Coverage-Based Testing with Symbolic Transition Systems

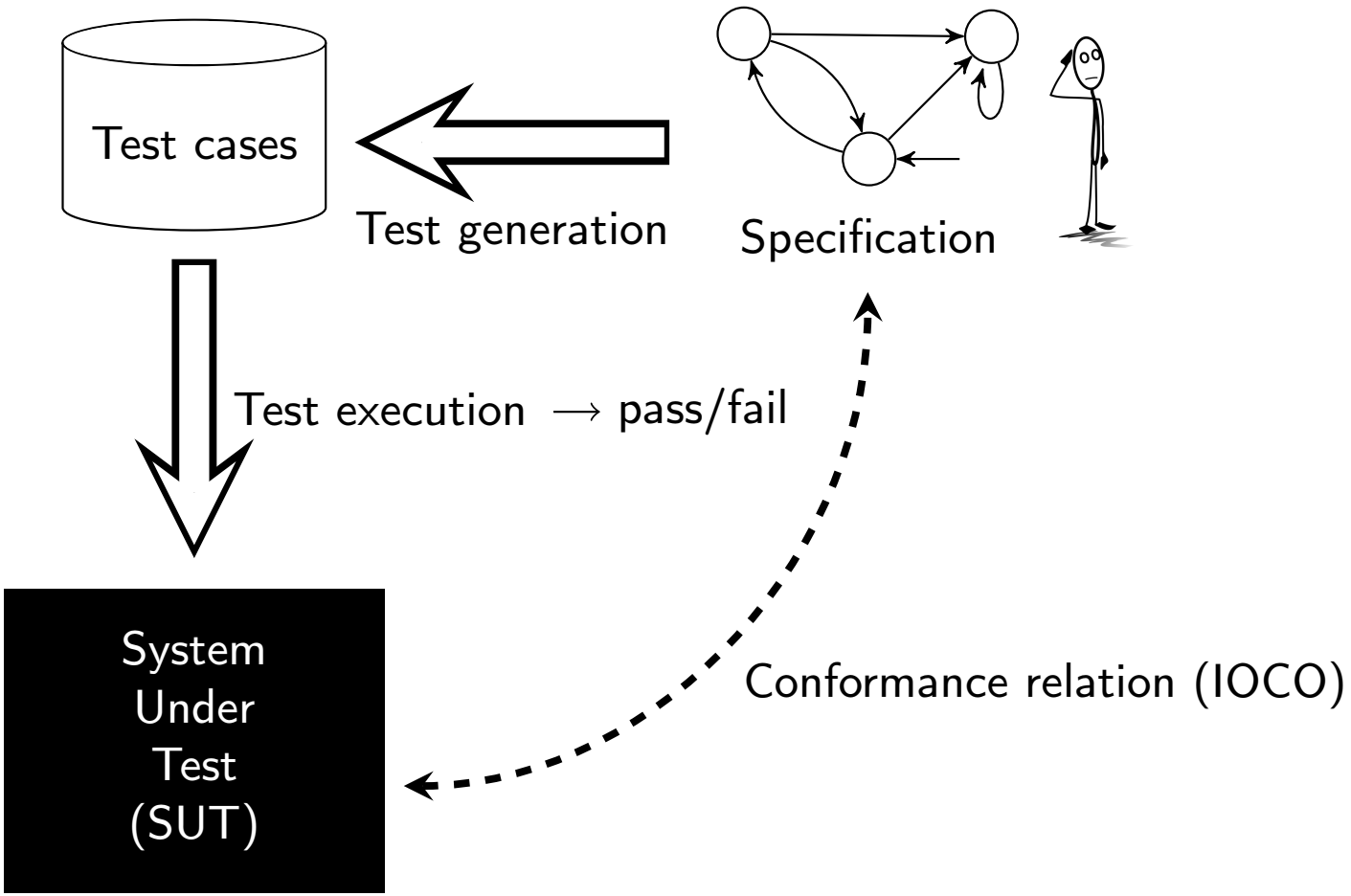
Petra van den Bos[1]
Jan Tretmans[1,2]

[1] Radboud University, Nijmegen, the Netherlands
[2] ESI (TNO), Eindhoven, the Netherlands

STW
Enabling new technology

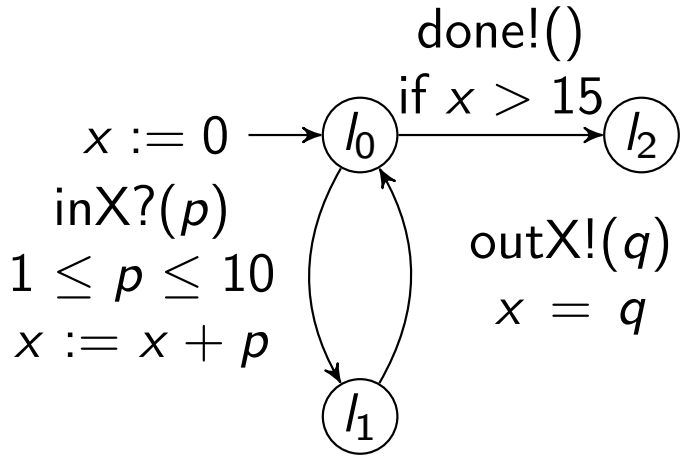
iCIS | Software Science
Radboud University

Model-Based Testing



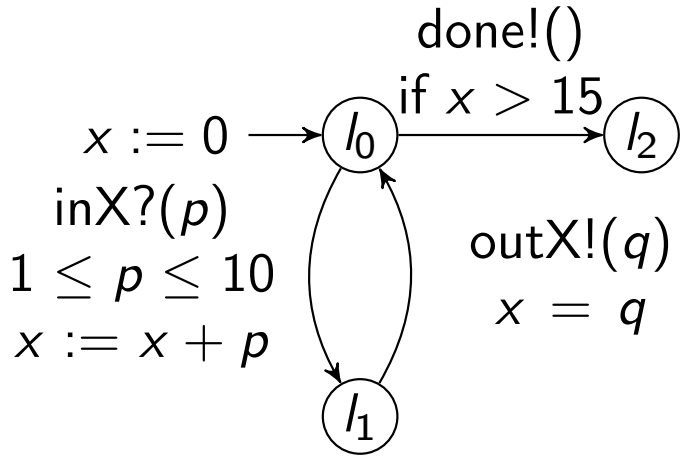
Testing Systems with Control Flow and Data

- Present-day systems use both:
  - control flow: states, actions
  - data: variables, parameters, conditions ...



Testing Systems with Control Flow and Data

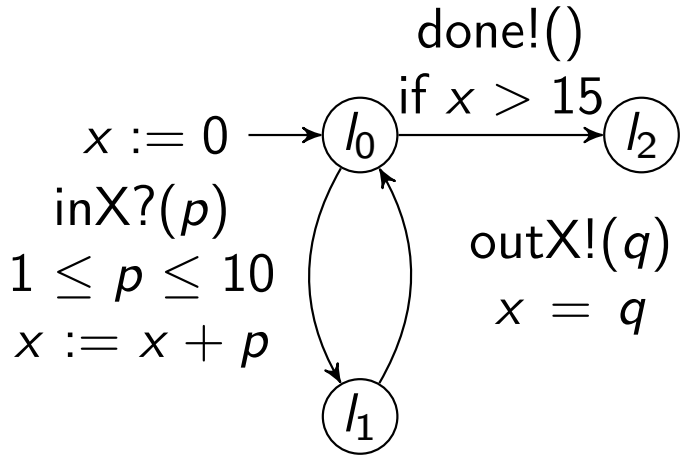
- Present-day systems use both:
  - control flow: states, actions
  - data: variables, parameters, conditions ...



- How to deal with state space explosion
- ... and bound test execution time?

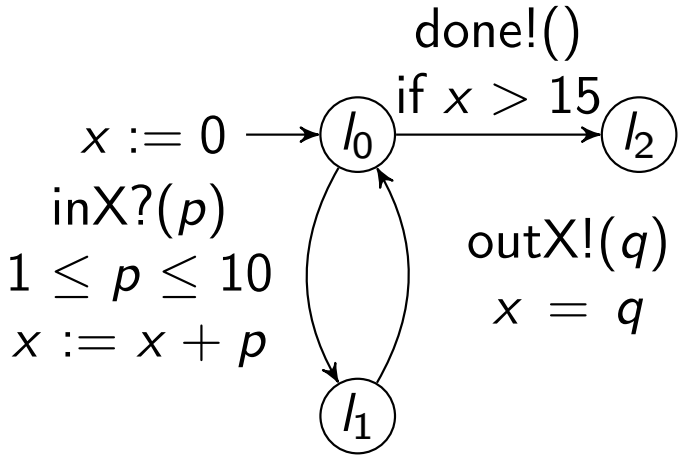
Testing Systems with Control Flow and Data

- Present-day systems use both:
  - control flow: states, actions
  - data: variables, parameters, conditions ...

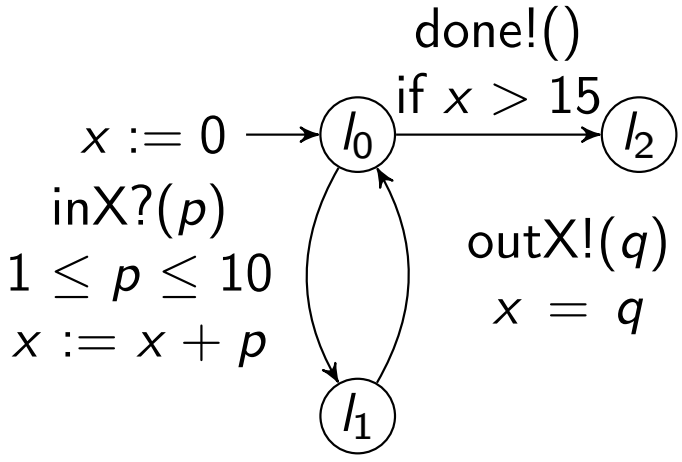


- How to deal with state space explosion
- ... and bound test execution time?
- Use switch coverage for test selection!

Model: Symbolic Transition System

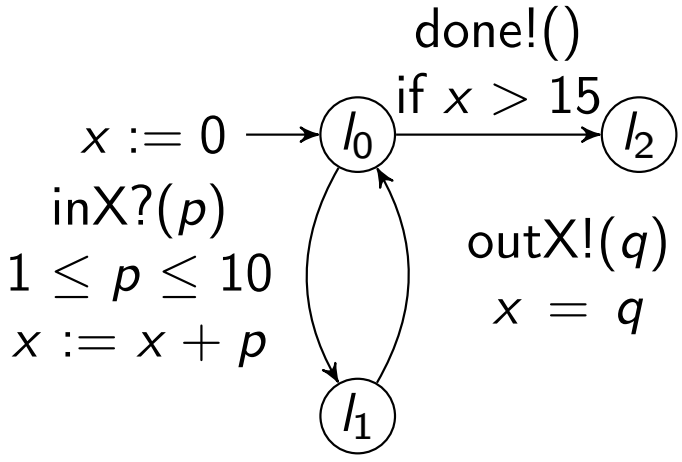


Switch coverage in STS: Reachability & Data



How to reach the done!() switch?

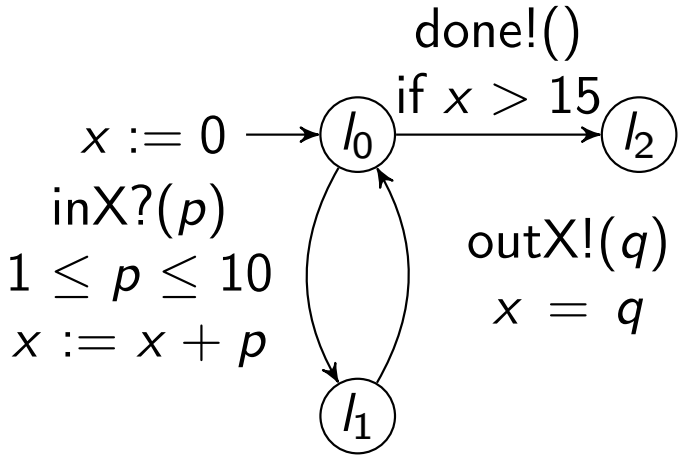
Switch coverage in STS: Reachability & Data



How to reach the done!() switch?

- inX?(6)
- outX!(6)

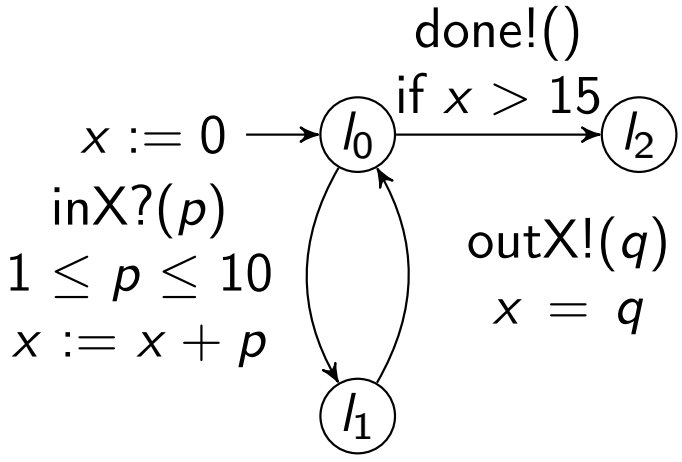
Switch coverage in STS: Reachability & Data



How to reach the done!() switch?

- inX?(6)
- outX!(6)
- inX?(10)
- outX!(16)

Switch coverage in STS: Reachability & Data



How to reach the done!() switch?

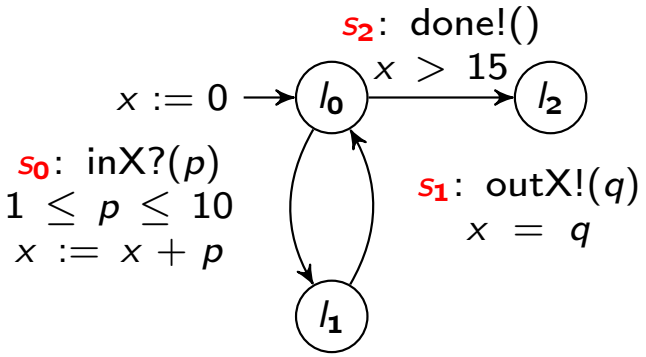
- inX?(6)
- outX!(6)
- inX?(10)
- outX!(16)
- done!()

Contributions

- Switch coverage test selection
- On-the-fly parameter value generation
- Deal with SMT solvers returning Unknown

Switch coverage test generation

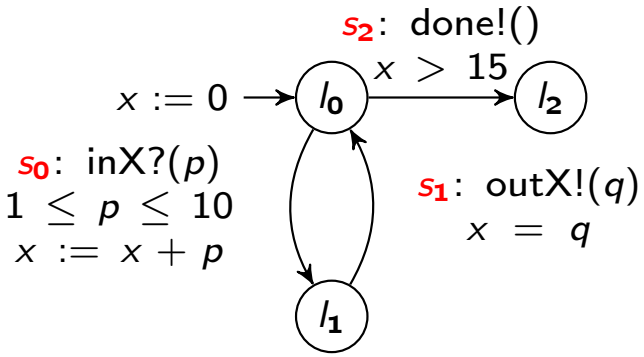
- Obtain test purposes using Symbolic Execution



$\longrightarrow (l_0, \{x \mapsto 0\}, True)$

Switch coverage test generation

- Obtain test purposes using Symbolic Execution



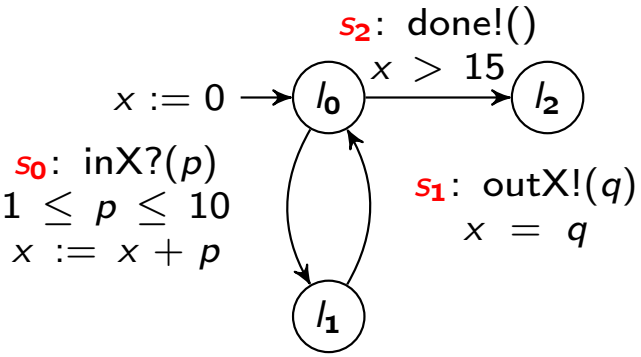
$\longrightarrow (l_0, \{x \mapsto 0\}, True)$

$\downarrow s_0$

$(l_1, \{x \mapsto 0 + p\}, 1 \leq p \leq 10 \wedge True)$

Switch coverage test generation

- Obtain test purposes using Symbolic Execution



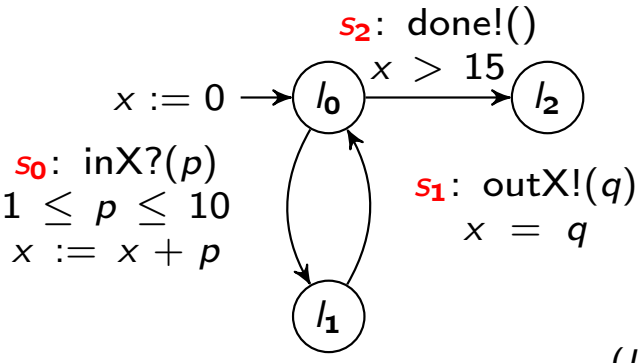
$\longrightarrow (l_0, \{x \mapsto 0\}, True)$

$\downarrow s_0$

$(l_1, \{x \mapsto 0 + p\}, 1 \leq p \leq 10 \wedge True)$

Switch coverage test generation

- Obtain test purposes using Symbolic Execution



$$\longrightarrow (l_0, \{x \mapsto 0\}, True)$$

$$\downarrow s_0$$

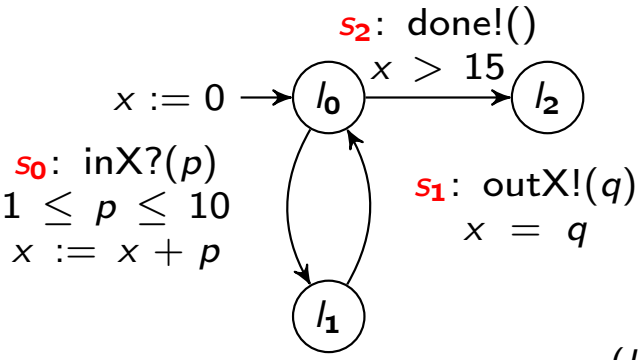
$$(l_1, \{x \mapsto 0 + p\}, 1 \leq p \leq 10 \wedge True)$$

$$\downarrow s_1$$

$$(l_0, \{x \mapsto 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$

Switch coverage test generation

- Obtain test purposes using Symbolic Execution



$$\longrightarrow (l_0, \{x \mapsto 0\}, True)$$

$$\downarrow s_0$$

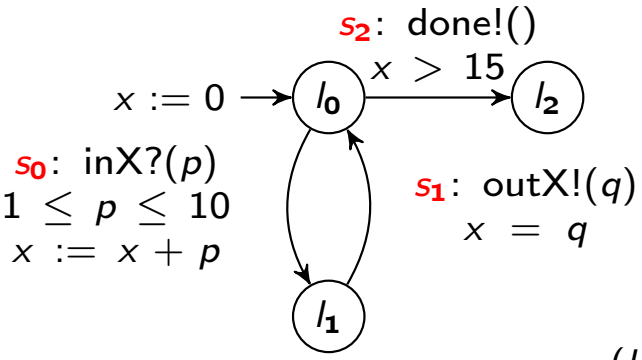
$$(l_1, \{x \mapsto 0 + p\}, 1 \leq p \leq 10 \wedge True)$$

$$\downarrow s_1$$

$$(l_0, \{x \mapsto 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$

Switch coverage test generation

- Obtain test purposes using Symbolic Execution



$$\longrightarrow (l_0, \{x \mapsto 0\}, True)$$

$$\downarrow s_0$$

$$(l_1, \{x \mapsto 0 + p\}, 1 \leq p \leq 10 \wedge True)$$

$$\downarrow s_1$$

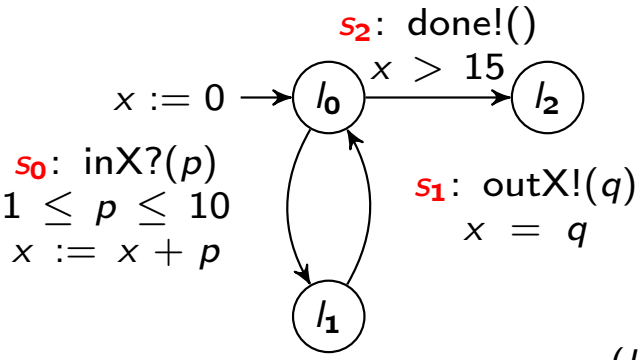
$$(l_0, \{x \mapsto 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$

$$\downarrow s_0$$

$$(l_1, \{x \mapsto 0 + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'' \wedge 1 \leq p'' \leq 10)$$

Switch coverage test generation

- Obtain test purposes using Symbolic Execution



$$\longrightarrow (l_0, \{x \mapsto 0\}, True)$$

$$\downarrow s_0$$

$$(l_1, \{x \mapsto 0 + p\}, 1 \leq p \leq 10 \wedge True)$$

$$\downarrow s_1$$

$$(l_0, \{x \mapsto 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$

$$\downarrow s_0$$

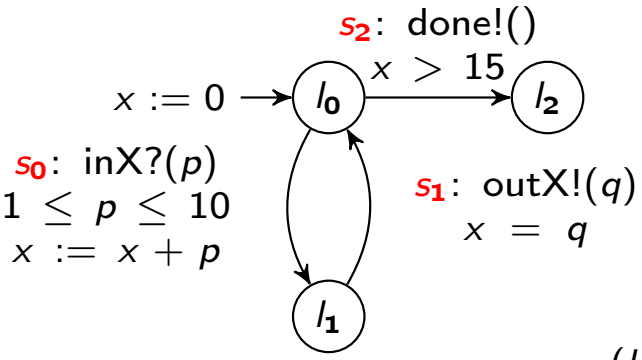
$$(l_1, \{x \mapsto 0 + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'' \wedge 1 \leq p'' \leq 10)$$

$$\downarrow s_1$$

$$(l_0, \{x \mapsto 0 + p''' + p'\}, q = 0 + p''' + p' \wedge 1 \leq p' \leq 10 \wedge q'' = 0 + p''' \wedge 1 \leq p''' \leq 10)$$

Switch coverage test generation

- Obtain test purposes using Symbolic Execution



$$\longrightarrow (l_0, \{x \mapsto 0\}, True)$$

$$\downarrow s_0$$

$$(l_1, \{x \mapsto 0 + p\}, 1 \leq p \leq 10 \wedge True)$$

$$\downarrow s_1$$

$$(l_0, \{x \mapsto 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$

$$\downarrow s_0$$

$$(l_1, \{x \mapsto 0 + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'' \wedge 1 \leq p'' \leq 10)$$

$$\downarrow s_1$$

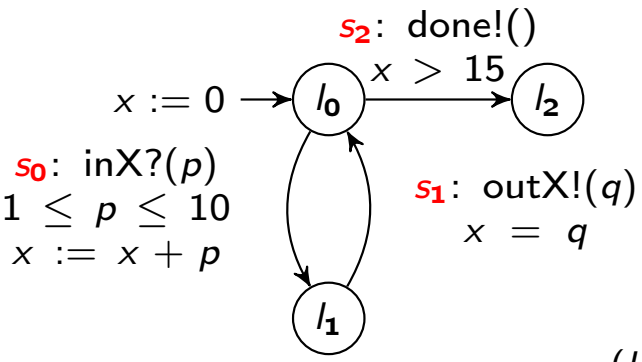
$$(l_0, \{x \mapsto 0 + p''' + p'\}, q = 0 + p''' + p' \wedge 1 \leq p' \leq 10 \wedge q'' = 0 + p''' \wedge 1 \leq p''' \leq 10)$$

$$\swarrow s_0$$

$$(l_0, \{x \mapsto 0 + p'''' + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10)$$

Switch coverage test generation

- Obtain test purposes using Symbolic Execution



$$\longrightarrow (l_0, \{x \mapsto 0\}, True)$$

$$\downarrow s_0$$

$$(l_1, \{x \mapsto 0 + p\}, 1 \leq p \leq 10 \wedge True)$$

$$\downarrow s_1$$

$$(l_0, \{x \mapsto 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$

$$\downarrow s_0$$

$$(l_1, \{x \mapsto 0 + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'' \wedge 1 \leq p'' \leq 10)$$

$$\downarrow s_1$$

$$(l_0, \{x \mapsto 0 + p''' + p'\}, q = 0 + p''' + p' \wedge 1 \leq p' \leq 10 \wedge q'' = 0 + p''' \wedge 1 \leq p''' \leq 10)$$

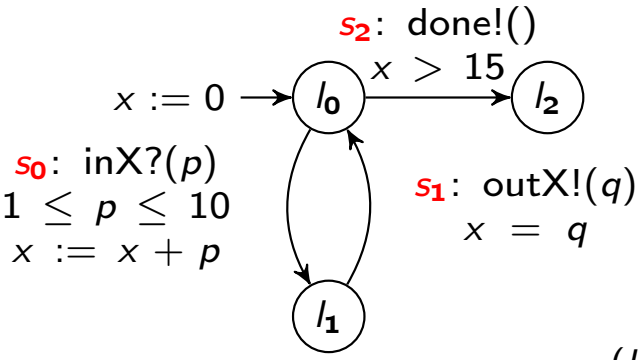
s_0 ↙ $\quad$ s_2 ↘

$$(l_0, \{x \mapsto 0 + p'''' + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10)$$

$$(l_2, \{x \mapsto 0 + p'''' + p''\}, 0 + p'''' + p'' > 15 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10)$$

Switch coverage test generation

- Obtain test purposes using Symbolic Execution



$$\longrightarrow (l_0, \{x \mapsto 0\}, True)$$

$$\downarrow s_0$$

$$(l_1, \{x \mapsto 0 + p\}, 1 \leq p \leq 10 \wedge True)$$

$$\downarrow s_1$$

$$(l_0, \{x \mapsto 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$

$$\downarrow s_0$$

$$(l_1, \{x \mapsto 0 + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'' \wedge 1 \leq p'' \leq 10)$$

$$\downarrow s_1$$

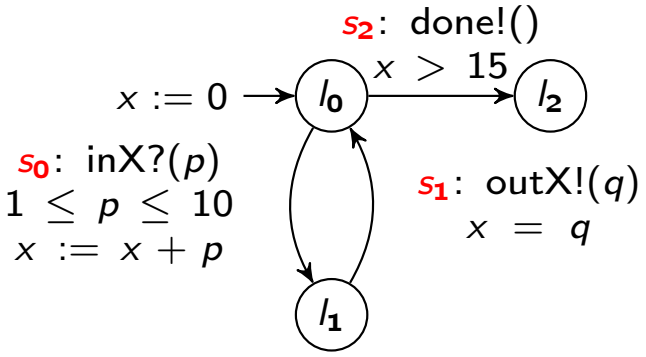
$$(l_0, \{x \mapsto 0 + p''' + p'\}, q = 0 + p''' + p' \wedge 1 \leq p' \leq 10 \wedge q'' = 0 + p''' \wedge 1 \leq p''' \leq 10)$$

$$\swarrow s_0 \qquad \searrow s_2$$

$$(l_0, \{x \mapsto 0 + p'''' + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10)$$

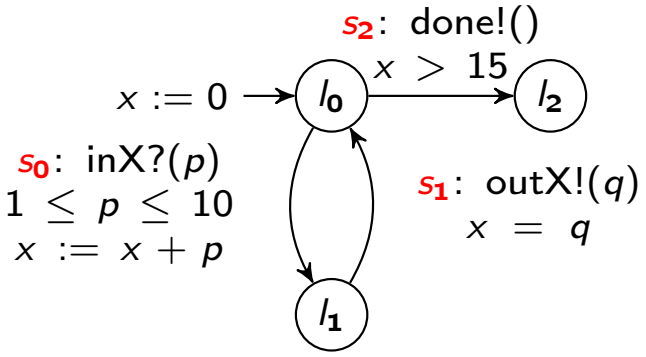
$$(l_2, \{x \mapsto 0 + p'''' + p''\}, 0 + p'''' + p'' > 15 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10)$$

Switch coverage test execution



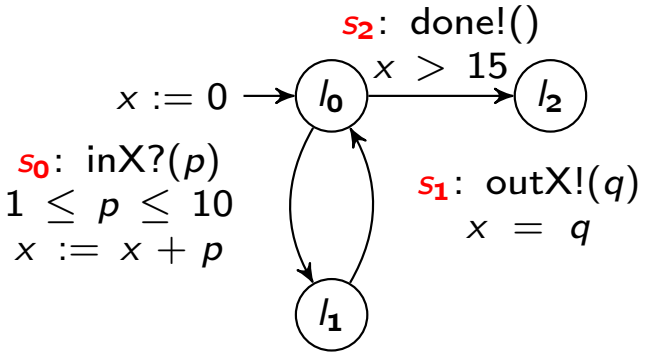
- A test purpose consists of
  - Sequence of swiches: $s_0 s_1 s_0 s_1 s_2$
  - Path condition: $0 + p'''' + p'' > 15 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10$

Switch coverage test execution



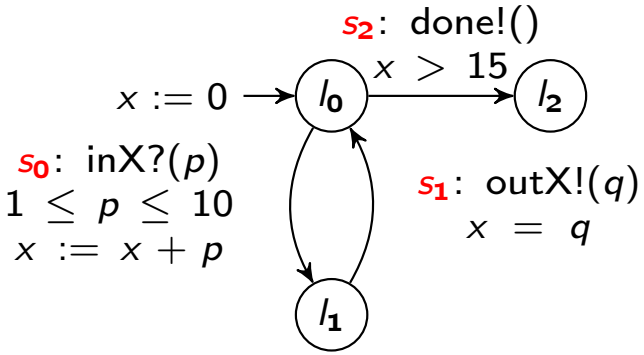
- A test purpose consists of
  - Sequence of swiches: $s_0 s_1 s_0 s_1 s_2$
  - Path condition: $0 + p'''' + p'' > 15 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10$
- Input switch s_0: SMT-solver: SAT, $p'''' = 6$,
- State: $(l_1, x := 6)$

Switch coverage test execution



- A test purpose consists of
  - Sequence of swiches: $s_0 s_1 s_0 s_1 s_2$
  - Path condition: $0 + p'''' + p'' > 15 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10$
- Input switch s_0: SMT-solver: SAT, $p'''' = 6$,
- State: $(l_1, x := 6)$
- Output switch s_1: SUT: outX(6), so $q''' = 6$
- Guard x = 6 holds; state: $(l_0, x := 6)$

Switch coverage test execution



- A test purpose consists of
  - Sequence of swiches: $s_0 s_1 s_0 s_1 s_2$
  - Path condition: $0 + p'''' + p'' > 15 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10$
- Input switch s_0: SMT-solver: SAT, $p'''' = 6$,
- State: $(l_1, x := 6)$
- Output switch s_1: SUT: outX(6), so $q''' = 6$
- Guard x = 6 holds; state: $(l_0, x := 6)$
- Substitute values: $0 + 6 + p'' > 15 \wedge q' = 0 + 6 + p'' \wedge 1 \leq p'' \leq 10 \wedge 6 = 0 + 6 \wedge 1 \leq 6 \leq 10$

Switch coverage execution (continued)

- What if SMT-solver returns Unknown?
  - Use switch guard

Switch coverage execution (continued)

- What if SMT-solver returns Unknown?
  - Use switch guard
  - Else: return verdict *Inconclusive*

Switch coverage execution (continued)

- What if SMT-solver returns Unknown?
  - Use switch guard
  - Else: return verdict *Inconclusive*
- Verdict:
  - *Inconclusive*: SMT-solver returns Unknown
  - *Pass*: end of test purpose reached
  - *Fail*: Output action/parameter not enabled

Switch coverage execution (continued)

- What if SMT-solver returns Unknown?
  - Use switch guard
  - Else: return verdict *Inconclusive*
- Verdict:
  - *Inconclusive*: SMT-solver returns Unknown
  - *Pass*: end of test purpose reached
  - *Fail*: Output action/parameter not enabled
- Test purposes give a-priori coverage

Switch coverage execution (continued)

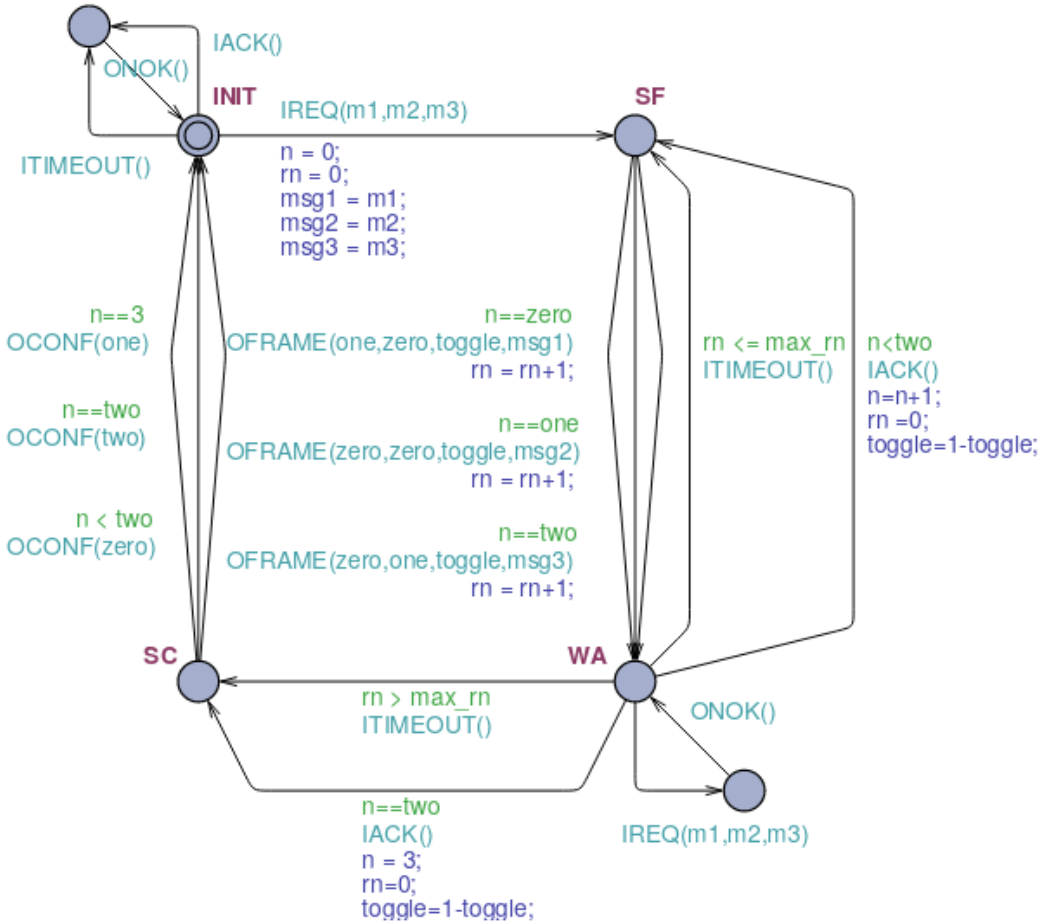
- What if SMT-solver returns Unknown?
  - Use switch guard
  - Else: return verdict *Inconclusive*
- Verdict:
  - *Inconclusive*: SMT-solver returns Unknown
  - *Pass*: end of test purpose reached
  - *Fail*: Output action/parameter not enabled
- Test purposes give a-priori coverage
- After execution: a-posteriori coverage

Implementation

- Obtaining test purposes: Maude & Z3
- Test execution: Python & Z3

Case study: Bounded Retransmission Protocol

- BRP: benchmark from the Automata Wiki (`http://automata.cs.ru.nl/`)



Experimental setup

- Compare:
  - Switch coverage testing (own implementation)
  - Random testing by TorXakis
- Count nr of switches before obtaining Fail

Experimental setup

- Compare:
  - Switch coverage testing (own implementation)
  - Random testing by TorXakis
- Count nr of switches before obtaining Fail

	Switch coverage	TorXakis
Mutant 1	44	12
Mutant 2	16	234
Mutant 3	8	12
Mutant 4	6	18
Mutant 5	18	1620
Mutant 6	164	76
Sum	256	1972
Geometric mean	21.5	64.9

Experimental setup

- Compare:
  - Switch coverage testing (own implementation)
  - Random testing by TorXakis
- Count nr of switches before obtaining Fail

	Switch coverage	TorXakis	
Mutant 1	44	12	
Mutant 2	16	234	
Mutant 3	8	12	
Mutant 4	6	18	
Mutant 5	18	1620	
Mutant 6	164	76	
Sum	256	1972	8 times more!
Geometric mean	21.5	64.9	3 times more!

Conclusions

- Switch coverage test selection
- On-the-fly parameter value generation
- Deal with SMT solvers returning Unknown

Future work

- Add data-coverage techniques to switch coverage

Future work

- Add data-coverage techniques to switch coverage
- Add quiescense (δ) to test generation/execution

Future work

- Add data-coverage techniques to switch coverage
- Add quiescense (δ) to test generation/execution
- Universal instead of existential quantification over output parameters

Future work

- Add data-coverage techniques to switch coverage
- Add quiescense (δ) to test generation/execution
- Universal instead of existential quantification over output parameters
- Improve efficiency: execute test purposes simultaneously

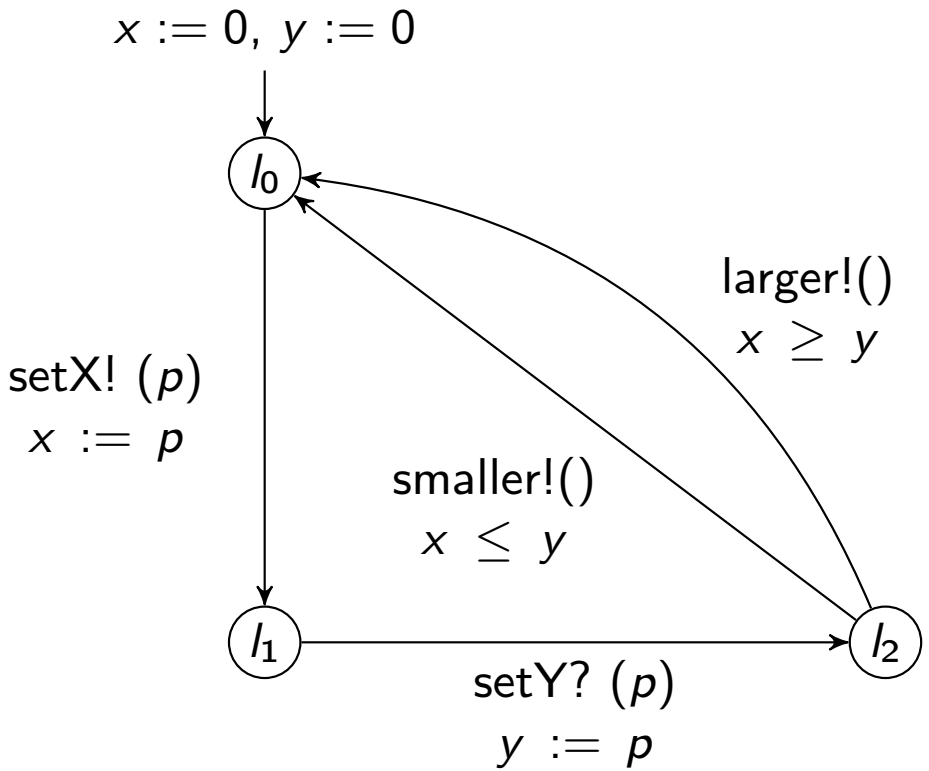
Future work

- Add data-coverage techniques to switch coverage
- Add quiescense (δ) to test generation/execution
- Universal instead of existential quantification over output parameters
- Improve efficiency: execute test purposes simultaneously
- More case studies

Thank you for listening!

Questions?

Generate inputs at latest moment



Cannot choose inputs parameters that always work