A tool for the analysis of dynamical systems by Quantified Constraint Satisfaction

Benjamin Martin

LIX, École Polytechnique

17 Septembre 2018

Réunion de projet MRIS / DGA

The problem

We consider the following (possibly non-linear, non-polynomial) dynamical system

$$\dot{x} = f(x, u)$$

(or its autonomous variant $\dot{x} = f(x)$) with $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ the input and $x(t) \in \mathbb{R}^n$ the states, for all $t \geqslant 0$.
From these systems we would like to be able to analyze :

- safety or stability properties
- liveness or reachability properties

without relying on simulation.

Computing Lyapunov-like functions, Barrier functions, Isolating blocks, Region of attraction, etc

Quantified Constraint Satisfaction Problem

How to ?

Find functions or (semi-algebraic) sets of states satisfying a some constraints, depending on the system.

Example: barrier functions

Given system $\dot{x} = f(x)$, $\mathcal{X}_0$ (initial states), $\mathcal{X}_u$ (forbidden states), a barrier function is a function h satisfying:

$$
\begin{aligned}
(\forall x \in K)\ (h(x) = 0 &\implies \langle \nabla h(x), f(x) \rangle < 0) && (1)\\
\wedge\ (x \in \mathcal{X}_0 &\implies h(x) \leqslant 0) && (2)\\
\wedge\ (x \in \mathcal{X}_u &\implies h(x) > 0). && (3)
\end{aligned}
$$

If such an h exists, $\mathcal{X}_u$ cannot be reached from $\mathcal{X}_0$.

Quantified Constraint Satisfaction Problem

How to ?

Find functions or (semi-algebraic) sets of states satisfying a some constraints, depending on the system.

Example: barrier functions

Given system $\dot{x} = f(x)$, $\mathcal{X}_0$ (initial states), $\mathcal{X}_u$ (forbidden states), a barrier function is a function h satisfying:

$$
\begin{aligned}
&(\forall x \in K)\ (h(x) \neq 0 && \vee \langle \nabla h(x), f(x) \rangle < 0)) && (1)\\
&\wedge\ (x \notin \mathcal{X}_0 && \vee h(x) \leqslant 0) && (2)\\
&\wedge\ (x \notin \mathcal{X}_u && \vee h(x) > 0). && (3)
\end{aligned}
$$

If such an h exists, $\mathcal{X}_u$ cannot be reached from $\mathcal{X}_0$.

Particular QCSP (other objects like invariants, isolating blocks can be modeled in a similar fashion)

The tool

Many (guaranteed) approaches in the literature:

- when the system is linear (Lyapunov and alike, abstractions, etc)
- when the system is polynomial (algebraic approaches, SOS, DSOS, etc)

The tool

Many (guaranteed) approaches in the literature:

- when the system is linear (Lyapunov and alike, abstractions, etc)
- when the system is polynomial (algebraic approaches, SOS, DSOS, etc)

For non-polynomial systems: different approaches based on interval analysis and constraint programming (e.g. Delanoue *et al* [2015], Djaballah *et al* [2017], Goldsztejn and Chabert [2017], Martin and Mullier [2018])

The tool

Many (guaranteed) approaches in the literature:

- when the system is linear (Lyapunov and alike, abstractions, etc)
- when the system is polynomial (algebraic approaches, SOS, DSOS, etc)

For non-polynomial systems: different approaches based on interval analysis and constraint programming (e.g. Delanoue *et al* [2015], Djaballah *et al* [2017], Goldsztejn and Chabert [2017], Martin and Mullier [2018])

I am currently implementing a tool in Ibex[a] (C++) for modeling and solving general problems of this form.

[a] http://www.ibex-lib.org/

Features

- Exploration algorithms for the functions and sets (via template functions or sets)
- Solving algorithms for the QCSP (Branch & Prune), with possible variants
- a "system" structure for modeling the constraints (CNF-like form), with some sugar

Features

- Exploration algorithms for the functions and sets (via template functions or sets)
- Solving algorithms for the QCSP (Branch & Prune), with possible variants
- a "system" structure for modeling the constraints (CNF-like form), with some sugar

(**no name yet !**)

Some examples

```
Variable x(2,"x");
Variable p(4,"p");

Function sys(x, Return(x[0] + x[1], x[0]*x[1] -0.5 * sqr(x[1])));
Function g0(x, sqr(x[0] + 1.25) + sqr(x[1] - 1.25) - 0.05);
Function gu(x, sqr(x[0] + 2.5) + sqr(x[1] - 0.8) - 0.05);
Function temp(x,p, (p[0]*p[1]*(x[0]+p[2]))/(sqr(x[0] + p[2])+sqr(p[1])) + x[1] + p[3]);

IntervalVector K(2); K[0] = Interval(-1000,0); K[1] = Interval(-1000, 1000);
IntervalVector P(4, Interval(-10,10));

UserTemplateFunction tg(sys, 2, 0, 4, temp, P);
tg.buildTemplate();

CertifierFactory cf(tg);
cf.addBarrier();
cf.addContainedSet(g0);
cf.addOutsideSet(gu);

IntervalVector dom(6); dom.put(0, K); dom.put(2, P);

FeasiblePointSearcher bp(...);
bp.branchprune(dom);
```

Work in progress / Future

WIP:

- search strategies for faster decision on the validity or invalidity of the QCSP (currently some good results for fast invalidation)
- additional algorithm variants for solving other problems or improving the current ones (e.g. integral evaluations for dual barriers)
- heuristic exploration of the set of functions and sets (still templated, but towards more variety of functions/sets)

Next ?:

- coupling algebraic approaches to have good guesses, or quantifier elimination / reformulation of the QCSP
- correlations to SMT