Verifying Plans and Scripts for Robotics Tasks Using Performance Level Profiles

Alexander KOVALCHUK, Shashank SHEKHAR, and Ronen I. BRAFMAN

1. BROAD MOTIVATION

- **Safety** and **reliability** are very important if one wishes to deploy autonomous robots in an unstructured environment
- The goal of this work is to develop methods for verifying plans and controllers that activate robotics code

2. OUR CONTRIBUTIONS

(I) OUR 3-STEP APPROACH

1. Model the behaviors of implemented robotics code/skills
   - We used Performance Level Profiles (PLPs) - an action description language geared for robotics
2. Build planners that can take these PLPs as input and build plans from them
3. Build tools for verifying these plans

(II) OUR CONTRIBUTIONS

1. We give *transnational semantics* to PLPs
   - We provide a mapping between different PLP classes and Probabilistic Timed Automata (PTAs)
2. **Automated Mappings**
   - PLPs $\longrightarrow$ PTAs
   - Controllers + their PLPs $\longrightarrow$ a "large" PTA
3. We test the scalability of the compilation and query answer process, and demonstrate it on a robotics case study

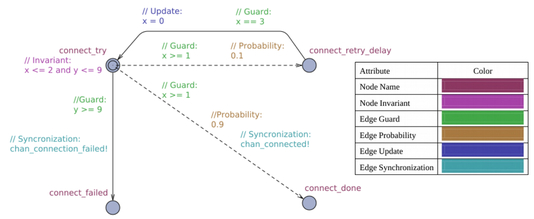
3. BACKGROUND

(I) PERFORMANCE LEVEL PROFILES (PLPS)

- PLPs are an action specification language designed to capture diverse aspects of the intended effect and the run-time behavior of robotics code.
- There are four PLP classes:
  Achieve, Maintain, Observe, and Detect
- It includes some new constructs like run-time distributions, progress measures, update frequency, etc.

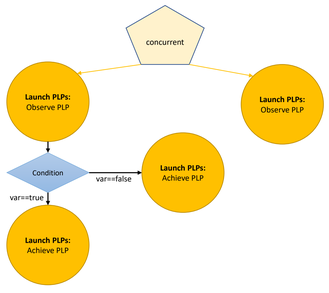
(II) PROBABILISTIC TIMED AUTOMATA (PTA)

PTA is an FSM enhanced with probabilistic transitions and clocks



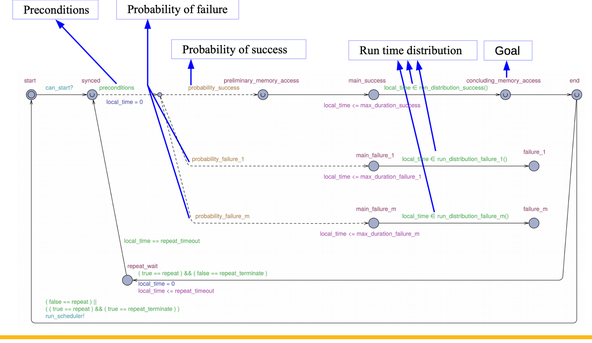
(III) CONTROL GRAPHS (CGS)

A graphical language for describing controllers that schedule the execution of existing code modules



Contain construction for sequential and parallel execution, conditions, and probabilistic choice

4. PLP SEMANTICS



- All logical conditions of this PLP become PTA's **guard** conditions, e.g., *preconditions*
- Exogenous events may affect its execution, and they are modeled as invariant conditions

5. VERIFICATION PROCESS

- CG + its components' PLPs are **MAPPED** to a "large" PTA
- This PTA (a PTAs set) is described in the input format of UPPAAL – *a well known model checker that supports PTAs, too*
- Queries about different properties of this control graph are answered using UPPAAL

6. EVALUATION

Evaluated system's **performance** and **scalability**

(I) TEST CASES

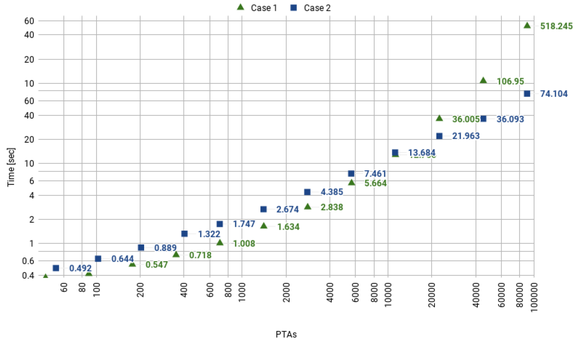
CASE 1 - A CG schema that contains all types of control nodes and is more challenging to compile

CASE 2 - A simple CG with long sequences of PLPs that is more challenging to query

(II) EXPERIMENTS

(a) Average Compilation Time (Phase 1)

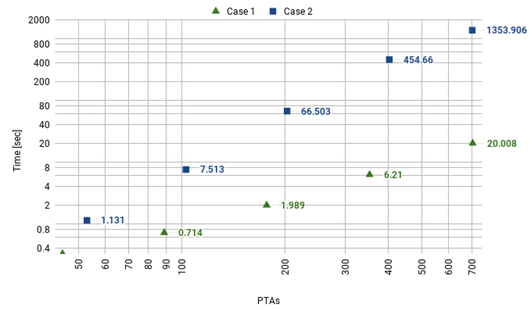
Case 1: GREEN △ **Case 2:** BLUE 2



(b) Answering Queries (Phase 2)

- Ask queries about its **temporal properties** to UPPAAL
- **Query Types**: 1. Path Existence 2. Probability
- **UPPAAL's response time**: depends on the query + properties of the PTAs graph

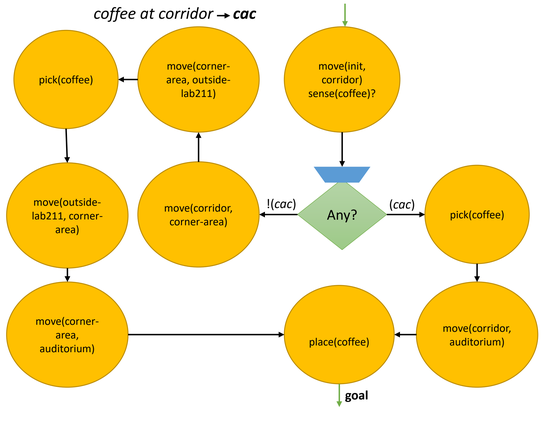
(c) Average Time for Probability Query (Phase 2)



- *Verifying* controller properties *offline* is *realistic*
- (**Up to** $\approx$ **200 PTAs**) we can verify plans online

(III) CASE STUDY (DELIVERY ROBOT)

Scenario: Robot executes this control graph and UPPAAL is used to study its run-time distribution



E.g. Query: Probability that the person will get the coffee before it gets cold? (i.e., within $\leq$ 25 units)

Response: *probability* $\approx$ 0.9 & *resp. time* $\approx$ 15s

ACKNOWLEDGEMENTS: Supported by ISF Grants 1651/19 and 1210/18, by the Israel Ministry of Science and Technology Grant 54178, and by the Lynn and William Frankel Center for Computer Science.