Nonlinear Analysis: Hybrid Systems 25 (2017) 60–78

Contents lists available at ScienceDirect

Nonlinear Analysis: Hybrid Systems

journal homepage: www.elsevier.com/locate/nahs

A Sums-of-Squares extension of policy iterations

Assalé Adjé [a,*,1], Pierre-Loïc Garoche [b], Victor Magron [c]

[a] *Laboratoire de Mathématiques et Physique (LAMPS), Université de Perpignan, Perpignan, France*
[b] *Onera, The French Aerospace Lab, France*
[c] *CNRS Verimag, 2 av de Vignate, 38610 Gières, France*

ARTICLE INFO

Article history:
Received 9 February 2016
Accepted 4 March 2017
Available online 28 March 2017

Keywords:
Piecewise polynomial systems
Policy iterations
Sums-of-Squares
Semi-algebraic invariants
Verification

ABSTRACT

In order to address the imprecision often introduced by widening operators in static analysis, policy iteration based on min-computations amounts to considering the characterization of reachable value set of a program as an iterative computation of policies, starting from a post-fixpoint. Computing each policy and the associated invariant relies on a sequence of numerical optimizations. While the early research efforts relied on linear programming (LP) to address linear properties of linear programs, the current state of the art is still limited to the analysis of linear programs with at most quadratic invariants, relying on semidefinite programming (SDP) solvers to compute policies, and LP solvers to refine invariants.

We propose here to extend the class of programs considered through the use of Sums-of-Squares (SOS) based optimization. Our approach enables the precise analysis of switched systems with polynomial updates and guards. The analysis presented has been implemented in Matlab and applied on existing programs coming from the system control literature, improving both the range of analyzable systems and the precision of previously handled ones.

1. Introduction

A wide set of critical systems software including controller systems, rely on numerical computations. Those systems range from aircraft controllers, car engine control, anti-collision systems for aircraft or UAVs, to nuclear power plant monitors and medical devices such pacemakers or insulin pumps.

In all cases, the software part implements the execution of an endless loop that reads the sensor inputs, updates its internal states and controls actuators. However the analysis of such software is hardly feasible with classical static analysis tools based on linear abstractions. In fact, according to early results in control theory from Lyapunov in the 19th century, a linear system is defined as asymptotically stable iff it satisfies the Lyapunov criterion, i.e. the existence of a quadratic invariant. In this view, it is important to develop new analysis tools able to support quadratic or richer polynomial invariants.

While most controllers are linear, their actual implementation is always more complex. e.g. in order to cope with safety, additional constructs such as antiwindups or saturations are introduced. Another classical approach is the use of linear

* Correspondence to: Laboratoire de Mathématiques et Physique (LAMPS), Université de Perpignan, 52, avenue Paul Alduy, 66000 Perpignan, France.
E-mail addresses: assale.adje@univ-perp.fr (A. Adjé), pierre-loic.garoche@onera.fr (P.-L. Garoche), victor.magron@imag.fr (V. Magron).

1 The work was done when the author was at IRIT, Université Paul Sabatier at Toulouse and was supported by the CIMI (Centre International de Mathématiques et d'Informatique) Excellence program ANR-11-LABX-0040-CIMI within the program ANR-11-IDEX-0002-02 during a postdoctoral fellowship.

http://dx.doi.org/10.1016/j.nahs.2017.03.001
1751-570X/

```
x ∈ X^in;
while (r_1^0(x)⋈0 and ... and r_{n_0}^0(x)⋈0){
  case (r_1^1(x)⋈0 and ... and r_{n_1}^1(x)⋈0): x = T^1(x);
  case ...
  case (r_1^i(x)⋈0 and ... and r_{n_i}^i(x)⋈0): x = T^i(x);
}
```

(a) A one-loop program with a switch-case loop body.

Let $X^0 = \{y \mid r_1^0(y) \bowtie 0 \text{ and } \dots \text{ and } r_{n_0}^0(y) \bowtie 0\}$
and for all cases i, $X^i = \{y \mid r_1^i(y) \bowtie 0 \text{ and } \dots \text{ and } r_{n_i}^i(y) \bowtie 0\}$.
We can define the discrete-time switched system:

$$x_0 \in X^{\text{in}},\ \forall k \in \mathbb{N},\ x_{k+1} = T(x_k),\ \text{where } T(x) = T^i(x) \text{ if } x \in X^i \cap X^0$$

(b) The discrete-time switched system correspondence of the program.

Fig. 1. One-loop programs with switch-case loop body and its representation as a switched system.

parameter varying controllers (LPV) where the gains of the linear controller vary depending on conditions: this becomes piecewise polynomial controllers at the implementation level.

We are interested here in bounding the set of reachable values of such controllers using sound analyses, that is computing a sound over approximation of reachable states. We specifically focus on a class of programs larger than linear systems: constrained piecewise polynomial systems.

A loop is performed while a conjunction of polynomial inequalities[2] is satisfied. Within this loop, different polynomial updates are performed depending on conjunctions of polynomial constraints. This class of programs is represented in Fig. 1(a). The program can be analyzed through its switched system representation (see Fig. 1(b)). In the obtained system, the conditions to switch are only governed by the state variable: at each time k, we consider the dynamics T^i such that $x_k \in X^i$. So, the switch conditions do not depend on the time (we do not consider the time when we reach X^i) and are not defined from random variables.

From the point-of-view of the dynamical systems or control theory, to find an over-approximation of reachable values set of a program of the class described in Fig. 1(a) can be reduced to find a positive invariant of its switched system representation (see Fig. 1(b)).

Moreover, the class of switched systems where the switch conditions only depend on the state variable is classical in control theory and includes the nonlinear systems with dead-zone, saturation, resets or hysteresis.

Related works

Reachability analysis is a long-standing problem in dynamical systems theory, especially when the system dynamics is nonlinear. In the particular case of polynomial systems, this problem has recently attracted several research efforts.

In [1], the authors use a method based on sublevel sets of polynomials to analyze the reachable set of a continuous time polynomial system with initial/general constraints being encoded by semialgebraic sets. Their method relies on a so-called iterative *advection algorithm* based on Sums-of-Squares (SOS) and semidefinite programming (SDP) to compute either inner of outer approximation of the backward reachable set, also named *domain of attraction* (DoA). The stability analysis of continuous-time hybrid systems with SOS certificates was investigated in [2]. Posa et al. [3] have recently applied analogous techniques to perform stability analysis and controller synthesis in the context of robotics. Other studies rely on SOS reinforcement and moment relaxations to obtain hierarchies of approximations converging to sets of interest such as the DoA in continuous-time, either from outside in [4] or from inside in [5]. This approach relies on a linearization of the ordinary differential equation involved in the polynomial system, based on Liouville's Equation satisfied by adequate occupation measures. This framework has been extended to hybrid systems in [6], as well as to synthesis of feedback controllers in [7]. Liouville's Equation can also be used to approximate other sets of interest, such as the maximal controlled invariant for either discrete- or continuous-time systems (see [8]).

In contrast with these prior works, our approach focuses on computing over approximation of the forward reachable set of a discrete-time polynomial system with finitely many guards, thanks to an algorithm relying on SOS and template policy iterations.

Template abstractions were introduced in [9] as a way to define an abstraction based on an a priori known vector of templates, i.e. numerical expressions over the program variables. An abstract element is then defined as a vector of reals defining bounds b_i over the templates p_i: $p_i(x_1, \dots, x_d) \le b_i$.

Initially templates were used in the classical abstract interpretation setting to compute Kleene fixpoints with linear functions p_i. Typically, the values of the bound b_i increase during the fixpoint computation until convergence to a post-fixpoint.

2 $\bowtie$ is either the strict $<$ or the weak ($\le$) comparison operator over reals.