2009 33rd Annual IEEE International Computer Software and Applications Conference

Toward the Design of Robotic Software with Verifiable Safety

Chih-Hong Cheng*, Markus Rickert*, Christian Buckl†, Edward A. Lee‡, Alois Knoll*
*Department of Informatics, TU Munich, Germany
†Fortiss GmbH, Germany
‡EECS, UC Berkeley, USA

1. Extended Abstract

In this abstract, we mention some challenges and preliminary solutions toward designing robotic software with verifiable safety. We use a 2D planar robot as example (fig. 1-a), which has an arm with fixed or moving base. Also, computational stereo for precise (or imprecise) location of the object or obstacle is assumed. Our simple scenario is to perform object retrieval, but safety requirements should be assured, for example, to avoid moving obstacles.

1.1. Challenge A: Definition of Safety

In the verification community, safety is the answer of the state reachability problem, but in robotics the concept of safety is coarse. Consider a case where during robot operation, an obstacle (e.g., a human) intentionally hits the robot, while the robot has tried its best to perform avoidance. From the view of verification safety, the previous case has two interpretations: (1) The environment behavior is underspecified. (2) It is not the responsibility of the robot. Two suggestions are thus proposed for the use of consensus languages in both verification and robotic communities.

Suggestion 1: For safety design in robotics, if possible, we should try to specify some behavior of the changing environment using negations to avoid underspecification[1].

Suggestion 2: Safety is based on responsibilities - either the robot or the environment should be responsible for system unsafety[2]. Verification only guarantees the safety within the robot's responsibilities.

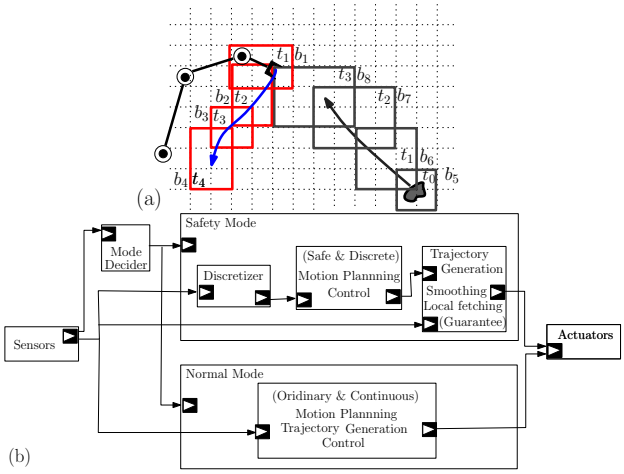

Figure 1. A robot performing obstacle avoidance (a), and the conceptual robot controller (b).

1.2. Challenge B: Predictable and Verifiable Controllers and Algorithms

The primary gap between robotics and verification is that continuous time and an infinite variable domain are widely used in robotics, but they lead to intractable results in verification. Nevertheless, safety can be checked by overapproximation[3], and the new problem is to design the software (controller and algorithm) such that overapproximation (e.g., trajectories) can be achieved easily. We think of two approaches.

1.2.1. Method 1: Overapproximation of existing algorithms. We only mention the main challenge for this approach: it is difficult to establish the mapping between abstract and concrete domains. Consider simplified cases where the robot understands the current obstacle position $\tilde{x}$, its own position $\tilde{y}$, and generates

Email:{chengch,rickert,buckl,knoll}@in.tum.de, eal@eecs.berkeley.edu.

1. A sample behavior can be described as follows: the object will **NOT** occur within the range of 2 cm from the end-effector without being previously sensed by the sensor in $1ms$.

2. Back to footnote 1, when in operation the object violates its rule and hits the robot, it is **NOT** the robot's responsibility. But for any other hitting, it **IS** the robot's responsibility.

3. For example, if we overapproximate the trajectory of the end-effector as a combination of boxes moving over time, our hand will not touch the end-effector if we do not touch the box.

0730-3157/09 $25.00 © 2009 IEEE
DOI 10.1109/COMPSAC.2009.89

the next robot position $\tilde{y}'$; $\tilde{x}$, $\tilde{y}$, and $\tilde{y}'$ are sets of parameters in floats. Let $\mathcal{D}_x$ represent the domain of variable x. If verification operates on a finite integer domain, then model checking should work on abstract values $\tilde{x}_{abs}$, $\tilde{y}_{abs}$, and $\tilde{y}'_{abs}$ of $\tilde{x}$, $\tilde{y}$ and $\tilde{y}'$. To do this, we must find two binary relations $\mathcal{R}_x$, $\mathcal{R}_y$ over $\mathcal{D}_{\tilde{x}} \times \mathcal{D}_{\tilde{x}_{abs}}$ and $\mathcal{D}_{\tilde{y}} \times \mathcal{D}_{\tilde{y}_{abs}}$, and an abstract algorithm $f_{abs}(\tilde{x}_{abs}, \tilde{y}_{abs})$, such that $\forall \tilde{x}, \tilde{y}, \tilde{x}_{abs}, \tilde{y}_{abs}$, if $(\tilde{x}, \tilde{x}_{abs}) \in \mathcal{R}_x, (\tilde{y}, \tilde{y}_{abs}) \in \mathcal{R}_y$, then $(\tilde{y}', \tilde{y}'_{abs}) = (f(\tilde{x}, \tilde{y}), f_{abs}(\tilde{x}_{abs}, \tilde{y}_{abs})) \in \mathcal{R}_y$. Since $f(\tilde{x}, \tilde{y})$ is very complicated in general with continuous mathematical operations (e.g., sine or cosine functions), the above goal is almost unlikely to be achievable.

1.2.2. Method 2: Discretized algorithm design with continuous smoothing. Intuitively, this approach reverses the process. We design a discrete controller (thus verifiable) which implements the algorithm $f_{abs}(\tilde{x}_{abs}, \tilde{y}_{abs})$. Every input is translated into discretized move for decision making, then the real output is further smoothed without losing the guarantee of the location offered by the discretized controller.

For the example in fig. 1-a, we illustrate our approach with controller in fig. 1-b. Two modes are used based on criticality. If an obstacle is far from reach, then ordinary continuous control and motion planning algorithms are applied for optimality. Safe mode with discretized controller is used for obstacle avoidance while tries its best to fetch the object. Now we consider the safety case.

1) First, sensor data are further processed into discretized values (boxes) by the discretizer, and the controller (or algorithm) makes the decision based on boxes. For the controller, at time t_1 it locates itself in the box b_1[4], and senses the stone in box b_6. With the prediction that the stone will proceed over box b_7 and b_8, it decides that it should proceed with the following boxes b_2, b_3, and b_4 for time t_2, t_3, and t_4. The trajectory is then generated based on the box-timing constraints; the robot should generate its trajectory to satisfy the constraint that for all i, at time t_i, it is within box b_i.
2) After the decision is made, (for example, to move from box b_1 to b_4 with timing constraints), the smoothing component generates the path. Since computation for previous steps needs time, the smoothing component should be able to calibrate and compensate the effect on the elapse of time.

In this way, robot safety can be partially guaranteed by verifying the discretized controller; the actual system behavior is constrained by over-approximation with box elements[5].

4. At t_1 only box b_1 is drawn containing the robot arm and the end-effector for simplicity reasons.

2. Preliminary Experiments

Some concepts of discretized control for robotic systems are preliminarily prototyped using the Ptolemy II software [3], and the iRobot Create is used as the implementation platform. For details, see [1][6]. The constructed system is simple: it does not have smoothing abilities, and the environment is static. With accelerometers and edge-detection sensors attached, the verified controller is guaranteed to climb up hill without falling off the edge.

3. Conclusion and More Challenges

Our contribution is as follows:

1) We try to mediate the gap between robotic safety and verification safety by proposing the concept of responsibility.
2) We tailor freely-designed methodologies in robotics to fit the theoretical limit of verification.

We outline several unconsidered challenges.

1) In 3D, even with discretized space and over-approximation, the state space is still too large for verification engines.
2) In robotics, sampling-based algorithms are commonly applied to reduce computational complexities. For these algorithms, introducing probabilistic model checking is needed.
3) An appropriate overapproximation for the body of the robot in discretized space can be challenging considering granularity[7]. Also, modeling dynamic behavior of the environment is difficult.

References

[1] C. Cheng, T. Fristoe, and E.A. Lee. Applied Verification: the Ptolemy Approach. Technical report, UCB/EECS-2008-41, UC Berkeley.

[2] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of robot motion*. MIT press, 2005.

[3] J. Eker, J.W. Janneck, E.A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming Heterogeneity: the Ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, 2003.

5. Of course, it requires further assumptions regarding the correctness of other components (e.g., predictable timing).

6. Video: http://www.youtube.com/watch?v=4JXj7oJyLg8

7. Our work in iRobot applies C-space approach [2] such that the round robot can be shrunk into single point.