hookrightarrow} k') \doteq \left\{ s \xrightarrow{e} s_1 \overset{c}{\rightsquigarrow} s' \;\middle|\; \textsc{stable}(s, k),\ \textsc{stable}(s', k') \right\}.$$

Since the stabilization criterion is more slack, $\mathit{\Gamma}(\mathrm{Run}(A)) \subseteq \mathit{\Gamma}^\sharp(\mathrm{Run}(\mathcal{A}^\sharp))$. The computation of convergence time information can be extended as well: the time needed by the approximated system to stabilize after an external input will be lower than the one computed for $\mathcal{A}$. Letting $ct^\sharp = \max\{M \mid k \xhookrightarrow{e,[m,M]} k' \text{ in } \mathcal{A}^\sharp\}$, we have that $\mathcal{H}_{ct^\sharp}$ is a *relaxation* of $\mathcal{H}_{ct}$ and $\mathrm{Run}(\mathcal{H}_{ct^\sharp}) \subseteq \mathit{\Gamma}^\sharp(\mathrm{Run}(\mathcal{A}^\sharp))$. Namely, $\mathcal{H}_{ct^\sharp}$ defines a new concrete semantics that is compliant with $\mathcal{A}^\sharp$. Moreover, for each $t \geq ct^\sharp$, we know that the abstraction soundly analyzes the evolution of predicates along the runs of $\mathcal{H}_t$.

6 Implementation and Experimental Evaluation

A possible approach for the computation of the approximation $\mathcal{A}^\sharp$ of the abstract system $\mathcal{A}$ is outlined in Pseudocode 1. Here, a reachability driven fixpoint computation incrementally adds ($\mathbb{P}$-stable) states and transitions to $\mathcal{A}^\sharp$; to this end, each abstract state $k \in \mathbb{K}$ being processed is paired with a corresponding reached set of states S, such that $\textsc{stable}(s, k)$ holds for every $s \in S$.

The procedure abstracts the initial state and then enters the main loop. Function '$\mathrm{post}_\mathcal{H}$' computes the image of an external discrete transition. The key processing step is the computation of a conservative evolution of a source set S in the closed system, which is performed within $\textsc{internal\_evolve}_\mathcal{H}(S, \mathbb{P})$. Here, we exploit the cartesian approximation to build the vector k' with a linear number of calls to a *region-stability check*: for each $P_i \in \mathbb{P}$ we test whether the evolution of S is eventually invariant in it or in its negation. The stabilization check is based on the search of abstract lasso-shaped traces, disregarding divergent variables like the global clock. Finally, the addition of abstract locations may call a widening operator, possibly incurring further overapproximations but ensuring the convergence of the procedure.

Pseudocode 1 Build the $\mathbb{P}$-stable abstraction of $\mathcal{H}$.

```
function BUILD_ABSTRACTION($\mathcal{H}$, $\mathbb{P}$)
    $\langle \mathcal{A}^\sharp, waiting \rangle \leftarrow \langle \emptyset, \emptyset \rangle$;
    $\langle k_0, S_0, ct_0 \rangle \leftarrow$ INTERNAL_EVOLVE$_\mathcal{H}$(init($\mathcal{H}$), $\mathbb{P}$);
    $\langle \mathcal{A}^\sharp, waiting \rangle \leftarrow$ add_init_state($\mathcal{A}^\sharp$, $\langle k_0, S_0 \rangle$, $ct_0$, $waiting$);
    while $waiting \neq \emptyset$ do
        $\langle k, S \rangle \leftarrow$ pop($waiting$);
        for all $e \in E$ such that $e$ enabled in $S$ do
            $S_e \leftarrow$ post$_\mathcal{H}$($S$, $e$);
            $\langle k', S', ct \rangle \leftarrow$ INTERNAL_EVOLVE$_\mathcal{H}$($S_e$, $\mathbb{P}$);
            $\langle \mathcal{A}^\sharp, waiting \rangle \leftarrow$ add_state($\mathcal{A}^\sharp$, $\langle k', S' \rangle$, $waiting$);
            $\langle \mathcal{A}^\sharp, waiting \rangle \leftarrow$ add_trans($\mathcal{A}^\sharp$, $\langle k, e, ct, k' \rangle$, $waiting$);
    return $\mathcal{A}^\sharp$;
```

We implemented the abstraction with a symbolic LRA-BDD based approach, built on the PPLite library for convex polyhedra [5], and evaluated it on a set of benchmarks representing circuits, shown in Table 1. External events close or open switches and trigger a run-to-completion process given by the activation of relays. Delayed elements are modeled either with their pragmatic approximation in timed components or by following the continuous dynamics of the internal process of charge/discharge. The predicates of interest focus on the state of some lamps. The abstraction is able to highlight the stabilization process of the signal in a still situation as well as in an oscillating one, e.g., in the periodic flashing of some lamps. The temporal properties extracted from the resulting abstract automata have been checked with the NUXMV model checker [11,13].

	$\mathcal{H}$				$\mathcal{A}^\sharp$		
test	locs	vars	$\#E$	$\#\mathbb{P}$	locs	trans	time
s3	512	21	2	6	2	4	0.152
s4	4096	28	2	8	2	4	2.708
s5	32768	35	2	10	2	4	67.570
r3	1024	29	6	6	7	42	2.060
s1_1	128	18	2	1	2	4	0.042
s1_2	2048	25	2	1	2	4	0.374
s1_3	24576	32	2	1	2	4	5.444
s1_4	262144	39	2	1	2	4	99.190
s2_1	1024	25	2	3	2	4	0.356
s2_2	16384	32	2	3	2	4	4.983
s2_3	196608	39	2	3	2	4	96.412
s3_1	8192	32	2	5	2	4	4.743
s3_2	131072	39	2	5	2	4	96.528
s4_1	65536	39	2	7	2	4	101.797
r2_1	512	29	4	3	4	16	1.662
r2_2	2048	36	4	3	4	16	24.592
r3_1	4096	40	6	5	7	42	62.653

	$\mathcal{H}$				$\mathcal{A}^\sharp$		
test	locs	vars	$\#E$	$\#\mathbb{P}$	locs	trans	time
s1	8	7	2	2	2	4	0.001
2×s1	64	14	4	4	4	16	0.061
3×s1	512	21	6	6	8	48	1.448
4×s1	4096	28	8	8	16	128	115.886
5×s1	32768	35	10	10	32	320	–
s2	64	14	2	4	2	4	0.013
2×s2	4096	28	4	8	4	16	13.784
3×s2	262144	42	6	12	8	48	–
r2	128	18	4	4	4	16	0.073
2×r2	16384	36	8	8	8	64	–
h2	4	16	2	2	2	4	0.037
2×h2	16	32	4	4	4	16	13.581
3×h2	64	48	6	6	8	48	–
h3	8	24	2	2	2	4	2.582
2×h3	64	48	4	4	4	16	–
of	16	24	4	1	2	6	0.501
2×of	256	48	8	2	4	24	–

Table 1. $\mathbb{P}$-stable abstraction of models with run-to-completion behaviors.

Compositional reasoning. The current implementation is based on the exploration of a single, monolithic automaton, which may result in reduced scalability. When the concrete system is the composition of multiple subsystems having no practical discrete interaction between them (or when the predicates in $\mathbb{P}$ are influenced only by details that are local to the subsystems), then the $\mathbb{P}$-stable abstraction can be approached compositionally. The right-hand side of Table 1 shows an exponential growth in computation time for the analysis of the parallel composition of several independent, identical circuits. In all the tests whose analysis completes without incurring the timeout threshold (150s), the abstract automaton for the composed system is exactly the cartesian product of the abstraction of its components, so it corresponds – without loss of precision – to the composition of the single abstractions. Hence, the analysis of the overall system can be factorized into the analysis of the subsystems, whose computation and composition are much more efficient. Clearly, in more general cases the predicates may involve relational constraints on the variables of different subsystems, or their internal interaction may have impact on their stability. Hence, the composition of the single abstractions is generally expected to be less precise than the abstraction of the network (but still guaranteed to be an overapproximation). When precision is not enough to verify the target property, a refinement step can be applied, based for example on considering larger subsystems, doing some compositions at the concrete level and hence reducing spurious behaviors.

7 Conclusions

In this paper we tackled the problem of synthesizing an abstract representation of the stabilizing behavior of hybrid automata. We defined $\mathbb{P}$-stable abstractions that have two key distinguishing features: first, they provide the most precise account – with respect to the given set of predicates – of the evolution between stable conditions in response to external events; second, they include timing information derived from the duration of the stabilization process, which provides suitable values for slow-switching control. We proved that the problem of synthesizing $\mathbb{P}$-stable abstractions can be cast in the framework of Abstract Interpretation, and presented a general synthesis algorithm which allows approximating $\mathbb{P}$-stable abstractions with precision depending on the abstract domain being adopted. We show that $\mathbb{P}$-stable abstractions are very informative from a representational standpoint. The experimental evaluation demonstrates that substantial performance improvements can be obtained by a compositional approach, leveraging the structure of hybrid automata networks.

In the future, we will investigate the use of symbolic techniques such as SMT to complement Abstract Interpretation and further improve the scalability and the precision of the engine. On the application side, the synthesis of $\mathbb{P}$-stable abstraction is currently being integrated within a industrial tool chain of the Italian Railway Network [12]. Specifically, the aim is to reverse-engineer legacy relay-based railways interlocking systems, using the $\mathbb{P}$-stable abstraction as reference specification for a computed-based equivalent solution.

References

1. Rajeev Alur, Thao Dang, and Franjo Ivancic. Reachability analysis of hybrid systems via predicate abstraction. In Claire Tomlin and Mark R. Greenstreet, editors, *Hybrid Systems: Computation and Control, 5th International Workshop, HSCC 2002, Stanford, CA, USA, March 25-27, 2002, Proceedings*, volume 2289 of *Lecture Notes in Computer Science*, pages 35–48. Springer, 2002.
2. Rajeev Alur, Thao Dang, and Franjo Ivancic. Counter-example guided predicate abstraction of hybrid systems. In Hubert Garavel and John Hatcliff, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 9th International Conference, TACAS 2003, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003, Proceedings*, volume 2619 of *Lecture Notes in Computer Science*, pages 208–223. Springer, 2003.
3. Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
4. Thomas Ball, Andreas Podelski, and Sriram K. Rajamani. Boolean and cartesian abstraction for model checking C programs. *Int. J. Softw. Tools Technol. Transf.*, 5(1):49–58, 2003.
5. Anna Becchi and Enea Zaffanella. An efficient abstract domain for not necessarily closed polyhedra. In Andreas Podelski, editor, *Static Analysis - 25th International Symposium, SAS 2018, Freiburg, Germany, August 29-31, 2018, Proceedings*, volume 11002 of *Lecture Notes in Computer Science*, pages 146–165. Springer, 2018.
6. Massimo Benerecetti, Marco Faella, and Stefano Minopoli. Automatic synthesis of switching controllers for linear hybrid systems: Safety control. *Theor. Comput. Sci.*, 493:116–138, 2013.
7. G. Birkhoff. *Lattice Theory*, volume XXV of *Colloquium Publications*. American Mathematical Society, Providence, Rhode Island, USA, third edition, 1967.
8. Sergiy Bogomolov, Corina Mitrohin, and Andreas Podelski. Composing reachability analyses of hybrid systems for safety and stability. In Ahmed Bouajjani and Wei-Ngan Chin, editors, *Automated Technology for Verification and Analysis - 8th International Symposium, ATVA 2010, Singapore, September 21-24, 2010. Proceedings*, volume 6252 of *Lecture Notes in Computer Science*, pages 67–81. Springer, 2010.
9. Michael Branicky. Stability of hybrid systems: State of the art. volume 1, pages 120 – 125 vol.1, 01 1998.
10. Robert Brayton and Chris Tong. Stability of dynamical systems: A constructive approach. *Circuits and Systems, IEEE Transactions on*, CAS-26:224 – 234, 05 1979.
11. Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. The nuXmv symbolic model checker. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 334–342. Springer, 2014.
12. Roberto Cavada, Alessandro Cimatti, Sergio Mover, Mirko Sessa, Giuseppe Cadavero, and Giuseppe Scaglione. Analysis of relay interlocking systems via SMT-based model checking of switched multi-domain Kirchhoff networks. In Nikolaj Bjørner

and Arie Gurfinkel, editors, *2018 Formal Methods in Computer Aided Design, FMCAD 2018, Austin, TX, USA, October 30 - November 2, 2018*, pages 1–9. IEEE, 2018.

13. Alessandro Cimatti, Alberto Griggio, Enrico Magnago, Marco Roveri, and Stefano Tonetta. Extending nuXmv with timed transition systems and timed temporal properties. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*, volume 11561 of *Lecture Notes in Computer Science*, pages 376–386. Springer, 2019.
14. Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Robert M. Graham, Michael A. Harrison, and Ravi Sethi, editors, *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*, pages 238–252. ACM, 1977.
15. Patrick Cousot and Radhia Cousot. Refining model checking by abstract interpretation. *Autom. Softw. Eng.*, 6(1):69–95, 1999.
16. Peter Giesl and Sigurdur F. Hafstein. Computation and verification of Lyapunov functions. *SIAM J. Applied Dynamical Systems*, 14(4):1663–1698, 2015.
17. Daniel Liberzon. *Switching in Systems and Control*. Systems & Control: Foundations & Applications. Birkhäuser, 2003.
18. Robin Milner. *Communication and concurrency*. PHI Series in computer science. Prentice Hall, 1989.
19. Sayan Mitra and Daniel Liberzon. Stability of hybrid automata with average dwell time: An invariant approach. volume 2, pages 1394 – 1399 Vol.2, 01 2005.
20. Antonis Papachristodoulou and Stephen Prajna. On the construction of Lapunov functions using the sum of squares decomposition. volume 3, pages 3482 – 3487 vol.3, 01 2003.
21. Andreas Podelski and Silke Wagner. Model checking of hybrid systems: From reachability towards stability. In João P. Hespanha and Ashish Tiwari, editors, *Hybrid Systems: Computation and Control, 9th International Workshop, HSCC 2006, Santa Barbara, CA, USA, March 29-31, 2006, Proceedings*, volume 3927 of *Lecture Notes in Computer Science*, pages 507–521. Springer, 2006.
22. Andreas Podelski and Silke Wagner. Region stability proofs for hybrid systems. In Jean-François Raskin and P. S. Thiagarajan, editors, *Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, Proceedings*, volume 4763 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2007.
23. Hadi Ravanbakhsh and Sriram Sankaranarayanan. Counter-example guided synthesis of control Lyapunov functions for switched systems. In *54th IEEE Conference on Decision and Control, CDC 2015, Osaka, Japan, December 15-18, 2015*, pages 4232–4239. IEEE, 2015.
24. Sriram Sankaranarayanan, Xin Chen, and Erika Ábrahám. Lyapunov function synthesis using Handelman representations. In Sophie Tarbouriech and Miroslav Krstic, editors, *9th IFAC Symposium on Nonlinear Control Systems, NOLCOS 2013, Toulouse, France, September 4-6, 2013*, pages 576–581. International Federation of Automatic Control, 2013.
25. Stefan Schupp, Erika Ábrahám, Xin Chen, Ibtissem Ben Makhlouf, Goran Frehse, Sriram Sankaranarayanan, and Stefan Kowalewski. Current challenges in the verification of hybrid systems. In Christian Berger and Mohammad Reza Mousavi, editors, *Cyber Physical Systems. Design, Modeling, and Evaluation - 5th International*

Workshop, CyPhy 2015, Amsterdam, The Netherlands, October 8, 2015, Proceedings, volume 9361 of *Lecture Notes in Computer Science*, pages 8–24. Springer, 2015.