Formal Modeling of Testing Software for Cyber-Physical Automation Systems

Igor Buzhinsky, Cheng Pang, Valeriy Vyatkin

igor.buzhinsky@gmail.com

DIAS Workshop, Helsinki

August 21, 2015
Introduction

- Cyber-physical systems can be modeled formally in a closed loop
- That is, the plant’s and the controller’s models are connected with sensor and actuator connections
- A known way of modeling such systems is the formalism of net condition/event systems (NCES), which are based on Petri nets
- Such models were previously used for verification, and we present a framework which utilizes them for testing
Suitable for modeling:

- Cyber-physical systems
- In particular, distributed ones
- IEC 61131 and IEC 61499 function blocks
Case example: pneumatic cylinder

- Two Boolean inputs (extend, retract)
- Two Boolean outputs (cylinder positions: start, end)
- Internal state: position (start, intermediate, end) and movement direction (left, right, stopped)
- A simple controller infinitely extends and retracts the cylinder
Proposed framework: overview

Test suite
- Test case 1
- Test case 2

Plant (to validate the test suite)

Controller under Test

Specification (additional conditions to be satisfied on the tests)

Input sensor values
- possible sensor values
- specification-specific values

Output sensor values
- output sensor values
- specification-specific values

Proposed framework: the test suite and the controller

Sequence of test elements

1. Set input and expected values
2. Wait until the controller produces desired values

If it does not, then there is a deadlock and the test is failed
An example for three test cases

- Vertices are states, edges are transition in NCES which execute simultaneously
- After initialization the state graph splits into several path graphs
- The final state of each path tells whether the test case is passed or failed
If the plant’s model is present, we have a closed-loop system.

But the state space is built only for the test cases, not for all possible behaviours of the plant.

This allows to analyze the system when the usual state space is too big.

In particular, the plant’s model allows to check the correctness of test inputs (i.e. that they can be generated by the plant).
Proposed framework: specification (optional)

- Additional conditions for the system to hold during test execution
- Example: the plant receives EXTEND and RETRACT simultaneously and reports an error
- This error is caught by the specification ('Observer' module)
Additional use case: time modeling

- TNCES is the extension of NCES which supports discrete time.
- If the system includes delays and especially timers, there is no need to wait during testing.
Additional use case: controllers with parameters and product lines

- If there is a family of controllers to test (i.e. the case of a product line), or a controller with parameters, the entire set of controllers can be tested in a single state space construction.

- This can be implemented as nondeterministic controller / module / parameter selection prior to test execution.
A framework for formal modeling of test execution, which can be used to:

- Analyze the system when its usual state space is too big
- Validate test suites
- Test software with delays and timers without the need to wait
- Test families of related controllers

The applicability of the framework was shown on the Cylinder example
Future work

- Tests which account for time, e.g. testing PLCs cycle by cycle
- Applying the framework on larger systems
- Generation of negative tests based on the plant’s model
- Coverage test generation based on the controller’s model
Thank you for your attention! Questions?

Formal Modeling of Testing Software for Cyber-Physical Automation Systems

Igor Buzhinsky, Cheng Pang, Valeriy Vyatkin

igor.buzhinsky@gmail.com