Data-Driven Test Generation for Black-Box Systems From Learned Decision Tree Models

Swantje Plambeck, Görschwin Fey
Agenda:

1. Motivation
2. Model-Based Testing
3. Testing with Decision Trees
4. Coverage Metric
5. Automatic Test Generation
6. Example
7. Conclusion
Motivation

• Testing is an important step in design, operation, and maintenance of systems

• For complex, black-box systems deriving test cases is particularly difficult

• A solution is model-based testing (MBT) with learned models

→ We propose a new MBT approach using decision tree models

→ Decision trees allow to learn from bounded history
Model-Based Testing

- MBT consists of a model learning and a test generation step. Often, finite automaton models are considered and state, transition, or other coverage criteria are used for test generation [1,2,3]

24.03.2023
Testing with Decision Trees

- Decision Tree Models represent the sequential behaviour of a system
- Feature vectors show $N$ previous time steps and predict the output of the next time step

In the following, we assume systems with a Mealy machine representation

### Feature Vector

- $i_t$: input symbol
- $o_t$: output symbol
- $t$: discrete time step

### Label | Input | Output
---|---|---
A | x | a
B | y | a
C | x | b
D | y | b
E | x | c
F | y | c
Testing with Decision Trees

- The decision tree is learned from observations of bounded history
  - Enables model learning without knowledge of an initial state or possibility to return to the initial state
    → testing without reset to an initial state

- We call this **Ad-hoc Testing**

- Knowing the current history, we want to find future inputs to cover a maximum amount of system behaviour
  → How to define coverage on a decision tree model?
Coverage Metric

- Most relevant system behaviour is encoded in the paths from root to leaf nodes
- A discrete time step corresponds to an update of the current history...

\[
i_k \quad o_k \quad i_{k+1} \quad o_{k+1} \quad i_{k+2} \quad o_{k+2} \quad \ldots \quad i_{k+N} \quad o_{k+N}
\]
Coverage Metric

- Most relevant system behaviour is encoded in the paths from root to leaf nodes

- A discrete time step corresponds to an update of the current history...

\[ i_{k+1} o_{k+1} i_{k+3} o_{k+3} \ldots i_{k+N} o_{k+N} i_{\text{new}} o_{\text{new}} \]

- and, thus to a transition between two leaves.

→ The coverage metric is \textit{leaf coverage} – this is a state coverage in the decision tree’s state machine
Automatic Test Generation

- Idea:

  Decision Tree Model → Automaton Representation of Leaf Transitions → Find a path that visits all states → Test Case

Test Case:
- ✓
- ✓
- ✓
- ✓
Automatic Test Generation

- Idea:
  - The Hamilton path problem is NP-hard $\rightarrow$ high computational complexity
Automatic Test Generation

• Idea:

  Decision Tree Model ➔ Automaton Representation of Leaf Transitions ➔ Find a (shortest) Hamilton Path ➔ Test Case

  • The Hamilton path problem is NP-hard ➔ high computational complexity

• Greedy Approach:

  Decision Tree Model ➔ Automaton Representation of Leaf Transitions ➔ Iteratively visit (closest) unreached states ➔ Test Case

  • Might end-up in dead-end states early
Automatic Test Generation

• Idea:

  - The Hamilton path problem is NP-hard → high computational complexity

• Greedy Approach:

  - Iteratively visit (closest) states with most reachable unreached states
Assumption: current history is \([\text{Clean/ok, Pod/ok, Water}]\)

\[ \rightarrow \text{we are in state 2 and go to state 4 of the automaton} \]
Assumption: current history is [Clean/ok, Pod/ok, Water]
⇒ we are in state 2 and go to state 4 of the automaton
⇒ The next output is ok

We choose a next input Button
⇒ The next history is [Pod/ok, Water/ok, Button]
Assumption: current history is \([\text{Pod/ok, Water/ok, Button}]\)

⇒ we are in state 4 and go to state 5 of the automaton

⇒ The next output is coffee

We choose a next input Clean

⇒ The next history is \([\text{Water/ok, Button/coffee, Clean}]\)
Assumption: current history is \([\text{Water / ok, Button / coffee, Clean}]\)

\(\rightarrow\) we are in state 5 and go to state 1 of the automaton

\(\rightarrow\) The next output is \text{ok}

We choose a next input \text{Water, Button}

\(\rightarrow\) The next history is \([\text{Clean / ok, Water / ok, Button}]\)


24.03.2023
Assumption: current history is \([\text{Clean/ok, Water/ok, Button}]\)

\[\rightarrow\] we are in state 3 and go to state 6 of the automaton

\[\rightarrow\] The next output is error

We choose a next input Pod

\[\rightarrow\] The next history is \([\text{Water/ok, Button/error, Pod}]\)
A leaf coverage of $\frac{5}{7}$ is reached while a full state coverage on the original automaton representation is achieved.
• We introduced a new strategy for Model-Based Testing (MBT) using decision tree models

• The advantage is the learnability and, thus testability from bounded history (ad-hoc testing)

• We proposed multiple strategies to apply automatic test generation

• Future work considers

  • Comparison to existing MBT approaches

  • Evaluation of complexity and scalability
References


Footnotes:


Thank You

Technische Universität Hamburg (TUHH)
Swantje Plambeck
Institut für Eingebettete Systeme
Am Schwarzenberg-Campus 3
21073 Hamburg
E-Mail: swantje.plambeck@tuhh.de
Tel.: +49 40 42878-3867
www.tuhh.de