A COMPETITION ON FORMAL METHODS FOR ROBOTICS

VERSION 0.0.4

SCOTT C. LIVINGSTON AND VASUMATHI RAMAN

Abstract. This is the normative reference for the first challenge on formal methods for robotics. 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 robotics challenge seeks to motivate advancement of the state of the art toward practical realization. The challenge is organized into three problem domains: arbitrary dimensional chains of integrators, roads with Dubins cars, and factory cart clearing.

Contents

Date: 2016-05-17.

1. Introduction

We present the first challenge on formal methods for robotics, which will henceforth be referred to simply as "the Challenge." It will be held at the International Conference on Robotics and Automation (ICRA) in May 2016. The purpose of this document is to describe in detail the Challenge, including the problem domains, rules, and scoring procedures. Unless explicitly stated otherwise, the description herein is normative.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

2. Summary

This section is non-normative.

The challenge 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. Results of the challange will be organized by problem domains. Control programs that are used in multiple domains will receive total scores as the sum of scores from each attempted domain. The intent in the scoring structure will be to have excellent performance in a single problem domain regarded as being comparable to good performance in all domains. Teams will be able to trade-off generality with problem-specific tuning.

In the preparation materials to be made available for potential competitors, we will release elementary solutions to complete each domain. These will 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.

3. Preliminaries

This section is non-normative.

This section introduces notation used throughout the paper. It also gives brief introduction to concepts that may not be widely known by potential participants.

Let A be a set. 2^A denotes the set of all subsets of A. The set of nonnegative real numbers is denoted by $\mathbb{R}_{\geq 0}$. Let $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. 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 for countably infinite sequences of events. The two basic temporal operators are $\mathbf{U}$ (pronounced "until") and $\bigcirc$ (pronounced "next"). $\psi \,\mathbf{U}\, \varphi$ requires that the 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 $\diamondsuit$ is dual to $\Box$; informally, $\diamondsuit\, \psi$ 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. There are many useful computations known for polytopes [3].

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. A trajectory through this space is labeled with the sequence of polytopes with which it intersects.

4. Terminology

A *team* is the basic entity that can compete and, depending on performance, receive rankings in the Challenge. A team must have a name by which it will be referred to during the Challenge.

There are three *problem domains*, which are described in Sections 6, 7, and 8 (one per section). Within a 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 problem instance is defined as a particular selection of values consistent with a problem domain. It can include refining details due to particular constraints of the problem variant. E.g., while the problem is defined in terms of dense-time, an instance may have a sampling period, which together with application of inputs using zero-order hold implies a system discretization.

Each problem domain may have one or more *variants*, which concern different manners of implementation or random instance generation. E.g., the traffic of Dubins cars domain (cf. Section 7) has a simulation variant, which relies on Gazebo, and a physical variant, which involves a physical testbed of several Kobuki bases as well as an overhead pose tracking system.

A *controller* is the basic entity that a team submits for the Challenge within a particular problem domain. Each controller is scored and ranked, and the score of a team is simply the maximum score of any controller belonging to that team.

5. Procedure

The scaling chains of integrators (Section 6) domain will proceed entirely in simulation. The other two domains (described in Sections 7 and 8) will include both simulation and physical variants.

6. Problem domain: Scaling chains of integrators

The first problem domain is the simplest in terms of dynamics and specifications, yet unlike the other domains, 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 the double-integrator is just the basic force equation of Newtonian mechanics, up to a scaling factor. Informally, this problem domain concerns controlling acceleration or a higher order derivative of a point-mass in high-dimensional spaces so as to visit goal regions and avoid obstacles. The precise description is given below.

6.1. **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). The case of $m = 2$ is known as 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 $m = 2$.

6.2. **Specifications.** Task specifications will require visitation of regions while avoiding obstacles. Compared with the specifications for the other problem domains presented in Sections 7 and 8, these can be regarded as a slight extension beyond classical motion planning. There are two forms of specifications:

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

and

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

where $\text{Obs} \subset \mathbb{R}^n$ is the obstacle set, which is represented by a finite union of polytopes. For each i, 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, $\text{init} \in \mathbb{R}^n$. 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.

6.3. **Controller Execution.** Controller execution has the following key components:

- A *problem generation engine* produces random problem instances in a format that uses JSON (`http://json.org`) as container and as demonstrated by the following example:

```
{
"version": 0,
"Xinit": [0, 0, 0, 0],
"goals": [
{ "label": "goal_0", "H": [[-1, 0], [1, 0], [0, -1], [0, 1]],
                     "K": [-6.3594, 9.66338, -2.61279, 3.93709] }
],
"obstacles": [],
"Y": { "H": [[-1, 0], [1, 0], [0, -1], [0, 1]], "K": [1, 10, 2, 10] },
"U": { "H": [[-1, 0], [1, 0], [0, -1], [0, 1]], "K": [1, 1, 1, 1] },
"period": 0.0800622
 }
```

  where $XInit$ is the initial position, the goals and obstacle polytopes are defined by intersections of half-spaces, i.e. $Hx \leq K$, and "period" is the interval (in seconds) used to discretize time.
- A *trial runner* runs the problem generation instance multiple times to generate several trials. Instances are published to the topic `dynamaestro/probleminstance_JSON` The trial runner also subscribes to a topic called `input`, on which it expects inputs. It generates trajectories as solutions of a linear time-invariant control system, where control input is applied using a zero-order hold, i.e., applied constantly during the duration corresponding to the time-discretization specified in the problem instance. Finally, it publishes states to a topic called `state`: an empty state message indicates that trial has ended.
- The form of the controller is left flexible, but it must do the following:
  - listen for problem instances on the `dynamaestro/probleminstance_JSON` topic

– publish control inputs to the `input` ROS topic. These control inputs are vectors of non-zero length corresponding to the input dimension of the system, or empty to declare unrealizability.

6.4. **Evaluation.** There will be a score computed as a weighted sum of the following criteria:

(1) correctness of deciding realizability: wrong answer is worse than no answer.
(2) time between receiving problem instance and declaring answer to #1.
(3) any violation of task formula is worse than not trying to answer the problem instance.
(4) provided all of the above are successfully met (and provided that the problem instance is realizable), synthesis times for increasing:
    (a) number of integrators;
    (b) number of state dimensions;
    (c) [NOT USED 2016] number of goal regions;
    (d) [NOT USED 2016] number of obstacle regions.

    Each of (a) and (b) above will be increased and the time to reach a correct decision will be measured for each controller.
(5) [NOT USED 2016] let the cycle-time be the time required to visit every goal region at least once. The minimum and maximum cycle-times in a trial are recorded.

6.5. **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. 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.

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 common compute hardware, such as through Amazon Web Services (`http://aws.amazon.com/`).

7. Problem domain: Traffic network of Dubins cars

This domain involves navigation in a small network of two-lane roads with vehicles that follow unicycle-like dynamics. Unlike the "scaling chains of integrators" problem (cf. Section 6), now there is an adversary that selects the motion of other cars. The adversary is subject to assumptions such as obeying traffic rules and not parking unfairly at locations that must be eventually reached by the controller.

7.1. **Dynamics and constraints.** All cars including the controlled robot are assumed to have the same rigid body shape and to have the same dynamics. Let $\mathcal{I}$ be the index set for the cars in the network, with r denoting the index of the controlled robot; all cars indexed with $i \in \mathcal{I} \backslash \{r\}$ are regarded as a part of the (adversarial) environment.

The pose of each car $i \in \mathcal{I}$ is specified by (x_i, y_i, θ_i), where $(x_i, y_i) \in \mathbb{R}^2$ is referred to as *position*, and $\theta_i \in S^1$ is referred to as *orientation*. The car's body is a rectangle or a circle, and the position is defined to be at the mean point (center of mass) of the body. Let w be the width of the car – this is identical for each $i \in \mathcal{I}$.

The trajectories of car i are solutions of

$$\dot{x_i} = u_i \cos\theta_i \tag{9}$$

$$\dot{y_i} = u_i \sin\theta_i \tag{10}$$

$$\dot{\theta_i} = \omega_i \tag{11}$$

where the control inputs are constrained as $|u_i(t)| \leq u_{\max}$ and $|\omega_i(t)| \leq \omega_{\max}$.

The workspace is a randomly generated road network constructed as follows. Create a planar graph $G = (V, E)$ in which each vertex v has degree $d_v \leq 4$ (i.e., at most 4 neighbors). Embed this graph in the plane, and expand the edges to have width $4w$ (recall w is the common vehicle width).

For this aspect of the "traffic network of Dubins cars" problem domain, we are able to vary the bounds $u_{\max}$ and $\omega_{\max}$ on the permissible control inputs, the size ($|V|$) and topology ($|E|$) of the road network, and the number of other cars, i.e. $|\mathcal{I}|$.

7.2. **Specifications.** Because there is now an environment that may behave adversarially, task specifications will be of the form $\varphi_{\text{env}} \Rightarrow \varphi_{\text{sys}}$, where φ_{env} is known as the "assumption" and φ_{sys} as the "guarantee." The desired behaviors to be realized by the robot are provided through φ_{sys} and include

(1) obstacle avoidance while repeatedly visiting regions of interest, as in (7);
(2) remaining in the right-lane except when it is blocked;
(3) [NOT USED 2016] stopping at intersections, which are treated as all-ways stops, and then proceeding based on order of arrival.

The other cars will be assumed to follow the same road rules as listed above. However, exceptions (violations) may occur, and thus additional fairness assumptions will be provided. In particular,

$$\bigwedge_i \Box\Diamond \texttt{free}(\text{goal}_\texttt{i}) \tag{12}$$

which provides that other cars will always eventually vacate the i-th goal region. The position of the other cars will be provided when in a predetermined radius, to simulate solving the associated vision problem.

7.3. **Controller Execution.** Controller execution has the following key components:

- A *problem generation engine* produces random problem instances from a configuration file that includes information about the e-agents, in particular their number and namespace. Problem instances are in a format that uses JSON (`http://json.org`) as container and as demonstrated by the following example:

```
{
"version": 0,
"goals": [
[2, 0]
],
"rnd": { "version": 0, "length": 3, "transform": [0, 0, 0],
"shape": [3, 4], "segments": [[0, 0, 1, 0], [0, 0, 0, 1], [0, 1, 1, 1],
[0, 1, 0, 2], [1, 0, 2, 0], [1, 0, 1, 1], [1, 1, 2, 1], [1, 1, 1, 2],
[2, 0, 3, 0], [2, 0, 2, 1], [2, 1, 3, 1], [2, 1, 2, 2], [0, 2, 1, 2],
[1, 2, 2, 2], [2, 2, 3, 2], [3, 0, 3, 1], [3, 1, 3, 2]] }
```

```
, "intersection_radius": 1
}
```

where *rnd* is a road network description consisting of segments, and *intersection_radius* is the distance from the end of a segment at which the vehicle is considered to be in the intersection.

- A *trial runner* runs the problem generation instance multiple times to generate several trials. The problem instances are published on the topic `probleminstance_JSON`. The trial runner subcribes to the model states in topic `gazebo/model_states` and publishes the labeled output to `loutput`.
- The form of the controller is left flexible, but it must do the following:
  - listen for problem instances by subscribing to the `probleminstance_JSON` topic
  - publish control inputs to the relevant topics for the kobuki, e.g. `/ego/mobile_base/commands/velocity` or `/agent0/odom/`.

7.4. **Evaluation.** There will be a score computed as a weighted sum of the following criteria:

(1) time to begin, i.e., duration between being first given the problem instance description and requesting start of the play.
(2) number of collisions.
(3) fraction of goals visited
(4) number of departures from lane when other vehicles are moving with positive speed.
(5) fraction of trial duration not on any road.
(6) fraction of trial duration outside of correct lane (moving against the flow of traffic) when doing so is not necessary to continue to progress (i.e. no e-agent is blocking the lane).
(7) [NOT USED 2016] cycle-time as defined for the problem domain integrator_chains.
(8) all of the above with varying:
    (a) numbers of e-agents;
    (b) size of the road net;
    (c) [NOT USED 2016] lane width.

7.5. **Implementation plan for the challenge.** We are currently exploring feasibility of simulation using CloudSim (`http://cloudsim.io`), which was developed for running Gazebo-based simulations (`http://gazebosim.org`) as part of the "virtual" component of the current DARPA challenge (`http://www.theroboticschallenge.org`).

Besides simulation, we plan to construct a reduced size form of this challenge domain for running on-site. Both the robot to be controlled and the adversarial vehicles will be based on Kobuki mobile bases (`http://yujinrobot.com/eng/?portfolio=kobuki`). Because all problem-relevant motion will occur within a fixed plane, high-speed overhead pose tracking is possible using a single camera mounted on the ceiling. We hope to install such a system on-site to provide ground truth positions for scoring purposes, even if the data is not available to competitors during the challenge.

8. Problem domain: Factory cart clearing

8.1. **Summary.** This section is non-normative.

This problem domain broadly concerns pick and place tasks, expressed using LTL contracts, as described below. The key challenges explored in this problem domain will include uncertainty in the shape and dynamics of the end-effector, as well as objects that are being grasped, and collision-avoidance with moving obstacles that come into the workspace, including humans. The focus is on picking up and moving objects around, but will not involve difficult grasping problems, like deformable surfaces. The end-effector will be something easy to use, like a magnet or some suction mechanism, such as a Universal Jamming Gripper [2].

The scenario will be as follows. The robot is required to process objects laid out on a wheeled cart that is brought to it and sort them into bins. Carts arrive after the robot indicates that it is ready, and carts are removed once a robot declares that it is done. Scoring will be based on how many carts can be processed correctly (described below) in a fixed amount of time.

The top surface of the cart will be unwalled but divided into an object placement grid. The size of the grid will vary; an example is a 5×4 arrangement, which has 20 parts total. Each object can occupy multiple cells in the grid. Each object has a priority level (possibly shared with other objects) which determines the order of processing. Processing involves picking up the object and waiting for a signal to determine which of a set of bins to deposit it in. Polygonal bounding boxes for the objects will be provided after a small delay once the cart arrives, to simulate solving the vision problem.

The gripping process is automatic and regarded as a subroutine, so that teams need only place the gripper within a certain region above the object then request a grasp, which may fail but will eventually succeed provided repeated attempts. There will be a clearance requirement for the arm and end-effector, so that it must maintain a minimum distance from obstacles throughout the trial. While different kinds of end-effectors will not be provided, a disc-shaped mount will be placed on the wrist to effect varying shape constraints.

The cart will be serviced in collaboration with some other agent, e.g., a human or another robot. When the partner is busy with a part of the cart, the corresponding cells and the space above them are unavailable, which may be indicated by literally covering them or shining a light from underneath those cells. The "clearance requirement" would then mandate that a safe distance is kept from this other agent and the part of the cart that it is using.

In summary, the salient features are that:

(1) carts are taken to and from the robot.
(2) the robot must sort items on the cart into bins.
(3) the decision of which bin to use for each item is given.
(4) a human or another robot may also use the cart and contribute to the sorting task.
(5) the details of grasping are abstracted so that teams merely need to request for a grasp to occur once sufficiently near the object.
(6) safe distances from obstacles must be maintained throughout.

8.2. **Dynamics and constraints.** For the tasks in this problem domain, the robot workspace is a three-dimensional Euclidean space.

TODO: dynamics of the arm

8.3. **Specifications.** The specified scenario will be as follows. The robot is to process objects laid out on a cart that is brought to it and sort them into bins. There are several possible variants:

- carts arrive after the robot indicates that it is ready
- cart is removed once a robot declares that it is done.
- carts arrive at a randomly time-varying rate, whether or not the robot is ready for the next.

A part of the specification will include space tolerances, e.g., the size of the cart, the size of the grid on it in which parts may be placed, and minimum clearance distances to be maintained during performance of the task.

8.4. **Implementation plan for the challenge.** Performance for a trial will be assessed by how many carts are processed within the time limit. We are exploring several simulation environments for this domain, including the following.

- Gazebo (`http://gazebosim.org/`)
- OpenGRASP (`http://opengrasp.sourceforge.net/`)
- Peter Corke's Robotics Toolbox (`http://petercorke.com/Robotics_Toolbox.html`)
- Klamp't (`http://www.iu.edu/~motion/klampt/`)

We are also exploring the possibility of a physical testbed on-site.

9. Evaluation

In this section, methods of evaluation are presented.

9.1. **Dynamic, trial-based, opaque.** 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. Submitted controllers are required to use a communication interface defining control commands, sensor values, state labels, and other details of input and output appropriate for the problem domain. Controllers are not otherwise constrained, and thus are described as *opaque*.

9.2. **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. Among the problem domains that are available at the time of writing, this applies applies only to Scaling chains of integrators (cf. Section 6).
- declare a minimum execution time required to demonstrate correctness.

9.3. **Scoring.** Entries 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, 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.
- For the network of cars domain, scoring will be on the basis of the time required to synthesize *and* execute a controller visiting a pre-specified set of regions, while satisfying specified safety conditions, in the presence of adversarial agents. For fairness, each team's entry will be run against the same set of environment strategies and in the same set of road networks.
- For the manipulation domain, scoring will be on the basis of the number of carts processed. The same set of carts will be provided to each team.

9.4. **Scheduling Trials.** Teams will not be allowed to modify their solutions once the trials for a specific domain have begun. The order of executing controllers will be randomized for each round of trials.

10. Drafts of evaluation methods

This section is non-normative.

Static evaluation methods prove that a submitted solution is correct without running it, for example, by requiring part of the solution to be expressed using Promela and thus amenable to model-checking by Spin, or by requiring controllers to linear or polynomial systems that have reachability or safety that can be demonstrated by certificates (e.g., using SOS programming).

Acknowledgments

The "factory cart clearing" problem domain is based on a practical scenario described by Brian Benoit in Oct 2014 (private communication).

References

[1] Key words for use in RFCs to indicate requirement levels, March 1997. `https://tools.ietf.org/html/rfc2119`.

[2] J. R. Amend, E. Brown, N. Rodenberg, H. M. Jaeger, and H. Lipson. A positive pressure universal gripper based on the jamming of granular material. *IEEE Transactions on Robotics*, 28(2):341–350, 2012.

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