A ROS (Robot Operating System) shim for Coq

Abhishek Anand

Coq Workshop 2015

Introduction Cyber-Physical Systems (CPS) such as ensembles of robots can be thought of as distributed systems where agents might have sensing and/or actuation capabilities. In fact the Robot Operating System (ROS) [4] presents a unified interface to robots where subcomponents (e.g. sensor, actuator, controller software) of even a single robot are represented as nodes that communicate with other nodes using asynchronous message passing. In recent work, we developed ROSCoq [1], a framework for developing certified Coq programs for robots. ROSCoq subsystems communicate using asynchronous message passing, as they do in ROS. We extend the logic of events [5] to enable holistic reasoning about the cyber-physical behavior of robotic systems. Software components are specified as message handlers written in Coq [1, Sec. 4.2]. The behavior of the physical world (e.g. Newton's laws) and associated devices (e.g. sensors, actuators) are specified axiomatically [1, Sec. 2, 4.1]. For reasoning about physics we use and extend CoRN's [3] theory of constructive real analysis. Instead of floating points, our Coq programs use CoRN's exact, yet fast computations on reals, thus enabling accurate reasoning about such computations.

As an application, we specified the behavior of an iCreate robot[1]. Our specification captures many real world imperfections. For example, it allows for imperfect actuation and delayed reaction to commands. We wrote a Coq program which receives requests to navigate to specific positions and computes appropriate commands for the robot. We proved correctness properties about this system [1, Sec. 5]. Using a ROS shim for Coq, we ran the program on the robot and provided even experimental evidence of correctness.

We propose to start with a brief talk describing ROSCoq and then discuss ongoing work towards generalization of the ROSCoq shim in several directions, e.g. supporting a richer collection of ROS-supported robots, allowing access to randomness. We also wish to discuss ways to reduce the Trusted Computing Base (TCB) of the shim and to make it more efficient.

Towards a general, trustworthy and efficient shim As mentioned before, ROS presents a unified interface to robots where subcomponents (e.g. sensor, actuator, controller software) of even a single robot are represented as nodes that communicate with other nodes using asynchronous message passing. ROS

[1] http://www.irobot.com/About-iRobot/STEM/Create-2.aspx

provides a collection of standard message types which are used by many devices. For example, the message type `geometry_msgs.Twist`[2] essentially represents a product of 3D-vectors, which are supposed to be interpreted by robots as linear and angular velocity, respectively in 3D-space. Typically when a robot receives such a message, it adjusts the motors to achieve the linear and angular velocity specified in the message. Similarly, there are message types used by devices to send sensor readings, typically to a software controller. The main job of language bindings for ROS is therefore to enable sending and receiving of ROS messages on specified topics.

The current ROSCoq shim is written in ROSJava and executes Coq programs by communicating with coqtop, the Coq toplevel. When the shim receives a message, it converts it to the corresponding Coq format, executes a stateful message handler [1, Sec. 4.2] in Coq to compute which messages have to be sent in response, converts those messages back to ROSJava format and sends them at times specified by the Coq message handler. ROSJava is responsible for serialization of messages before sending and de-serialization after receiving. For a more trustworthy shim, we wish to instead define the (de-)serialization functions entirely in Coq. We could then at least prove that serialization followed by de-serialization is equivalent to the identity operation. Also, not all ROS message types have yet been defined in Coq. We wish to provide a way to automatically import ROS message types to Coq. Due to similarities between Coq and Haskell's datatypes, we hope to be able to reuse parts of the roshask implementation [2]. Finally, we wish to discuss the possibility of implementing the shim in OCaml instead of Java. This might result in a more efficient shim that directly accesses Coq's internal representation instead of parsing coqtop's textual output. Also, we will discuss our new type of Coq message handlers which allow them to use random numbers supplied by the shim. Many robotic programs need randomness for robustness.

References

[1] Abhishek Anand and Ross Knepper. *ROSCoq: Robots powered by constructive reals*. Submitted to ITP, decision due on May 15, 2015. 2015. URL: http://www.cs.cornell.edu/~aa755/ROSCoq/Confidential.pdf.

[2] A. Cowley and C. J. Taylor. "Stream-oriented robotics programming: The design of roshask". In: *IROS*. IEEE, 2011, pp. 1048–1054.

[3] R. Krebbers and B. Spitters. "Type classes for efficient exact real arithmetic in Coq". In: *LMCS* 9.1 (Feb. 14, 2013).

[4] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 2009, p. 5.

[5] N. Schiper, V. Rahli, R. V. Renesse, M. Bickford, and R. L. Constable. "Developing correctly replicated databases using formal tools". In: *DSN*. IEEE, 2014, pp. 395–406.

[2] http://docs.ros.org/api/geometry_msgs/html/msg/Twist.html