Demo: Interactive Robot Transition Repair

Demonstration

Jarrett Holtz   Arjun Guha   Joydeep Biswas
University of Massachusetts
Amherst, MA, United States

ABSTRACT

Complex robot behaviors are often structured as state machines, where states encapsulate actions and a transition function switches between states. Since transitions depend on physical parameters, when the environment changes, a roboticist has to painstakingly readjust the parameters to work in the new environment. In this demo we present *Interactive SMT-based Robot Transition Repair* (SRTR): instead of manually adjusting parameters, we ask users to identify a few instances where the robot is in a wrong state and what the right state should be. A lightweight automated analysis of the transition function's source code then 1) identifies adjustable parameters, 2) converts the transition function into a system of logical constraints, and 3) formulates the constraints and user-supplied corrections as a MaxSMT problem that yields adjustments to parameter values. This demo uses a simulated RoboCup Small Size League platform, allows users to correct faulty behaviors, and then uses SRTR to adjust parameters automatically.

KEYWORDS

Humans and Agents; Learning and Adaptation

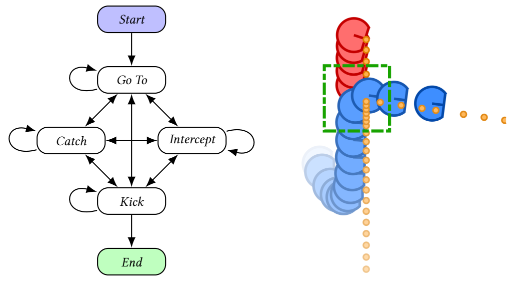
ACM Reference Format:
Jarrett Holtz, Arjun Guha, and Joydeep Biswas. 2018. Demo: Interactive Robot Transition Repair. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 2 pages.

1 INTRODUCTION

Complex robot control software is typically structured as a state machine, where each state encapsulates a feedback controller. Even if each state is correct, the transitions between states depend on parameters that are hard to get right, even for experienced roboticists. It is very common for parameter values to work in simulation but fail in the real world, to work in one physical environment but fail in another. For example, Figure 1 shows the trajectory of a robotic soccer player as it tries to kick a moving ball. A very small change to its parameter values determines whether or not it succeeds.

Even a simple robot may have a large parameter space, which makes exhaustive-search impractical. Moreover, robot performance is usually non-convex with respect to parameter values, which makes general optimization techniques susceptible to local minima. For some cases, there exist calibration procedures to adjust parameters automatically (*e.g.,* [3, 7]), but these are not general

This work is supported in part by AFRL and DARPA under agreement #FA8750-16-2-0042, and by NSF grant CCF-1717636.



(a) Attacker state machine **(b) Execution traces**

Figure 1: A robot soccer attacker a) state machine, with b) successful (blue) and unsuccessful (red) traces to *Intercept* a ball (orange) and *Kick* at the goal. The green box isolates the error: the successful trace transitions to *Kick*, the unsuccessful trace remains in *Intercept*.

procedures. Other work on adjusting state machines for new environments [1, 6] synthesizes new state machines or transitions in state machines instead of correcting existing transitions. Therefore, roboticists usually resort to adjusting parameters manually—a tedious task that can result in poor performance.

We make the following observations: a roboticist debugging a robot can identify when something goes wrong, and what should have happened. When the robot control software is structured as a state machine, this corresponds to identifying when the robot is in the wrong state and what the correct state should be. This is a *partial specification* of expected behavior: the roboticist does *not* need to enumerate a complete sequence of states, the parameters to adjust, how to adjust them, or even identify all errors.

Based on these observations, we have developed a semi-automated procedure for adjusting the parameters of robot state machines, which we call *SMT-based Robot Transition Repair* (SRTR) [4]. We demo SRTR using behaviors from a RoboCup Small Size League (SSL) team. SRTR execution trace of the transition function for a faulty behavior. Users provide corrections to this execution trace, and SRTR converts the set of corrections and the transition function into a logical MaxSMT formula. A solution to this MaxSMT formula is the minimal adjustment to the parameters that satisfies the maximum number of corrections. We then demonstrate the RSM performance after SRTR adjustment.

2 APPROACH

The SRTR algorithm has four inputs: 1) the transition function, 2) a map from parameters to their values, 3) an execution trace, and 4) a set of user-provided corrections. The result of SRTR is an adjustment to parameters that maximizes the number of corrections satisfied and minimizes the changes to the input parameters.

SRTR has three major steps. 1) It parses the transition function code and converts it to an abstract syntax tree for repair. 2) It uses a lightweight program analysis to identify parameters that can be repaired, and for each user-provided correction, it partially evaluates the transition function for the inputs and variable values at the time of correction, yielding *residual transition functions*. 3) Finally, it uses the residual transition functions to formulate an optimization problem for an off-the-shelf MaxSMT solver [2]. The solution to this problem is an adjustment to the parameter values.

Execution Trace and Corrections. To abstract away language-specific details of our procedure, we first parse the source code and convert it into an idealized imperative language that only has features essential for transition functions. We use this language to record the execution trace of the transition function. A trace element records the values of sensor inputs, ordinary variables, and the state at the start the time-step. A user-provided *correction* specifies the expected state at the end of the time-step. To adjust parameters, SRTR establishes constraints that relate the initial state, the expected output state, and the trace elements.

Program Analysis and Partial Evaluation. SRTR needs to translate the transition function into a logical formula for the SMT solver. However, SMT solvers do not have decision procedures for non-linear arithmetic and trigonometry, which are both very common in robot transition functions. *Partial Evaluation* is a technique that specializes programs to work on specific inputs. For example, if we partially evaluate the program $\cos(x) + y$ for $x \mapsto 0$, we get the program $1 + y$. In general, partial evaluation first substitutes identifiers with concrete values and then simplifies the program as much as possible. SRTR uses a canonical partial evaluator for transition functions.

We refer to the specialized transition function for a particular trace element as a *residual transition function*. To calculate the residual transition function we first partially evaluate the transition function with respect to all the sensor inputs and variable values recorded in the trace element. This eliminates most functions that the solver cannot represent. However, it is possible for the transition function to use a parameter in a context that the solver cannot represent. To address this, we use a lightweight program analysis to calculate the set of parameters that cannot be repaired and partially evaluate once again with respect to their concrete values.

2.1 Transition Repair as a MaxSMT Problem

To formulate a MaxSMT problem SRTR uses three steps 1) It translates each correction into an independent formula. A solution to this formula corresponds to parameter adjustments that satisfy the correction. 2) It combines the formulas from the previous step into a single formula with independent weights for each sub-formula. 3) Finally, we formulate a MaxSMT problem that minimizes the magnitude of adjustments and the weight of violated sub-formulas.

To transform a single correction into a formula, SRTR 1) calculates the residual transition function, 2) determines the repairable parameters, and 3) produces a formula with adjustable variables that correspond to the adjustments for each repairable parameter. For multiple corrections we iteratively build a conjunctive formula, where the weight is the cost of violating a clause.

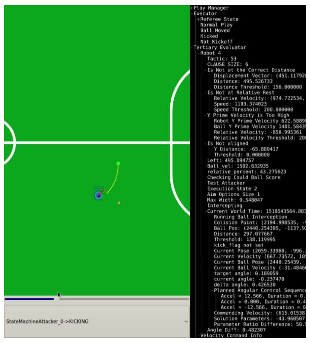
Finally, SRTR invokes the MaxSMT solver with the conjuctive formula of all corrections. SRTR directs the solver to minimize the sum of weights and the sum of the magnitude of parameter changes, and returns adjustments to repairable parameters.

3 DEMONSTRATION

We demonstrate SRTR by repairing Robocup SSL behaviors using corrections provided by conference participants as follows:

(1) We provide participants a basis for identifying failures by demonstrating the correctly functioning behavior.
(2) We demonstrate the complexity of manual parameter tuning by providing participants with an example of the transition function and parameters.
(3) We show the effect of poor parameter tuning by running the behavior with poorly modified parameters.
(4) We ask participants to identify and correct failures using an interactive log viewer (Figure 2). This log viewer visualizes the field and robot behavior, displays a text log of internal behavior state, the trace of the transition function, and allows control of playback for providing corrections.
(5) We use SRTR to find new parameter values using participant-provided corrections.
(6) We show that SRTR-adjusted parameters perform well.

A video of this demo is available online [5] and a full explanation and performance analysis of SRTR can be found in the full paper [4].



```
+Play Manager
-Executor
  +Referee State
   Normal Play
   Ball Moved
   Kicked
   Not Kickoff
-Tertiary Evaluator
  -Robot 4
     Tactic: 53
     CLAUSE SIZE: 6
    -Is Not at the Correct Distance
       Displacement Vector: (451.11792
       Distance: 495.526733
       Distance Threshold: 156.000000
    -Is Not at Relative Rest
       Relative Velocity: (974.722534,
       Speed: 1193.374023
       Speed Threshold: 200.000000
    -Y Prime Velocity is Too High
       Robot Y Prime Velocity 622.5889
       Ball Y Prime Velocity 1481.5843
       Relative Velocity: -858.995361
       Relative Velocity Threshold: 20
    -Is Not aligned
       Y Distance: -65.880417
       Threshold: 0.000000
     Left: 495.094757
     Ball_vel: 1502.632935
     relative_percent: 43.275623
     Checking Could Ball Score
     Test Attacker
     Execution State 2
     Aim Options Size 1
     Max Width: 0.548047
     Intercepting
    -Current World Time: 1518543564.883
       Running Ball Interception
       Colision Point: (2194.998535, -
       Ball Pos: (2440.254395, -1137.9
       Distance: 297.077667
       Threshold: 138.119995
       kick_flag not set
       Current Pose (2059.33960, -996.
       Current Velocity (667.73572, 10
       Current Ball Pose (2440.25439,
       Current Ball Velocity (-31.4846
       target angle: 0.189059
       current angle: -0.237470
       delta angle: 0.426530
      -Planned Angular Control Sequence
         Accel = 12.566, Duration = 0.
         Accel = 0.000, Duration = 0.4
         Accel = -12.566, Duration = 0
       Commanding Velocity: (615.01538
       Solution Parameters: -43.960507
       Parameter Ratio Difference: 50.
     Angle Diff: 0.482307
 -Velocity Command Info
```

Figure 2: The SRTR tool. Main panel visualizes world state, right panel is a log of internal state, and bottom drop down box selection of desired transition.

REFERENCES

[1] Aijun Bai and Stuart Russell. 2017. Efficient Reinforcement Learning with Hierarchies of Machines by Leveraging Internal Transitions. In *ICRA*.
[2] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. 2015. νZ-an optimizing SMT solver. In *TACAS*.
[3] Jarrett Holtz and Joydeep Biswas. 2017. Automatic Extrinsic Calibration of Depth Sensors with Ambiguous Environments and Restricted Motion. In *IROS*.
[4] Jarrett Holtz, Arjun Guha, and Joydeep Biswas. 2018. Interactive Robot Transition Repair with SMT. In *IJCAI-ECAI*.
[5] Jarrett Holtz, Arjun Guha, and Joydeep Biswas. SRTR Website. https://amrl.cs.umass.edu/index.php?id=srtr. Accessed Apr 30 2018. (2018).
[6] Kai Weng Wong, Rüdiger Ehlers, and Hadas Kress-Gazit. 2014. Correct High-level Robot Behavior in Environments with Unexpected Events. In *RSS*.
[7] Xiangdong Yang, Liao Wu, Jinquan Li, and Ken Chen. 2014. A Minimal Kinematic Model for Serial Robot Calibration Using POE Formula. *RCIM*.