Reverse engineering with P-stable Abstractions

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

[1]Fondazione Bruno Kessler, Italy
[2]University of Parma, Italy

Abstract

The analysis of legacy systems such as electro-mechanical circuits requires the extraction of the underlying specification from a high level point of view. We consider $\mathbb{P}$-stable abstraction, an analysis for transition systems that synthesizes the properties of a given set of events in terms of their eventual and stable effects on the system state. The abstract transitions simulate convergent runs of the concrete model and correspond to a set of temporal properties which are satisfied. We show an application of the $\mathbb{P}$-stable abstraction on a relay-based circuit, showing its ability to extract a high level description of the implemented behaviour.

Keywords

Hybrid systems, abstraction, properties extraction

1. Introduction

Context. Hardware and software solutions adopted in safety critical contexts must be verified. Several techniques have been developed leveraging a formal modeling of the behaviours under consideration whose abstraction level must be chosen adequately to describe the properties of interest. We focus on Relay-based Interlocking Systems (RIS), adopted in the railway domain for the control of stations: such systems are built on electro-mechanical components (like power supplies, switches, resistors and relays) which respond to differential equations *and* depend on discrete controlling actions. In [1] the authors propose a methodology to analyze RIS by reduction to a network of hybrid automata: depending on the interest in representing with precision internal inertial electro-mechanical phenomena connecting stationary conditions, these automata can be simplified in systems with piecewise constant dynamics, or even as timed ones. The aforementioned formal representations are amenable for verification with model checkers like: HyCOMP [2], PHAVerLite [3] or nuXmv [4, 5] for timed automata.

Reverse engineering RIS. We consider a complementary analysis to verification, which tackles the *extraction* of the properties that a system satisfies. This research activity is motivated by the need to reverse-engineer RIS arising in the toolchain presented in [6]. Here, the goal is to replace the legacy systems used in the control of stations with a new computer-based logic. One of the difficulties in this task is that it is hard to understand the underlying specifications of RIS, where the controlling logic is left implicit and hidden by non-trivial physical laws, at the

OVERLAY 2021: 3rd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, September 22, 2021, Padova, Italy
abecchi@fbk.eu (A. Becchi); cimatti@fbk.eu (A. Cimatti); enea.zaffanella@unipr.it (E. Zaffanella)



CEUR Workshop Proceedings (CEUR-WS.org)

level of abstraction needed to specify the software control. Our aim is to automatically build a high level description of what are the behaviours exposed by a circuit in response to external events. The specifications extracted from the transition system of the analogue circuit will be cross checked on the new software-based implementation model. Since the two systems we want to compare rely on different sets of variables, we want to synthesize properties defined on the common ones, i.e., the high level observable railway properties, such as the status of a semaphore lamp or a railroad switch. In addition to this "static" abstraction of a state by quantification of the local variables, we consider more sophisticated path-based abstractions. Two implementations may realize a certain behaviour in a different number of internal steps or by passing through different intermediate conditions. For this reason, in order to extract the high level effects of an event, we shall consider its eventual *stable* target.

2. $\mathbb{P}$-stable Abstraction

$\mathbb{P}$-stable abstraction has been firstly introduced in [7] for the characterization of the stabilization behaviour of a hybrid system. This concept is built on two main elements: (a) the set of properties of interest $\mathbb{P}$, defining the granularity of the abstraction; (b) the set of external events E acting on the system, whose specifications we want to extract.

Stability is defined in terms of the set of predicates $\mathbb{P}$. Intuitively, given a $\mathbb{P}$-representable region ϕ, i.e., a region expressible as a boolean combination of predicates in $\mathbb{P}$, a state s is stable in ϕ if it can exit from it only by means of an external event. In other words, if the system is untouched, all the evolutions of s remain in ϕ ($s \models^c \mathrm{AG}\phi$) [1]. When assessing stability, the minimality of the region is also required: namely, ϕ must be the most precise representation available in the grid induced by $\mathbb{P}$ on the state space.

Stability can be broken by an external stimulus $e \in E$ which triggers an evolution that will converge in a possibly different region ϕ' (i.e., $s \xrightarrow{e} s'$ and $s' \models^c \mathrm{EFAG}\phi'$). In the context of hybrid systems, this run-to-completion process can involve both discrete and continuous internal transitions.

The $\mathbb{P}$-*stable abstraction* of a transition system is a finite state automaton whose locations are $\mathbb{P}$-stable states. With respect to the known predicate abstraction [8], here, an abstract transition simulates a convergent *path* in the closed system and the length of the run is not preserved. Moreover, the predicates are not evaluated in transient states, i.e., before stability is obtained.

The $\mathbb{P}$-stable abstraction simulates all the runs of the underlying system, provided that the external stimuli do not occur too frequently: it disregards runs in which an event interrupts a stabilization still in progress. This restriction is justified when considering human controlled devices that have a fast internal response.

By taking into account the duration of this stabilization process, the abstract automaton can be enriched with timing information. The *timed* $\mathbb{P}$-stable abstraction automaton is a timed automaton whose transitions are equipped with an interval which must be inclusive of all possible convergence times. The timing extension of the $\mathbb{P}$-stable abstraction is able to provide the maximum frequency at which the system is allowed to receive inputs, i.e., the value of a *slow-switching constraint* for the external environment that makes the abstraction sound.

[1] '$\models^c$' is a shortcut meaning that the model transition relation is restricted to internal events.

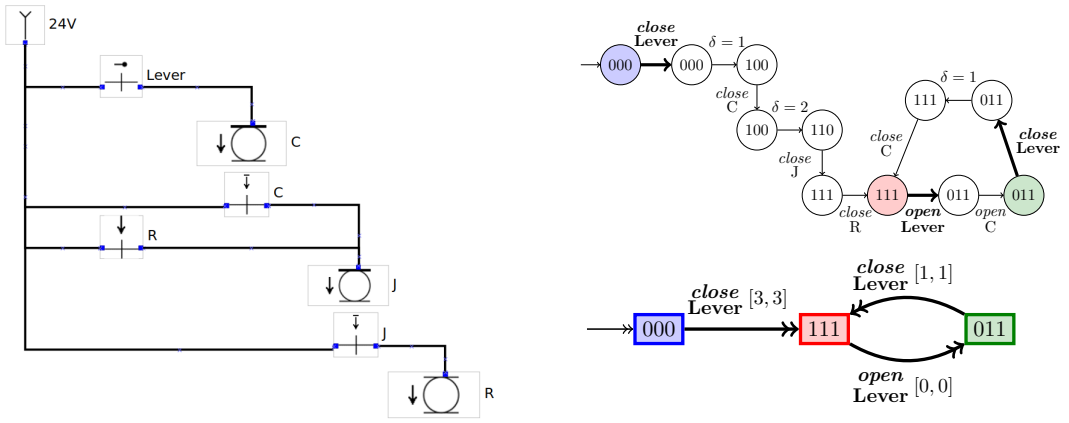

Figure 1: On the left-hand side, a circuit with three relays $\mathrm{C}, \mathrm{J}, \mathrm{R}$, regulated by an external lever Lever; on the right-hand side, a concrete run of the circuit starting from the initial state and its $\mathbb{P}$-stable abstraction.

Computation via abstract interpretation. [7] formulates the problem in the general framework of abstract interpretation [9] which provides a formal setting to trade precision for efficiency. $\mathbb{P}$-stability is defined in terms of a Galois connection (using sets of states as concrete domain and sets of formulae as abstract domain). Sound overapproximations are then obtained by composition with further connections on simpler domains: in particular the abstraction can be made deterministic, by joining the possible next stable formulae, and stable regions can be overapproximated with their cartesian abstraction (therefore losing relational combinations between predicates in $\mathbb{P}$).

The algorithm for the deterministic cartesian abstraction has been implemented with an abstract reachability analysis, manipulating LRA formulae by combining BDDs (with the CUDD library [10]) and convex polyhedra (with the PPLite library [11]).

3. Example

Consider the circuit shown in the left-hand side of Figure 1. The involved components are a voltage generator, three relays and four switches. A relay is an electro-mechanical component able to control a remote switch: when the current that flows in it is above a certain threshold, we say that the relay is activated and its magnetic field can close (or open) a remote switch. A relay may require some time in order to activate after the current starts flowing: we consider relays C and J with an activation delay of 1 and 2 seconds respectively, and a relay R, whose activation is instantaneous. The deactivation process is immediate for all of them. All the switches are controlled by a "master": switches C, J, R are closed if and only if the homonym relay is active; switch Lever, instead, is controlled by an external operator.

The circuit is modeled with a hybrid automaton $\mathcal{H}$ with 77 continuous variables and 18 boolean variables. At the top of the right-hand side of Figure 1 we show a concrete run of $\mathcal{H}$; below it, we show the (deterministic and cartesian approximation of the) corresponding $\mathbb{P}$-stable abstraction obtained with predicates $\mathbb{P} = \{\mathrm{activeC}, \mathrm{activeJ}, \mathrm{activeR}\}$, and events

$E = \{close\mathrm{Lever}, open\mathrm{Lever}\}$. For exposition purposes we label a state with a vector $\mu \in \{0,1\}^3$ whose positions denotes the active status of relays C, J, R respectively.

Initially, all the relays are inactive and all the switches are open: the system is stable in state 000. When Lever is closed, current starts to flow in relay C and after $\delta = 1$ it gets activated. With a discrete transition, it closes the corresponding switch, allowing relay J to start its activation process. After $\delta = 2$, another discrete transition closes switch J and relay R is immediately active. It follows that the closure of Lever triggers a run-to-completion process towards state 111, whose convergence time is 3 seconds. Starting from this state, if Lever opens, then relay C gets deactivated: nonetheless, thanks to the switch R, relay J continues to receive current and the system is stable in state 011. If Lever closes again, the system only requires the activation of relay C to converge again in state 111 after 1 second.

We have synthesized the $\mathbb{P}$-stable abstraction which highlights the stable states, i.e., the ones from which the system can exit only with an external event. Each transition is equipped with a time interval, defining the duration of the convergence process: in this case, it defines that if two subsequent inputs are separated by at least 3 seconds, then the system has always sufficient time to stabilize and the $\mathbb{P}$-stable abstraction is a sound simulation of all its behaviours.

The reverse engineering value of the obtained automaton relies on the abstraction of the transient states, which are proper of this particular implementation of the desired logic. As an example, a state in which R is active but the corresponding switch is still closed is disregarded, since this process is actually instantaneous and introduced only by the modeling choices. As a matter of fact, instantaneous transitions between the activation of a relay and the action on the corresponding switch are fundamental to faithfully model the causality of the events, especially with circular topologies, but have no physical relevance. Moreover, also the states which are reachable only within a process with a non-null convergence time are skipped: consider that "$\mathrm{activeR} \leftrightarrow \mathrm{activeJ}$" is not an invariant of the circuit, but it is respected in the $\mathbb{P}$-stable abstract automaton.

4. Conclusions

We presented $\mathbb{P}$-stable abstractions, an approach proposed in [7] for the extraction of the properties that a hybrid system satisfies. We showed its application on a relay circuit.

In the future we are considering a more general *stability-abstraction* framework, able to provide different levels of expressiveness and precision for reverse engineering purposes.

References

[1] R. Cavada, A. Cimatti, S. Mover, M. Sessa, G. Cadavero, G. Scaglione, Analysis of relay interlocking systems via smt-based model checking of switched multi-domain kirchhoff networks, in: FMCAD, IEEE, 2018, pp. 1–9.

[2] A. Cimatti, S. Mover, S. Tonetta, HyDI: A language for symbolic hybrid systems with discrete interaction, in: EUROMICRO-SEAA, IEEE Computer Society, 2011, pp. 275–278.

[3] A. Becchi, E. Zaffanella, Revisiting polyhedral analysis for hybrid systems, in: SAS, volume 11822 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 183–202.

[4] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, S. Tonetta, The nuXmv symbolic model checker, in: CAV, volume 8559 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 334–342.

[5] A. Cimatti, A. Griggio, E. Magnago, M. Roveri, S. Tonetta, Extending nuXmv with timed transition systems and timed temporal properties, in: CAV (1), volume 11561 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 376–386.

[6] A. Amendola, A. Becchi, R. Cavada, A. Cimatti, A. Griggio, G. Scaglione, A. Susi, A. Tacchella, M. Tessi, A model-based approach to the design, verification and deployment of railway interlocking system, in: ISoLA (3), volume 12478 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 240–254.

[7] A. Becchi, A. Cimatti, E. Zaffanella, Synthesis of P-stable abstractions, in: SEFM, volume 12310 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 214–230.

[8] R. Alur, T. Dang, F. Ivancic, Reachability analysis of hybrid systems via predicate abstraction, in: HSCC, volume 2289 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 35–48.

[9] P. Cousot, R. Cousot, Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: POPL, ACM, 1977, pp. 238–252.

[10] F. Somenzi, Cudd: Cu decision diagram package release, 1998.

[11] A. Becchi, E. Zaffanella, PPLite: Zero-overhead encoding of NNC polyhedra, Inf. Comput. 275 (2020) 104620.