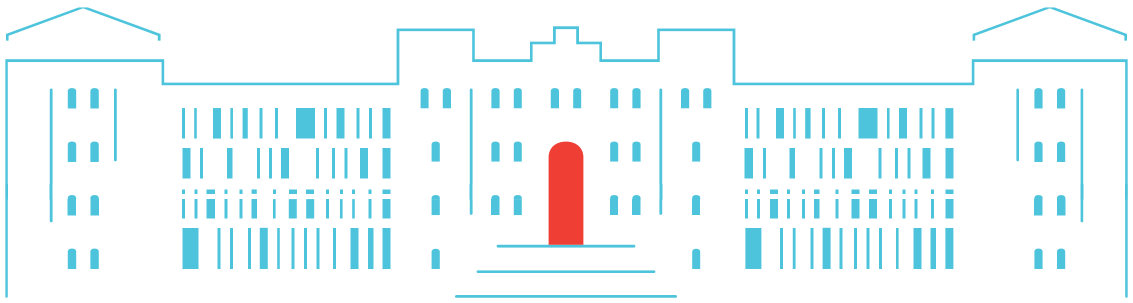
Data-Driven Test Generation for Black-Box Systems From Learned Decision Tree Models

TUHH Technische Universität Hamburg

24.03.2023



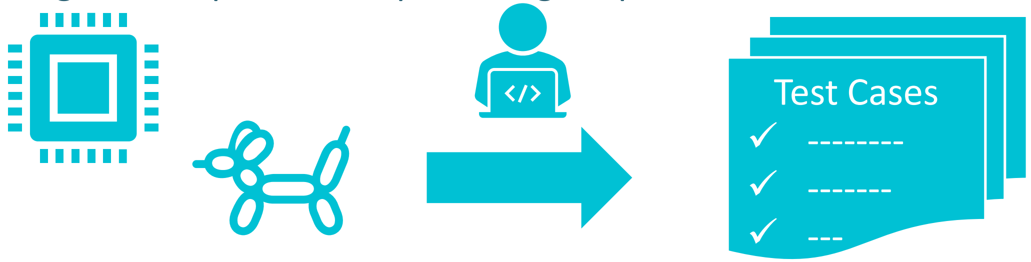
Swantje Plambeck, Görschwin Fey

Agenda:

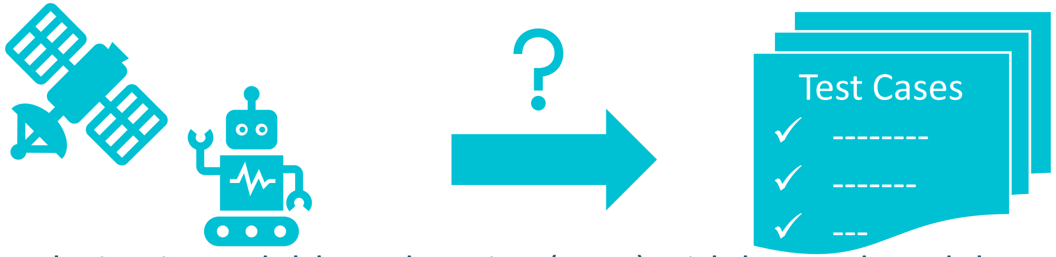
1. Motivation
2. Model-Based Testing
3. Testing with Decision Trees
4. Coverage Metric
5. Automatic Test Generation
6. Example
7. Conclusion

Motivation

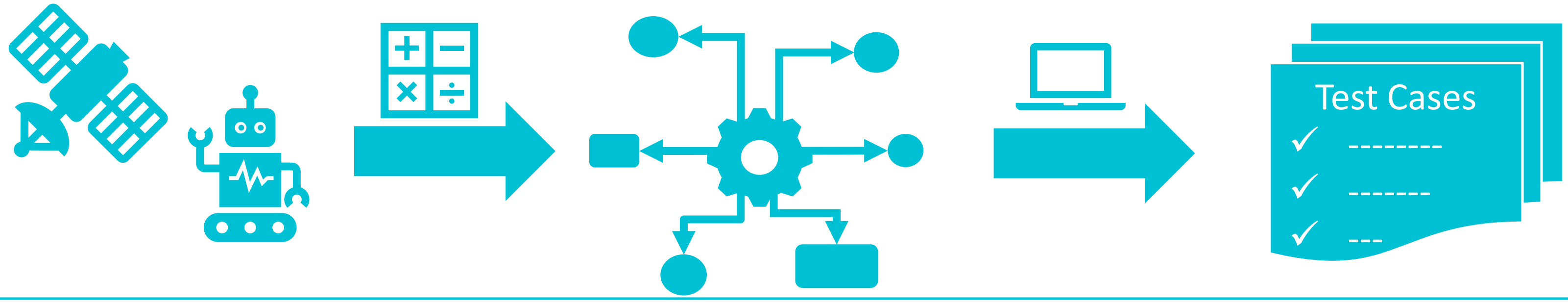
- Testing is an important step in design, operation, and maintenance of systems



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



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

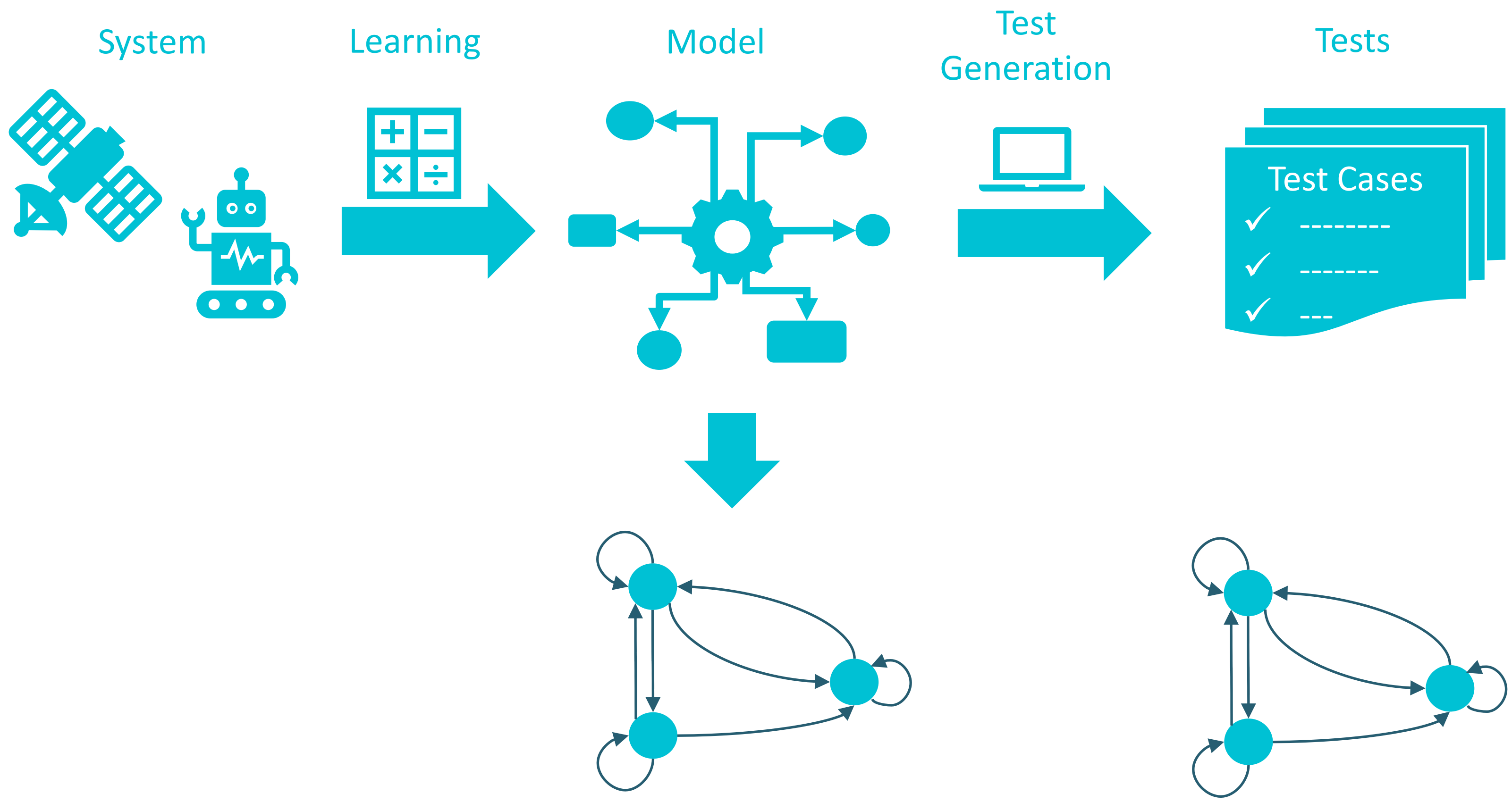


→ We propose a new MBT approach using decision tree models

→ Decision trees allow to learn from bounded history

Model-Based Testing

TUHH

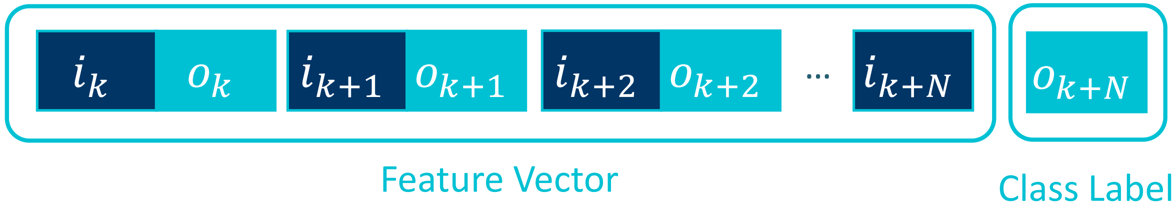


- MBT consists of a model learning and a test generation step. Often, finite automaton models are considered and state, transition, or other coverage criteria are used for test generation [1,2,3]

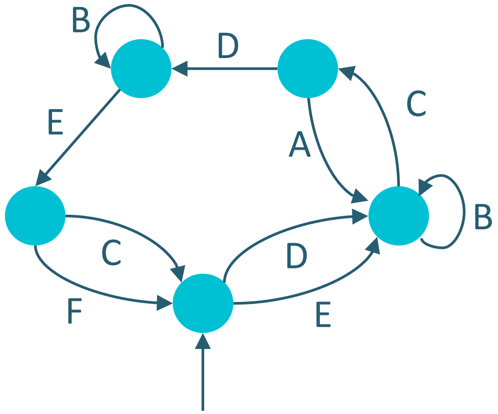
Testing with Decision Trees

i_t: *input symbol*
o_t: *output symbol*
t: *discrete time step*

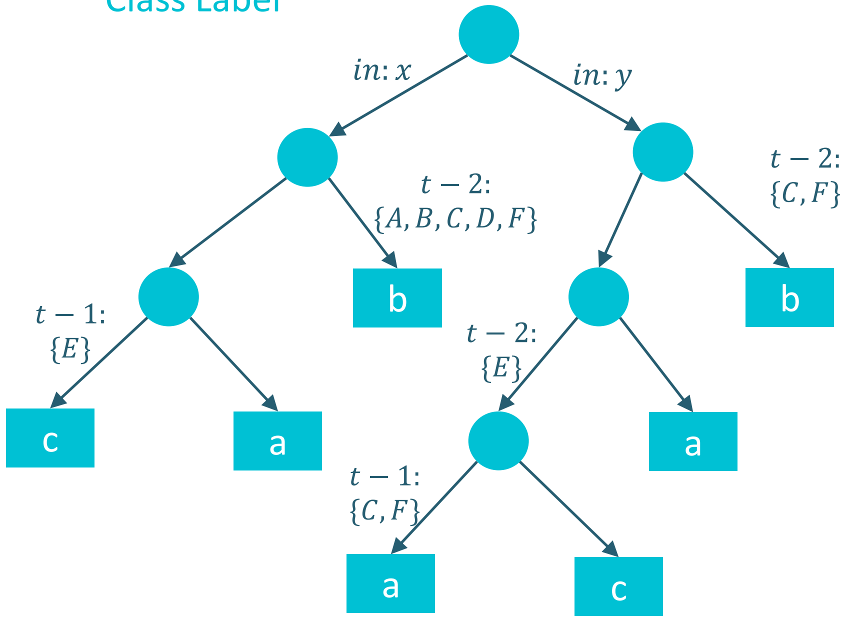
- Decision Tree Models represent the sequential behaviour of a system
- Feature vectors show N previous time steps and predict the output of the next time step



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



Label	Input	Output
A	x	a
B	y	a
C	x	b
D	y	b
E	x	c
F	y	c



S. Plambeck, L. Schammer, and G. Fey, “On the viability of decision trees for learning models of systems,” in Asia and South Pacific Design Automatio n Conference (ASP-DAC), 2022, pp. 696–701.

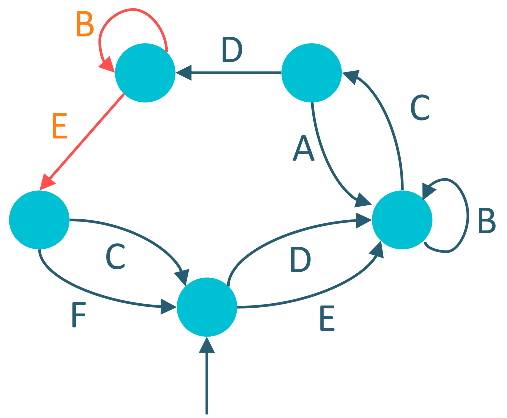
Testing with Decision Trees

TUHH

- 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**



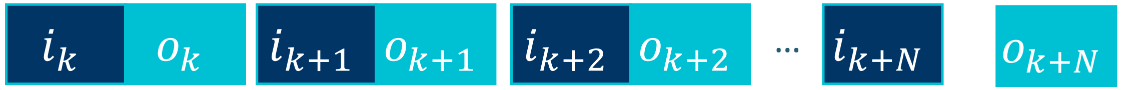
- Knowing the current history, we want to find future inputs to cover a maximum amount of system behaviour

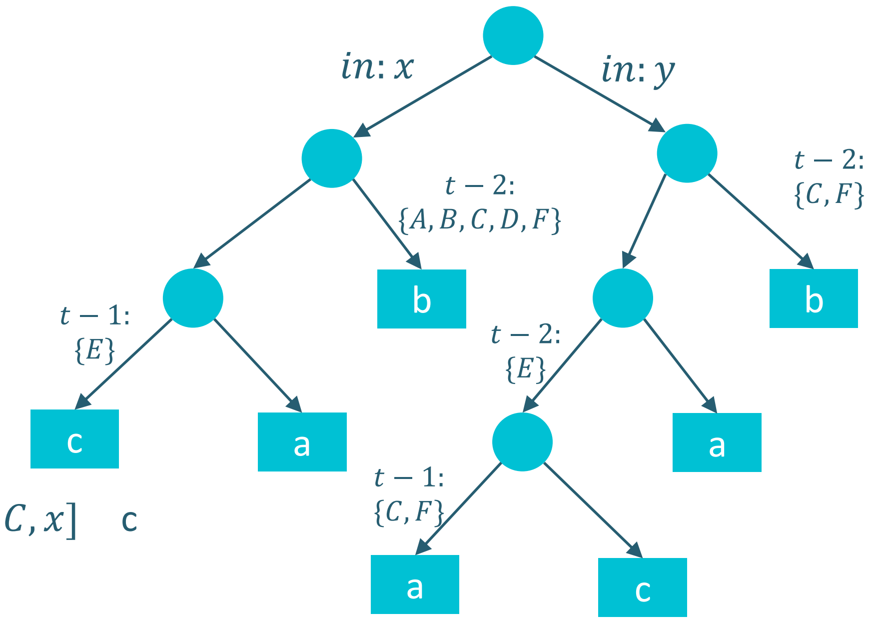
→ How to define coverage on a decision tree model?

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…





$[E, C, x]$ c

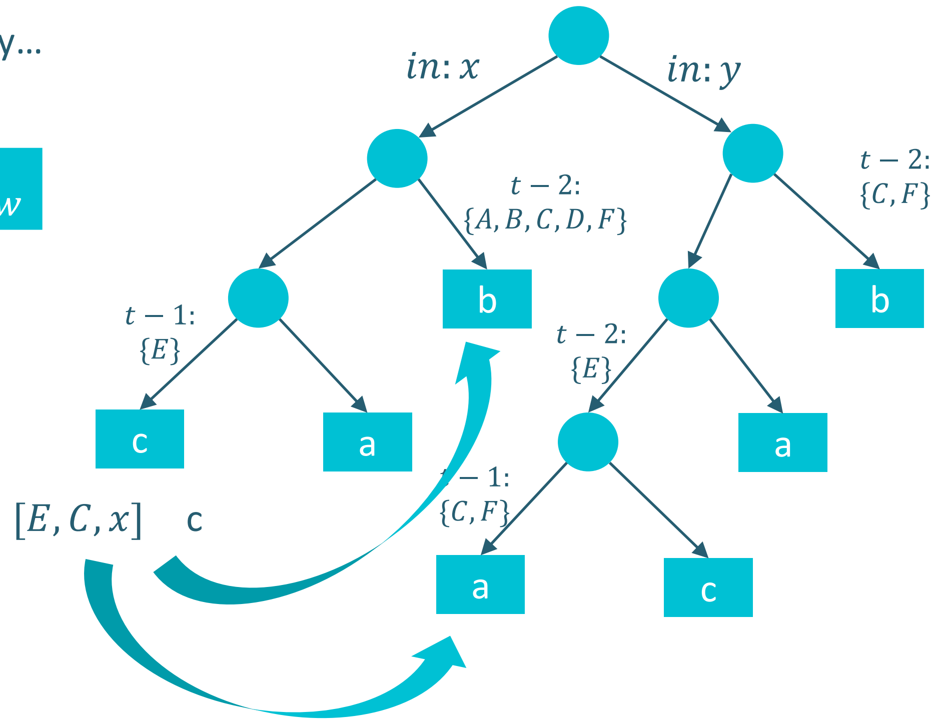
Coverage Metric

TUHH

- 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} … i_{k+N} o_{k+N} i_{new} o_{new}

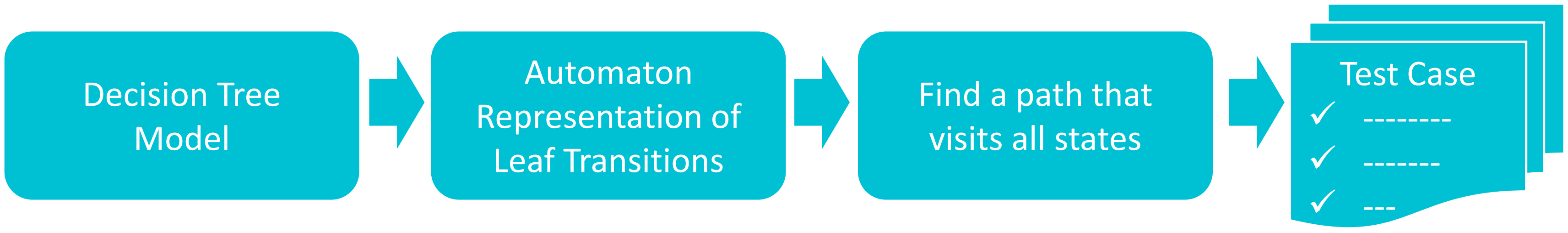
- and, thus to a transition between two leaves.



→ The coverage metric is **leaf coverage** – this is a state coverage in the decision tree‘s state machine

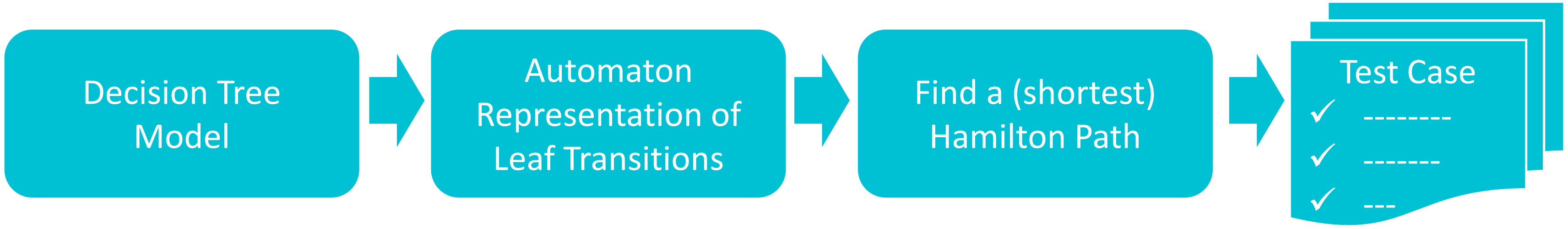
Automatic Test Generation

- Idea:



Automatic Test Generation

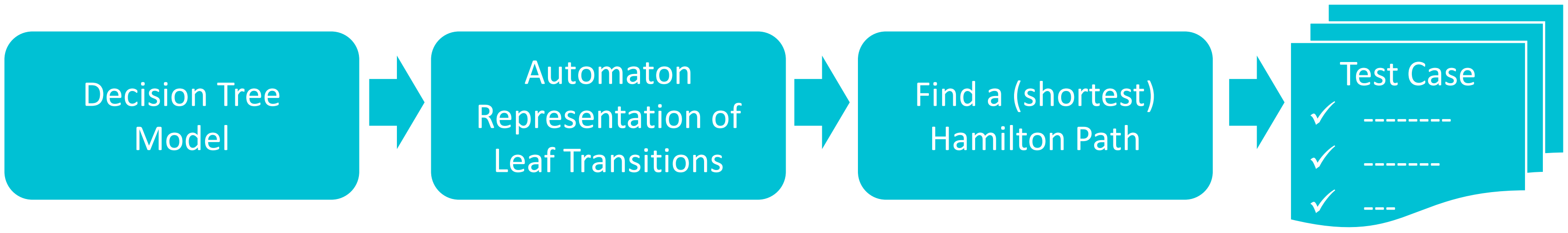
- Idea:



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

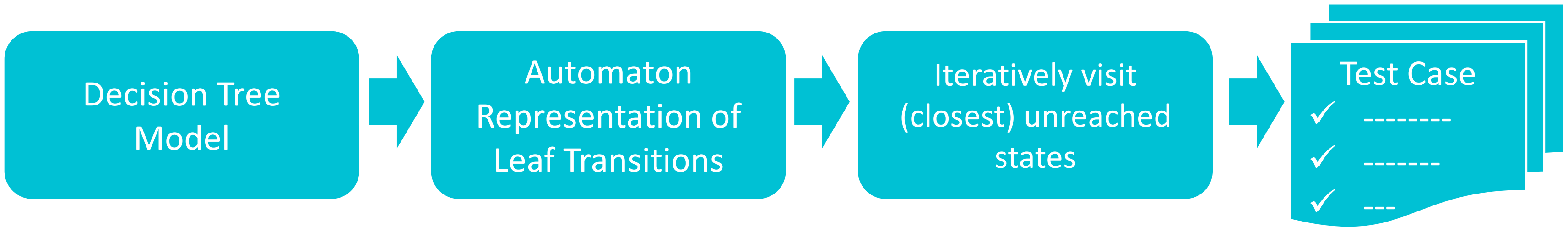
Automatic Test Generation

- Idea:



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

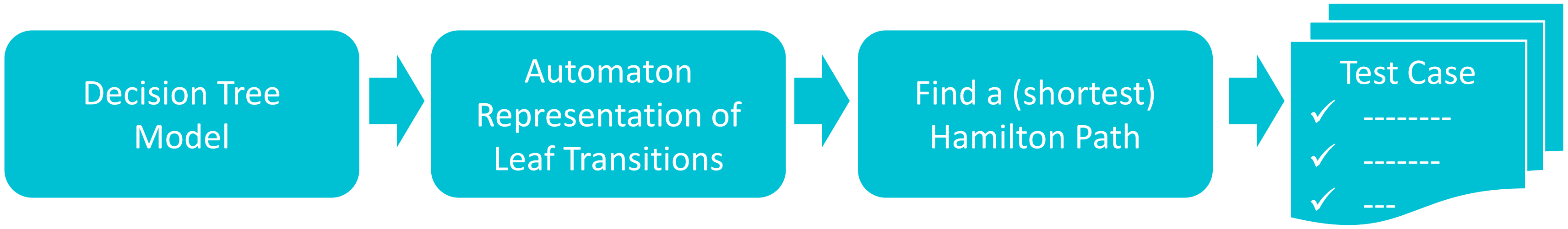
- Greedy Approach:



  - Might end-up in dead-end states early

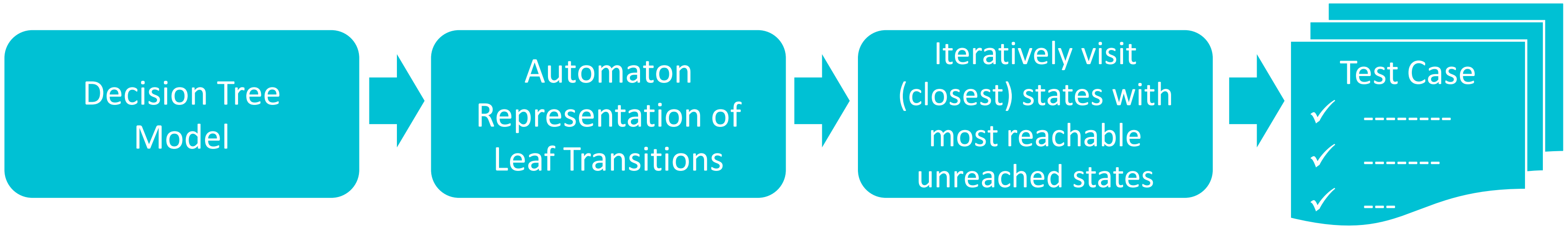
Automatic Test Generation

- Idea:

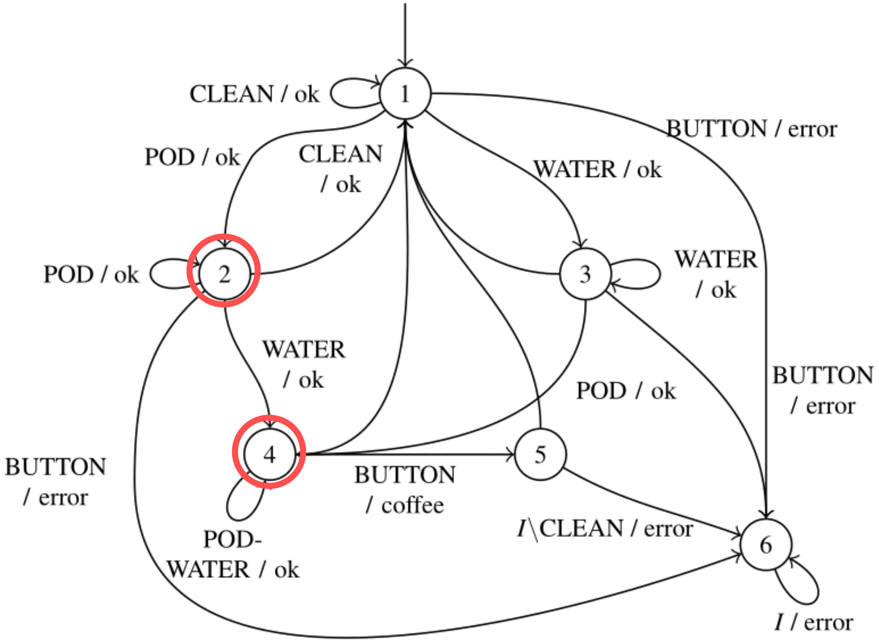


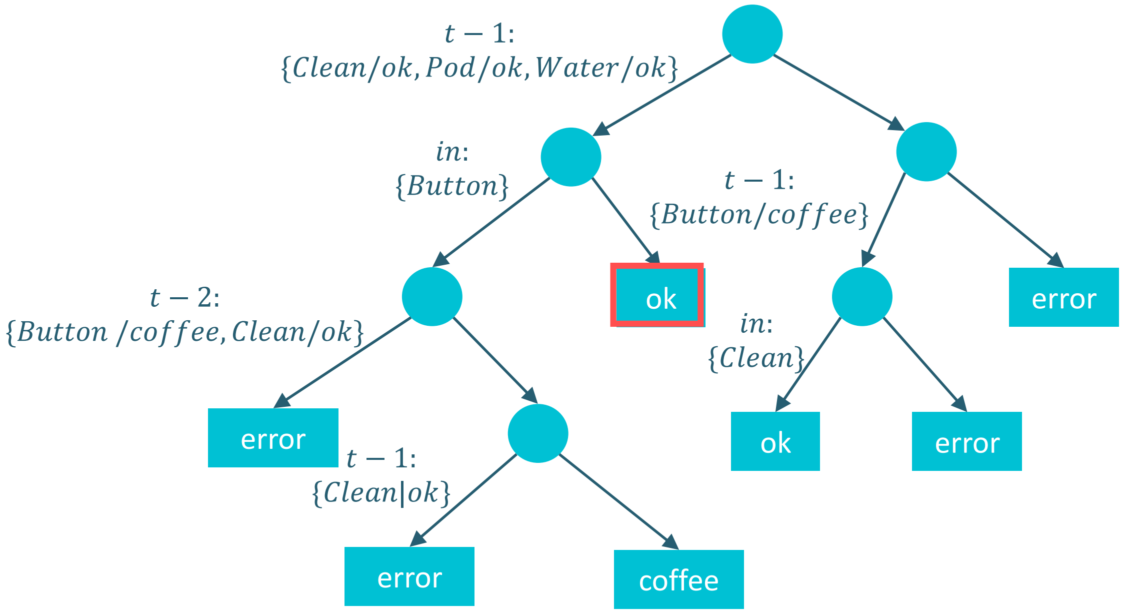
  - The Hamilton path problem is NP-hard → high computational complexity

  - Greedy Approach:



Example

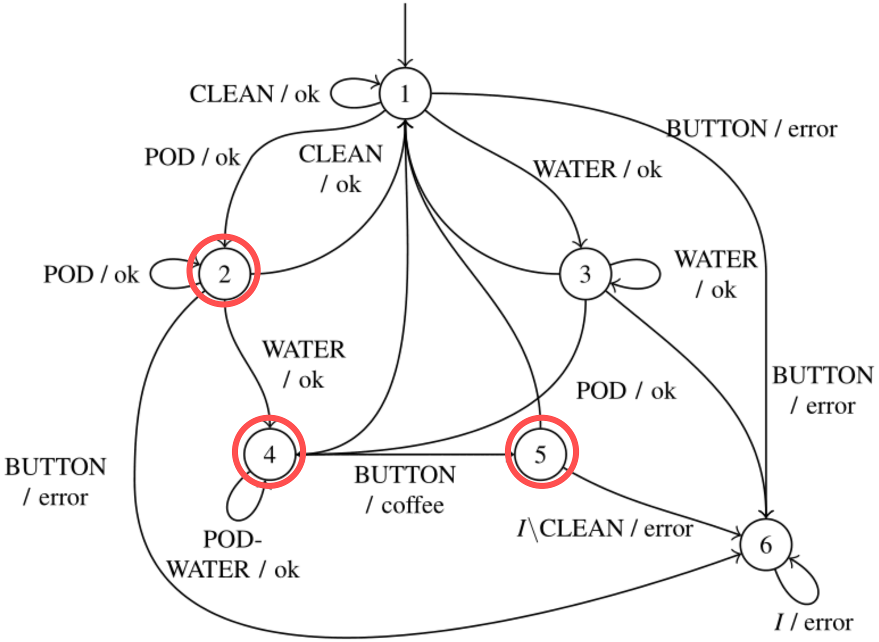


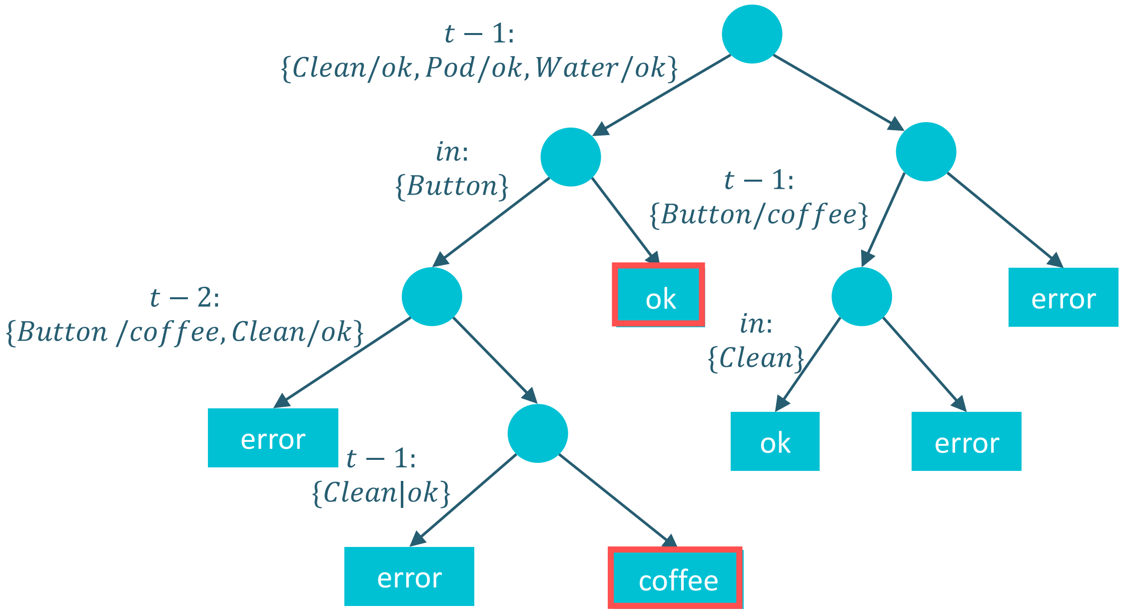


Assumption: current history is $[Clean/ok, Pod/ok, Water]$
→ we are in state 2 and go to state 4 of the automaton

Coffee Machine Example from: B. Steffen, F. Howar, and M. Merten, “Introduction to active automata learning from a practical perspective,” in Formal Methods for Eternal Networked Software Systems, 2011, pp. 256–296.

Example



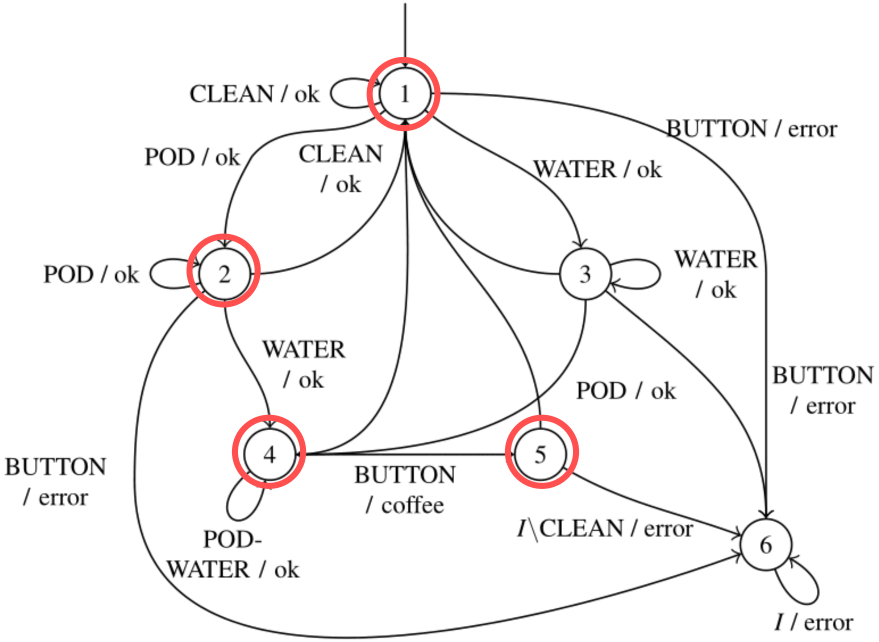


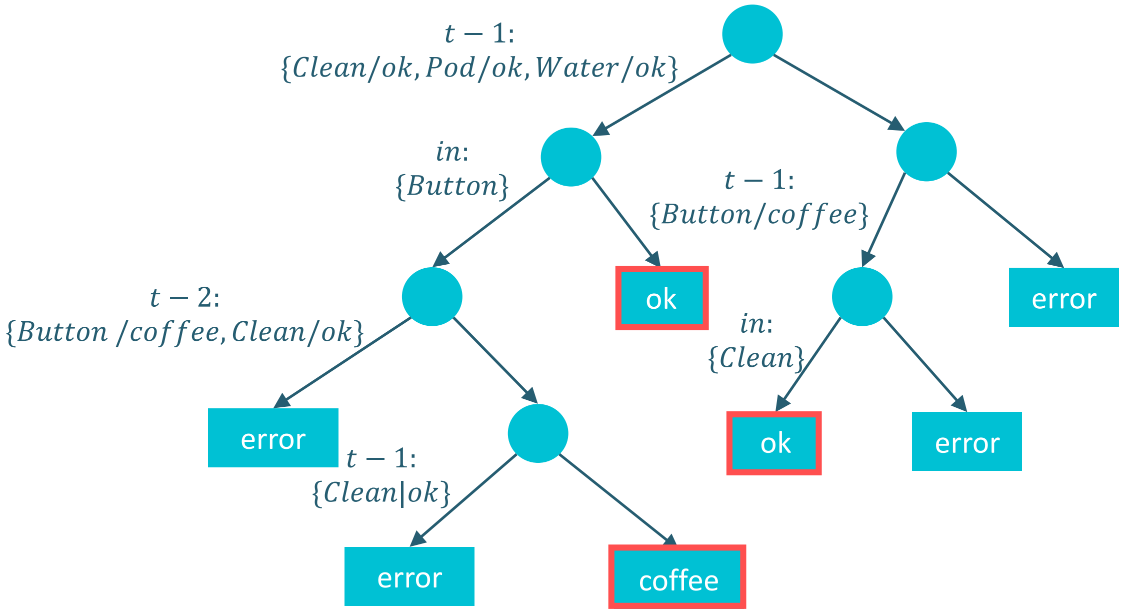
Assumption: current history is $[Clean/ok, Pod/ok, Water]$
→ we are in state 2 and go to state 4 of the automaton
→ The next output is ok

We choose a next input $Button$
→ The next history is $[Pod/ok, Water/ok, Button]$

Coffee Machine Example from: B. Steffen, F. Howar, and M. Merten, “Introduction to active automata learning from a practical perspective,” in Formal Methods for Eternal Networked Software Systems, 2011, pp. 256–296.

Example





Assumption: current history is $[Pod/ok, Water/ok, Button]$

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

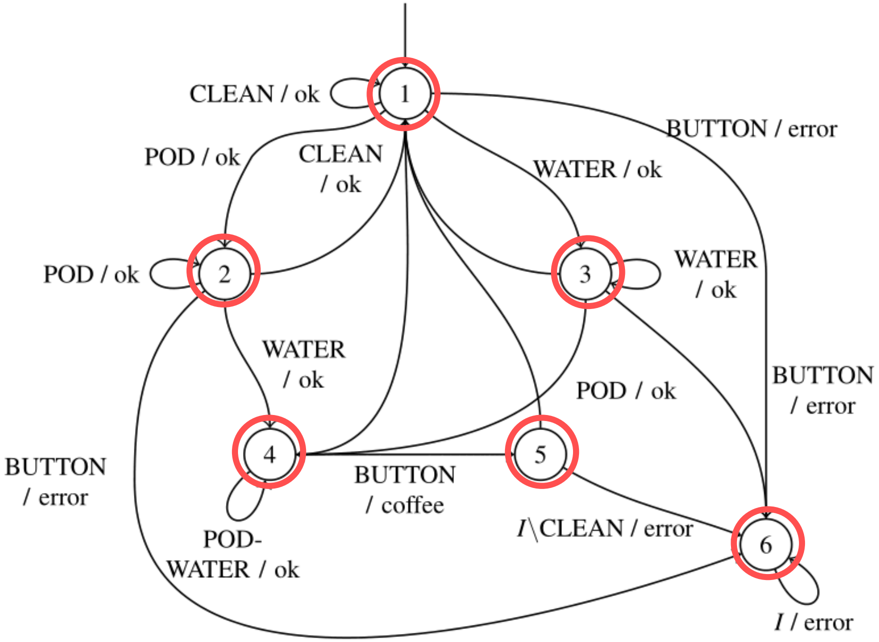
→ The next output is $coffee$

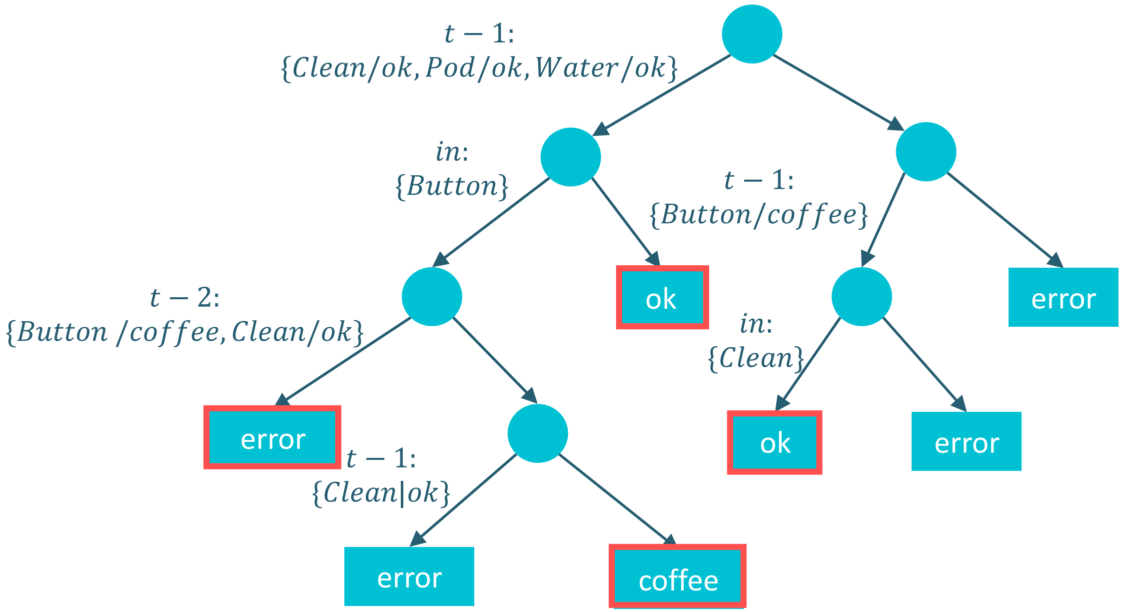
We choose a next input $Clean$

→ The next history is $[Water/ok, Button/coffee, Clean]$

Coffee Machine Example from: B. Steffen, F. Howar, and M. Merten, “Introduction to active automata learning from a practical perspective,” in Formal Methods for Eternal Networked Software Systems, 2011, pp. 256–296.

Example



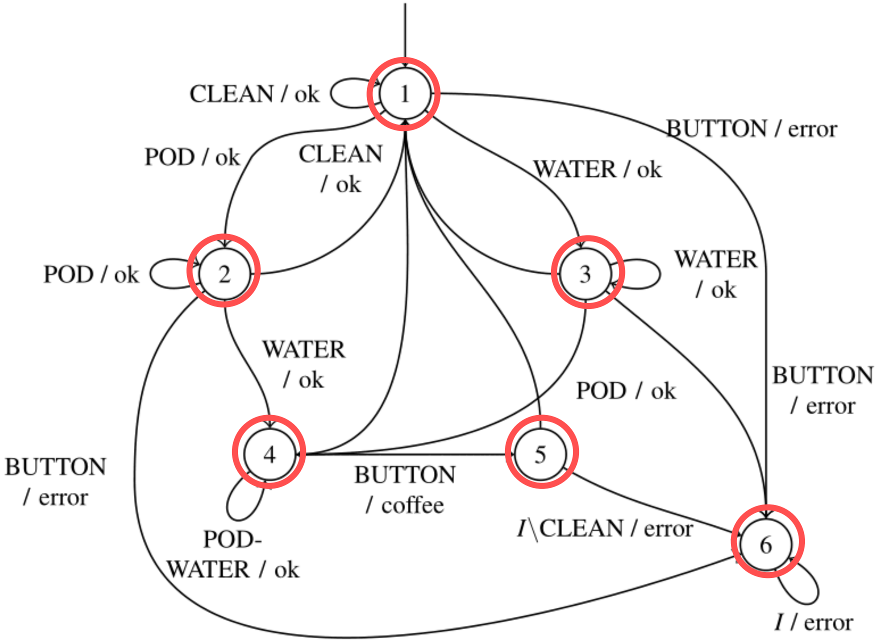


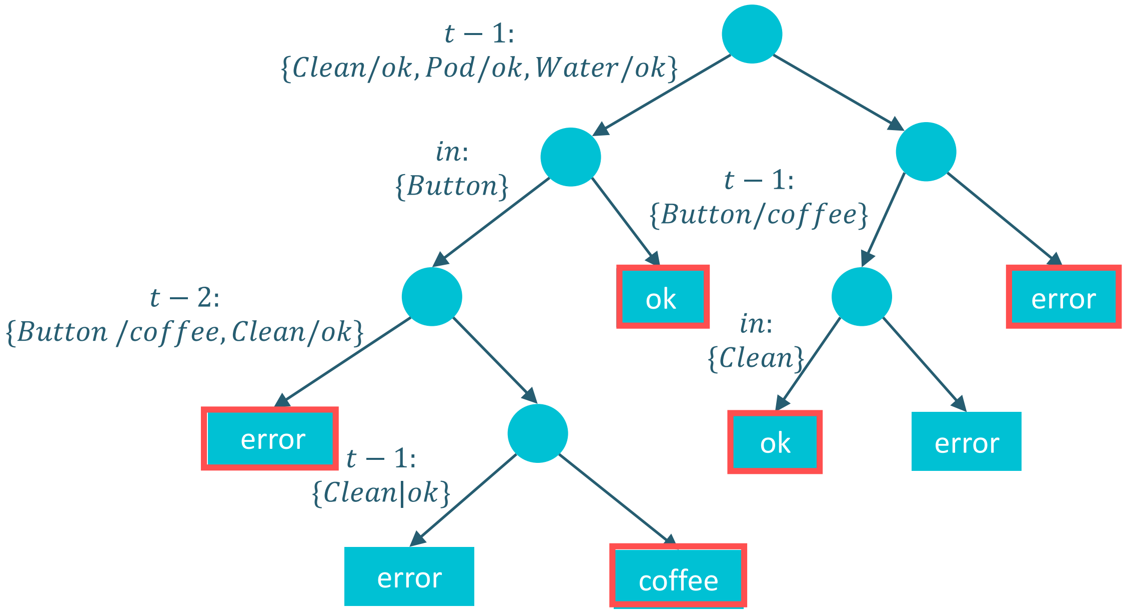
Assumption: current history is $[Water/ok, Button/coffee, Clean]$
→ we are in state 5 and go to state 1 of the automaton
→ The next output is ok

We choose a next input $Water, Button$
→ The next history is $[Clean/ok, Water/ok, Button]$

Coffee Machine Example from: B. Steffen, F. Howar, and M. Merten, "Introduction to active automata learning from a practical perspective," in Formal Methods for Eternal Networked Software Systems, 2011, pp. 256–296.

Example



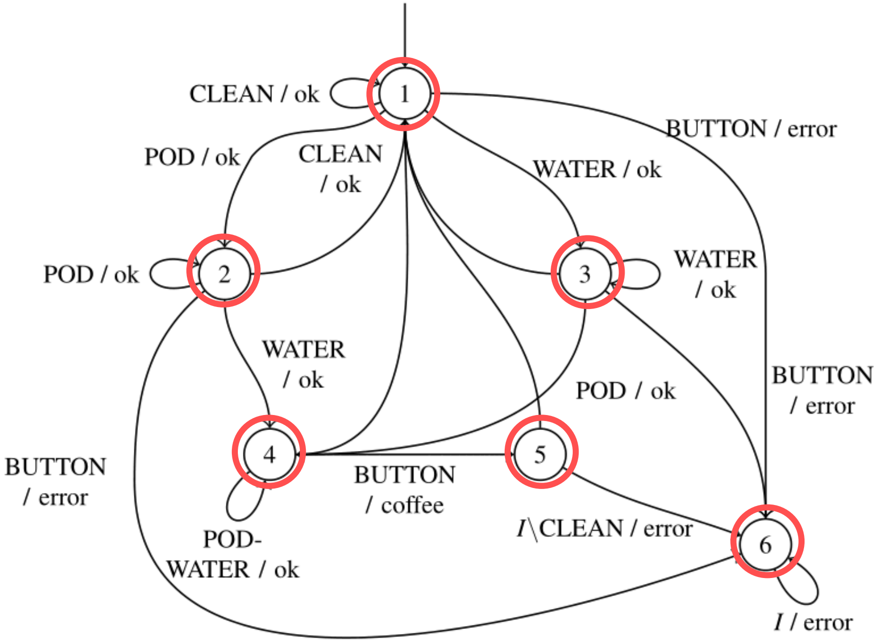


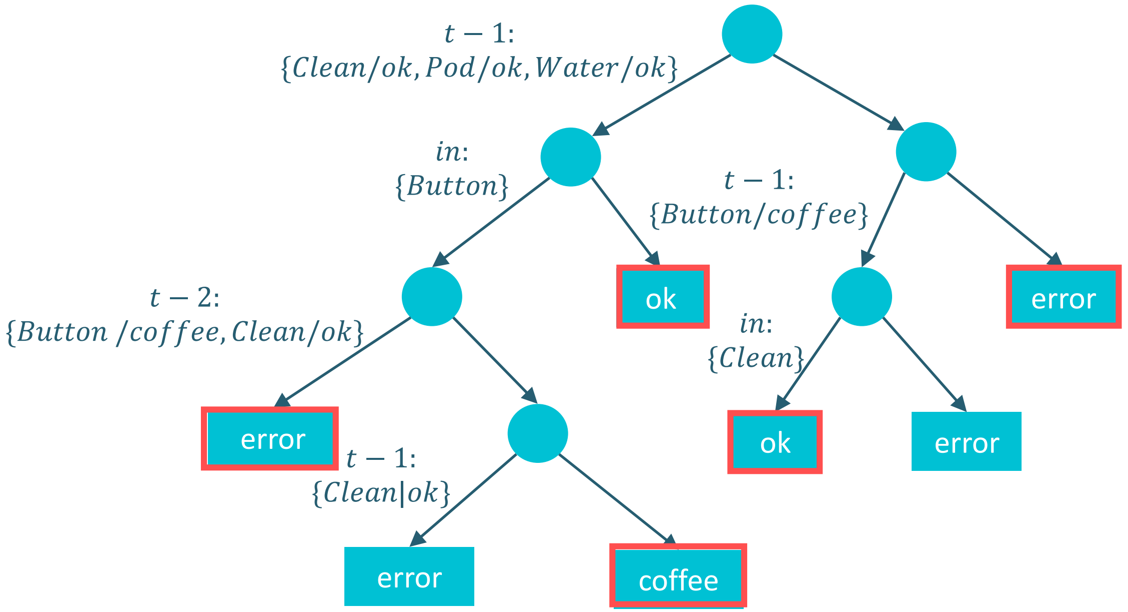
Assumption: current history is $[Clean/ok, Water/ok, Button]$
→ we are in state 3 and go to state 6 of the automaton
→ The next output is $error$

We choose a next input Pod
→ The next history is $[Water/ok, Button/error, Pod]$

Coffee Machine Example from: B. Steffen, F. Howar, and M. Merten, "Introduction to active automata learning from a practical perspective," in Formal Methods for Eternal Networked Software Systems, 2011, pp. 256–296.

Example





→ A leaf coverage of $\frac{5}{7}$ is reached while a full state coverage on the original automaton representation is achieved

Coffee Machine Example from: B. Steffen, F. Howar, and M. Merten, "Introduction to active automata learning from a practical perspective," in Formal Methods for Eternal Networked Software Systems, 2011, pp. 256–296.

Conclusion

TUHH

- 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

[1] K. I. Eder, W. ling Huang, and J. Peleska, "Complete agent-driven model-based system testing for autonomous systems," in Workshop on Formal Methods for Autonomous Systems (FMAS), 2021.

[2] K. A. El-Fakih, R. Dorofeeva, N. V. Yevtushenko, and G. V. Bochmann, "Fsm-based testing from user defined faults adapted to incremental and mutation testing," Programming and Computer Software, vol. 38, no. 4, 2012.

[3] J. Peleska, E. Vorobev, and F. Lapschies, "Automated test case generation with smt-solving and abstract interpretation," in NASA Formal Methods. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 298–312.

Footnotes:

[4] S. Plambeck, L. Schammer, and G. Fey, "On the viability of decision trees for learning models of systems," in Asia and South Pacific Design Automation Conference (ASP-DAC), 2022, pp. 696–701.

[5} B. Steffen, F. Howar, and M. Merten, "Introduction to active automata learning from a practical perspective," in Formal Methods for Eternal Networked Software Systems, 2011, pp. 256–296.

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

tuhh.de

TUHH
Technische Universität Hamburg