My Types Depend on My Robots

My Robots Depend on My Types

Anthony Cowley Camillo J. Taylor

GRASP Laboratory, University of Pennsylvania

{acowley,cjtaylor}@seas.upenn.edu

Abstract

Robotics programmers stand to gain a great deal from expressive type systems. The types that help traditional software developers design systems and catch bugs at compile time may be extended to encode operational constraints of physical systems. Statically checking these constraints provides verification that actuation plans adhere to electromechanical system limitations, as well as mandated standard operating procedures. Rejecting programs that can't possibly work, or may violate important constraints, can save roboticists from experiencing costly bugs, and mitigate the dangers of risky autonomous behaviors.

Categories and Subject Descriptors I.2.9 [*Autonomous vehicles*]

Keywords Linear Logic, Dependent Types, Robotics

1. Logically Safe

Current work applying rich types has targeted platforms such as that shown in Figure 1. This mobile robot is driven by several low-level, closed-loop controllers regulating robot position, and a high-level scripting interface used to specify construction plans. The scripting interface is implicitly imperative in that the core premise is that each action mutates the work space.

Linear logic provides a language to make the finitude of resources explicit [1]. In order to provide proof as a platform, we fully automate discharge of the proof obligations associated with each primitive action via Coq tactics [2]. An example commanding the robot to place one block atop another demonstrates the minimal Coq accessorizing needed to adapt an imperative syntax to a platform that verifies conservation of gripper resources and collision freedom of placed blocks.

```
Example two_stack_test :
  [env ∅, empty] ⊢ (table X ⊗ on Y X) ⊗ ⊤.
Proof.
  get_from_cache.
  put_on_table X.
  get_from_cache.
  stack Y on X.
  isolate [table X, on Y X].
Qed.
```

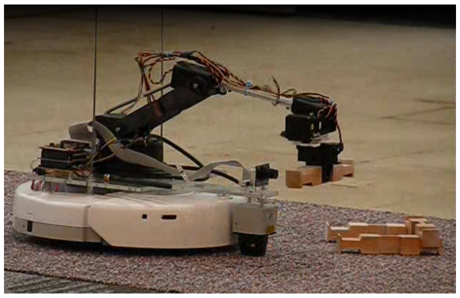
Figure 1. Mobile manipulator platform stacking blocks.

2. Current Work: Trustworthy Intelligence

Deductive verification of scalable protocols [3] and reactive behaviors [4] suggest a future where roboticists may more confidently implement reactive behaviors that carry risk. Beyond the familiar military and disaster rescue scenarios, "risky" operations for robots may be as mundane as handling an open container of liquid. In order to protect sensitive electronics, engineers can enforce that a safety protocol is followed by high-level behaviors. We are currently engaged in exploring the mechanisms by which formal verification may be integrated into experimental robotics.

References

[1] J. Power and C. Webster. Working with linear logic in coq. In *The 12th International Conference on Theorem Proving in Higher Order Logics*, 1999.

[2] A. Cowley and C. J. Taylor. Towards language-based verification of robot behaviors. In *Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[3] M. Bozzano and G. Delzanno. Automated protocol verification in linear logic. In *Proceedings of the 4th ACM SIGPLAN international conference on Principles and practice of declarative programming*, PPDP 2002.

[4] A. Jeffrey. Ltl types frp. In *The Sixth ACM SIGPLAN Workshop Programming Languages Meets Program Verification (PLPV 2012)*, 2012.