EPiC Series in Computing

Volume 43, 2017, Pages 52–59

ARCH16. 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems

Chains of Integrators as a Benchmark for Scalability of Hybrid Control Synthesis (Benchmark Proposal)

Scott C. Livingston[1] and Vasumathi Raman[2]

[1] slivingston@cds.caltech.edu
[2] vasu@cds.caltech.edu

Abstract

Formal methods refers broadly to techniques for the verification and automatic synthesis of transition systems that satisfy desirable properties exactly or within some statistical tolerance. Though historically developed for concurrent software, recent work has brought these methods to bear on motion planning in robotics. Challenges specific to robotics, such as uncertainty and real-time constraints, have motivated extensions to existing methods and entirely novel treatments. However, compared to other areas within robotics research, demonstrations of formal methods have been surprisingly small-scale. The proposed benchmark seeks to motivate advancement of the state of the art toward practical realization by testing scalability of existing tools, and motivating improvements.

1 Introduction

The presented benchmark is part of the first challenge on formal methods for robotics, which will henceforth be referred to simply as "the Challenge." It is to be hosted at the International Conference on Robotics and Automation (ICRA) in May 2016. There are two founding ambitions of the Challenge: to develop benchmark problems for research in so-called "formal methods for robotics," and to create standard interfaces, formats, etc. for expressing problems and using tools that implement methods described in the research literature. Our effort is analogous to that of SMT-LIB, which is for research in satisfiability modulo theories

The Challenge itself will consist of three problem domains that touch on a diversity of difficulties one would need to address in a practical deployment. These problem domains together involve many fronts of current work in formal methods for robotics. To avoid requiring too general of a solution—and in particular, to improve accessibility for a broad group of potential competitors—entries to the challenge are permitted to select a subset of these domains. More precisely, each team may enter any number of control programs, each of which may be used on any subset of the problem domains.

Here we present details of the first such problem domain, that of synthesis for high-dimensional systems modeled by chains of integrators. We describe the problem domain, as well as currently available preparation materials for potential competitors, such as elementary solutions that establish feasibility of the problems, provide reference implementations for the challenge execution framework, and give teams something on which to build, should they choose to do so.

2 Preliminaries

This section introduces notation used throughout the rest of the paper. Let A be a set. 2^A denotes the set of all subsets of A. The set of real numbers is denoted by $\mathbb{R}$. Let $x = (x_1, \dots, x_n) \in \mathbb{R}^n$. For $p \geq 1$, the p-norm of x is defined to be

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}. \tag{1}$$

The 2-norm is also known as the Euclidean distance. The ∞-norm is defined as

$$\|x\|_\infty = \max_i |x_i|. \tag{2}$$

Linear-time temporal logic (LTL) is an extension of Boolean (or propositional) logic that describes properties of countably infinite sequences of events [1]. The two basic temporal operators are $\mathbf{U}$ (pronounced "until") and $\bigcirc$ (pronounced "next"). The formula $\psi \, \mathbf{U} \, \varphi$ requires that formula ψ holds until a state satisfying φ is reached and such a state must eventually be reached. The operator $\Box$ is also used; informally, $\Box \, \psi$ requires ψ is satisfied by all states reached during an execution. Intuitively the operator $\Diamond$ is dual to $\Box$; informally, $\Diamond \, \psi$ requires that a state satisfying ψ occurs in finite time.

A *convex polytope* is a bounded region defined by the intersection of finitely many halfspaces, each of which is defined by a linear inequality. Formally, a polytope defined by k halfspaces, in a space $\mathcal{X} \subseteq \mathbb{R}^n$ is defined by

$$\{x \in \mathcal{X} \mid Hx \leq K\}$$

for some $H \in \mathbb{R}^{k \times n}$ and $K \in \mathbb{R}^k$. There are many useful computations known for polytopes, many of which are fast, e.g., intersection [2].

The discrete notions of temporal logic are related to the real-valued spaces of continuous dynamical systems as follows. Suppose a state space contains finitely many polytopes, and each polytope is regarded as labeled by a Boolean proposition. Let AP be the set of all such Boolean propositions. A trajectory through the space of polytopes is labeled with a sequence of elements of 2^{AP} corresponding to the sequence of polytopes with which it intersects.

3 Terminology

The *problem domain* described in this paper is referred to as the *scaling chains of integrators* domain. Within this problem domain, a problem *instance* is a particular workspace, arrangement of obstacles, labeling of the workspace, selection of parameters for the robot dynamics, and a task formula. A *controller* is the basic entity provided as a solution to the problem domain, and must be able to solve any instance.

4 Scaling chains of integrators

Of the domains in the Challenge, this problem domain is the simplest in terms of dynamics and specifications, yet the system to be controlled can be scaled easily to arbitrarily many dimensions. While this problem is abstract in the sense that it is not modeling a specific physical system, it is well-motivated because it is one of the normal forms for expressing linear systems. More precisely, for a linear system to be controllable, it is necessary and sufficient condition to express the system as a chain of integrators. Moreover, the double-integrator is just the basic force equation of Newtonian mechanics, up to a scaling factor, and so widely used to model systems for which acceleration is the control input. Informally, this benchmark problem domain concerns controlling acceleration or a higher order derivative of a point-mass in a high-dimensional space so as to visit goal regions and avoid obstacles. The precise description is given below.

4.1 Dynamics and constraints

Let n and m be positive integers. Consider the differential equation

$$D^m q = u, \tag{3}$$

where $q : [0, \infty[\to \mathbb{R}^n$ and D is the differential operator. The input u is bounded as $\|u(t)\|_1 \leq u_{\max}$. The system is called a *chain of integrators* because it can be rewritten as a linear control system

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_{m-1} &= x_m \\ \dot{x}_m &= u \\ y &= x_1 \end{aligned} \tag{4}$$

where for time t, each $x_i(t) \in \mathbb{R}^n$ for $i = 1, \ldots, m$, and $x(t) = (x_1(t), \ldots, x_m(t)) \in \mathbb{R}^{nm}$. The output trajectories of (4) are exactly those of (3), given the same input, and thus we call them equivalent. In (3), control input is applied as the m-th derivative of an n-dimensional variable q, and the intermediate derivatives are made explicit in (4). A notable particular case is $m = 2$, which is called the “double integrator”. If $n \leq 3$ then q may be referred to as the “position”.

To introduce process and sensor noise, consider the 2-dimensional system

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u + \xi \tag{5}$$

$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} x + \eta, \tag{6}$$

where ξ and η are either bounded disturbances (nondeterministic) or stochastic processes. Equivalence with (3) in the case of $n = 1, m = 2$ and $\xi = \eta = 0$ is apparent using $x_1 = q$ and $x_2 = \dot{q}$. For $n > 1$, the matrices in (5) and (6) can be repeated in block diagonal form to yield a new linear time-invariant system of dimension $2n$ and which is again equivalent to (3) for $m = 2$.

4.2 Specifications

Imposing task specifications in LTL makes the problem of control synthesis for systems subject to the dynamics in Section 4.1 a hybrid control problem. Task specifications will require visitation of regions while avoiding obstacles. These can be regarded as a generalized version of classical motion planning. There are two forms of specifications:

$$q(0) = \mathrm{init} \wedge \square\,(q(t) \notin \mathrm{Obs}) \wedge \bigwedge_i \square\lozenge\,\mathrm{goal}_i \tag{7}$$

and

$$q(0) = \mathrm{init} \wedge \square\,(q(t) \notin \mathrm{Obs}) \wedge \bigwedge_i (\mathrm{counter}_i \leq T_i)\,\mathbf{U}\,\mathrm{goal}_i, \tag{8}$$

where $\mathrm{Obs} \subset \mathbb{R}^n$ is the obstacle set, which is represented by a finite union of polytopes. For each i, $\mathrm{counter}_i$ is a discrete clock that enforces the real-time deadline T_i of reaching region goal_i. Goal region goal_i is defined by a convex polytope. The initial position is a single point, $\mathrm{init} \in \mathbb{R}^n$. As part of the specification form (8), a time of initialization and rate of progression of the discrete counters is specified. These are significant because they determine in what order and how quickly the goal regions must be reached.

Many approaches in the literature on formal methods for robotics provide at least fragments of a relevant solution, e.g., by providing methods of discrete abstraction. Others can be directly applied, e.g., Kloetzer and Belta [3], and Wolff et al. [4] present algorithms for linear systems, labeled polytopes, and LTL specifications. The objective of this benchmark is to provide a platform for demonstrating scalable implementations of these existing approaches, as well as to motivate new ones.

4.3 Implementation plan for the challenge

This problem domain will be evaluated entirely within a numerical simulation. As such, competitors will submit controllers for this part of the challenge before the conference, and results will be obtained during live runs at ICRA. After completion of each run in real-time, using software released as part of the problem domain source code, animations will be generated to depict results. In order to facilitate visualization of the results, each specification will consist of a conjunction of specifications, each of which governs no more than three dimensions of the state space. This enables plotting trajectories in these 3D subspaces, and visually verifying satisfaction of the specification. Figure 1 depicts such a visualization. Section 6 describes our implementation in further detail.

The main evaluation metric for this domain will be scalability of solutions to high dimensions and large problem sizes. As such, we are currently investigating feasibility of running the trials for this domain on clusters or cloud computing services.

5 Evaluation

Our goal is to allow maximal flexibility of solution form, and allow entries to be provided as black-box programs with defined interfaces. A static evaluation proving that the solution is correct would violate this ideal by, for example, requiring submissions to be expressed in part using Promela, so that they could be checked it with Spin, or provide a linearized-form closed system in order to generate a certificate via SOS programming.

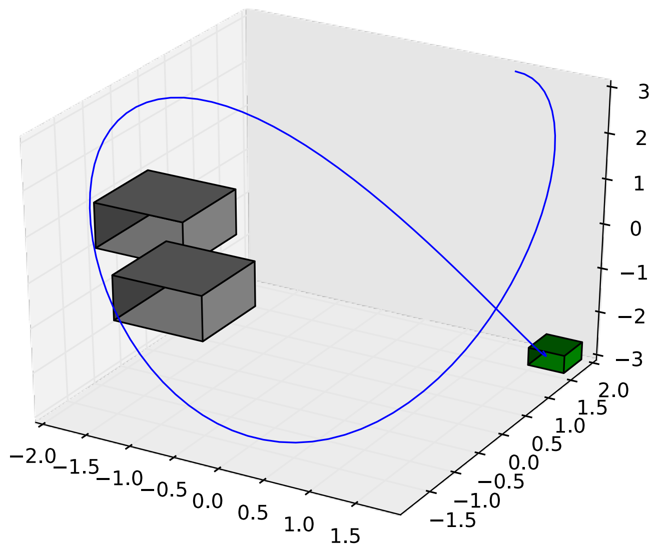

Figure 1: Illustration of an obstacle-avoiding double integrator trajectory in three dimensions

Instead, the evaluation will be *dynamic*, i.e., submissions will be evaluated by running them. It will also be *trial-based*, i.e., each controller will be run for a fixed set of trials that each have the adversary playing a particular strategy. Some care will be made to cover the set of environment strategies.

5.1 Trajectory durations

Since the semantics of LTL is over infinite executions, the duration for which each trial will be run must be specified. The Challenge will feature trials with both pre-specified and unspecified (yet finite) execution durations.

- **unspecified in the problem:** The duration for which each trial will run (in terms of execution time) will be selected randomly at the time of evaluation.
- **specified in the problem instance:** There are two approaches here:
  - A fixed duration for a timeout will be given as part of the instance description. Time discretization of this duration is part of the solution.
  - Termination occurs upon reaching a final state. These specifications are necessarily co-safe.

The chief difficulty in the evaluation is in verifying liveness properties of black-box controllers. To mitigate this problem, we will allow teams the following two options:

- provide trajectories of lasso form, with a pre-specified tolerance for closing the loop of the lasso.
- declare a minimum execution time required to demonstrate correctness.

5.2 Scoring

Entries to the Challenge will be scored on the basis of the cumulative time required to achieve pre-specified milestones once given a specification. For the scaling chains of integrators domain in particular, scoring will be purely on the basis of the time required to synthesize a trajectory that satisfies the specification. Satisfaction will be checked by inspection by plotting the trajectories in each 3D subspace appearing in the subformulas.

6 Implementation

We now describe our implementation for this benchmark against which potential competitors can test their controllers. The source code available includes elementary solutions that establish feasibility of the problem instances, provide reference implementations for the challenge execution framework, and give teams something on which to build, should they choose to do so. All code and documentation is available on the Formal Methods for Robotics Challenge repository, at `https://github.com/fmrchallenge/fmrbenchmark/`.

There are four major kinds of entities in the repository:

- benchmarks;
- analysis tools for reviewing results from using benchmarks;
- examples demonstrating components of benchmarks and solution controllers;
- documentation.

Spanning all four of the above kinds is the supporting infrastructure. This refers to header files, message formats, etc. that may be used by more than one benchmark and that may be of independent interest, besides benchmarking.

6.1 Support for platforms and programming languages

While it may be possible to build the benchmarks and infrastructure on other platforms, most testing currently is done with Ubuntu 14.04 running Linux x86_64 and the following:

- ROS Indigo
- Gazebo as used with ROS
- The benchmarks are primarily implemented in C++ and C. As of version 0.0.0, most of the examples and tools for reviewing results are in C++ and Python.

Additional dependencies required and instructions for building the scaling chains of integrators domain can be found in the User's Guide at `http://docs.fmrchallenge.org/en/v0.0.0/integrator_chains.html`.

6.2 Problem generation

Recall that the basic problem is to find a controller for a given chain of integrators system so that all trajectories repeatedly reach several regions while avoiding others. Let `$FMRBENCHMARK` be the absolute path to a copy of the fmrbenchmark repository on your machine. After building the examples demonstrating the chain of integrators benchmark according to the above instructions, one of the resulting programs is `genproblem`, the source of which is

```
$FMRBENCHMARK/domains/integrator_chains/dynamaestro/examples/standalone/genproblem.cpp.
```

The output is a problem instance in JSON, which can be visualized using the script

```
$FMRBENCHMARK/domains/integrator_chains/analysis/plotp.py.
```

A collection of trials is defined by a configuration file in JSON format. For example:

```
{
  "number_trials": 5,
  "output_dim_bounds": [1, 3],
  "number_integrators_bounds": [1, 3],
  "number_goals_bounds": [1, 10],
  "number_obstacles_bounds": [0, 4],
  "period_bounds": [0.01, 0.05],

  "Y": [10],
  "U": [1],

  "duration_bounds": [30, 60]
}
```

Trials are randomly generated subject to the constraints in the configuration file. In the above example, 5 trials will be generated with between 1 and 3 output dimensions, for a chain of between 1 and 3 integrators, with specifications of between 1 and 10 goals and between 0 and 4 obstacles, with the original system discretized using a period randomly drawn from between 0.01s and 0.05s, the output space bounded above and below by 10m in each dimension and the input space bounded above and below by 1m in every dimension.

6.3 Running controllers

The `lqr.py` example controller is provided to demonstrate how a submitted entry should interact with the benchmark code. To initiate the performance of a collection of trials defined by the configuration file `mc-small-out3-order3.json` in the ROS package `sci_concrete_examples` of example controllers,

- create a catkin workspace as described in the User's Guide
- run

  ```
  $FMRBENCHMARK/domains/integrator_chains/trial-runner.py -l -f mydata.json
            src/sci_concrete_examples/trialconf/mc-small-out3-order3.json
  ```

  This will cause trial data to be saved to the file `mydata.json` in the local directory from where the above command is executed.
- In a separate terminal, run the example controller using:

  ```
  roslaunch sci_concrete_examples lqr.launch
  ```

- You can observe the sequence of states and control inputs using `rostopic echo state` and `rostopic echo input`, respectively. At each time increment, the state labeling is published to the topic `/dynamaestro/loutput` as an array of strings (labels) corresponding to the polytopes containing the output at that time.
- Once all trials have completed, the trial data can be examined using `tdstat.py`. E.g., to get a summary about the data for each trial,

```
$FMRBENCHMARK/domains/integrator_chains/analysis/tdstat.py -s mydata.json
```

To get the labeling of the trajectory for trial 0, modulo repetition,

```
$FMRBENCHMARK/domains/integrator_chains/analysis/tdstat.py -t 0
          --wordmodrep mydata.json
```

References

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking.* MIT Press, 2008.

[2] K. Fukuda. Frequently asked questions in polyhedral computation, August 2004. `http://www.ifor.math.ethz.ch/~fukuda/polyfaq/polyfaq.html`.

[3] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Trans. Automat. Contr.*, 53(1):287–297, 2008.

[4] E. M. Wolff, U. Topcu, and R. M. Murray. Optimization-based trajectory generation with linear temporal logic specifications. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 5319–5325, 2014.