TEAMOD
Test and Model Checking

Bachelor and Master Project
Winter Semester 2018/19 – Summer Semester 2019
Jan Peleska and Wen-ling Huang
{jp,huang}@cs.uni-bremen.de
Background

- Embedded control systems are omnipresent in our daily life
Background

- The growing complexity of embedded applications leads to a dramatic increase of verification costs.
- These costs will further increase with autonomous systems deployment:
  - Robots
  - Vehicles
  - Cars
  - Ships
  - Trains
We can efficiently handle low complexity . . .

. . . but are not yet prepared for the complexity of future applications
Background

- Two approaches to cope with the complexity problem
  - Correctness by construction – synthesis of software and electronic circuits
  - Automated verification – model checking, theorem proving, and testing
TEAMOD Objectives

- Bounded model checking
- Model-based testing
I(s_0) \land \bigwedge_{i=1}^{k} R(s_{i-1}, s_i) \land G(s_0, \ldots, s_k)
Concrete test suite

1. \((v_{est}, v_{mrsp}) = (3.1, 100), (50, 100), (80, 100), (100.5, 100)\)
2. \((v_{est}, v_{mrsp}) = (90.1, 100), (110, 120), (110.5, 110), (110.5, 100)\) ...
3. \((v_{est}, v_{mrsp}) = (90.1, 200), (210, 220), (210.5, 210), (210.5, 200)\) ...
4. ...

Semantic representation

\[
\mathcal{R} = \bigvee_{i \in \text{IDX}} (\alpha_i \land (\tilde{m}, \tilde{y}) = (\tilde{d}_i, \tilde{e}_i) \land (\tilde{m}', \tilde{y}') = (\tilde{d}_i, \tilde{e}_i)) \\
\lor \bigvee_{(i,j) \in J} (g_{i,j} \land (\tilde{m}, \tilde{y}) = (\tilde{d}_i, \tilde{e}_i) \land (\tilde{m}', \tilde{y}') = (\tilde{d}_j, \tilde{e}_j) \land \tilde{x}' = \tilde{x})
\]

IDX = ...

\(J = \) ...

\((\tilde{d}_1, \tilde{e}_1) = \) ...

executed against SUT

Abstract test suite

\[
\mathcal{W} = P. (\bigcup_{i=0}^{m-n} \overline{I}_i . W)
\]
TEAMOD Objectives

- Bounded model checking and model-based testing have common foundations
  - Underlying theory
  - Algorithms
TEAMOD Objectives

• Bounded model checking and model-based testing should be integrated in a common tool platform

• Model checking is needed to verify the test model

• Model-in-the-loop testing is a light-weight version of model checking
TEAMOD Bachelor Project

• Three sub-projects
  • Methods
  • System under test development
  • Modelling, bounded model checking, and model-based testing
Hardware-in-the-loop test

Test engine

System under test
TEAMOD Master Project

- Four sub-projects
  - Algorithms
  - Autonomous train control
  - Safety monitor development
  - Graphical interfaces for scenario-based testing
2. Algorithm design

8. For each \((i, j) \in J\), collect all disjuncts

\[
g_{i', i_1' \ldots i_n'}(m, y) = (d_{i'}, e_i) \land (m', y') = (d_{i'\ldots i_n'}, e_{i'}) \land (x' = x)
\]

satisfying \(i', i_1' \ldots i_n' \in \text{RTR}_{i, j}\) and consequently \(i' = i, i_n' = j\) and merge them into a single disjunct

\[
g_{i, j} \land (m, y) = (d_i, e_i) \land (m', y') = (d_j, e_j) \land (x' = x)
\]

where

\[
g_{i, j} \equiv \bigvee_{i', i_1' \ldots i_n' \in \text{RTR}_{i, j}} g_{i', i_1' \ldots i_n'}
\]

9. Terminate by returning \(\mathcal{R}\).

3. Programming

```cpp
void RttTgenGenerator::generateTestCases()
{
    // The root of the test procedure tree carries the memory state before
    tprocRoot = [&] {
        auto mSys = static_cast<RttTgenConcreteLatticeMemory*>(system->get
            mSys->setParentSystem(system);
        return new RttTgenTestProcTree(mSys, 0, true, 0);
    }();
    currentProcNode = tprocRoot;

    remainingSimulationSteps = parms->getSimSteps();

    // Initialise interpreters
    sim = new simlib::Simulator(*system);
    sim->setAddGoalsUnordered(additionalGoals->getUnordered());
    sim->setAddGoalsOrdered(additionalGoals->getOrdered());
    sim->setTestCaseDb(tcDb);
    sim->setParms(parms);
    nextTickFromSimulator = 0;
}
```
Autonomous train control & Safety monitor

Safety monitor

Märklin Modelleisenbahn

Märklin Central Station 6021

Generator

ΦSafe
Graphical interfaces for scenario-based testing
Accompanying Lectures

- **Test automation** [highly recommended]
- Theory of reactive systems
- Systems of high quality, safety, and security
- Specification of embedded systems
- Operating systems
- Real-time operating systems development