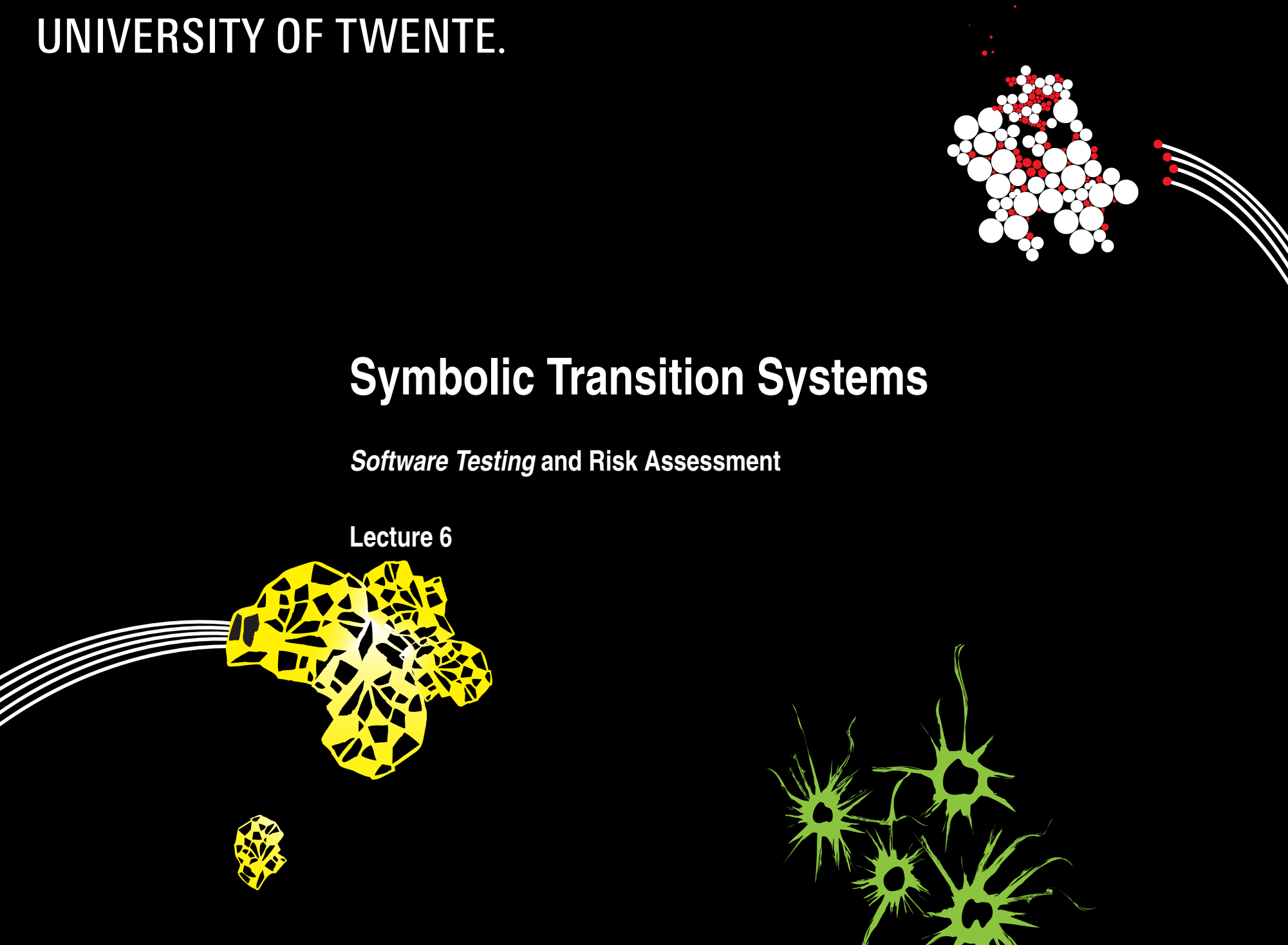
UNIVERSITY OF TWENTE.



Symbolic Transition Systems

Software Testing and Risk Assessment

Lecture 6

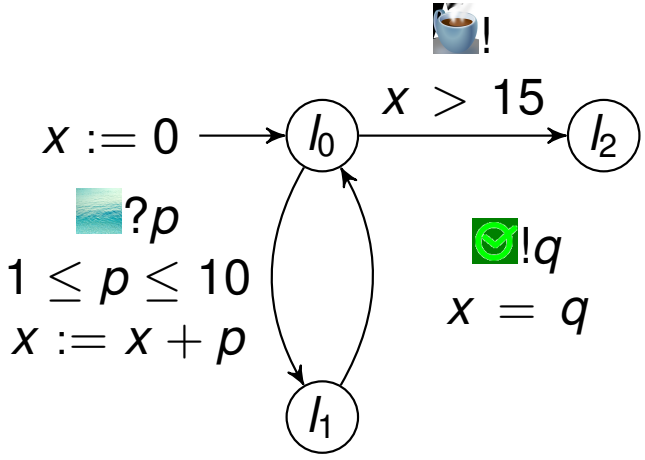
Today

- Symbolic Transition Systems
- Relation between STS and LTS
- On-the-fly test generation and execution for STS
- Switch coverage test generation and execution
- Case study: Bounded Retransmission Protocol (if time allows)

Symbolic Transition System

Testing Systems with Control Flow and Data

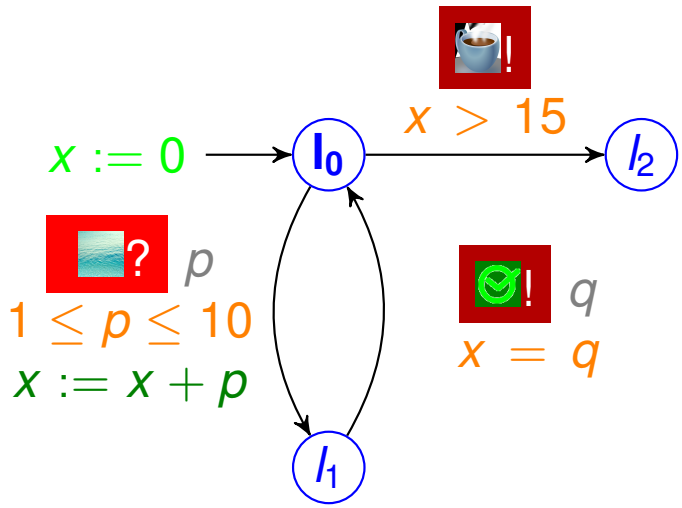
- Present-day systems use both:
  - control flow: states, actions
  - data: variables, parameters, conditions . . .
- Symbolic Transition Systems for modelling real systems!



Literature

- Coverage-Based Testing with Symbolic Transition Systems.
  Petra van den Bos, Jan Tretmans.
  `https://doi.org/10.1007/978-3-030-31157-5_5`

Symbolic Transition System (STS)



- Locations
- Initial assignment (assigns values to location variables)
- Input gates
- Output gates
- Gate parameters
- Guards (boolean expresssions)
- Assignments to location variables
- Switches

Symbolic Transition Systems: definition

A Symbolic Transition System (STS) with inputs and outputs is a tuple $(\mathcal{L}, l_0, \mathcal{V}_I, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L}$ is a finite set of *locations*,

Symbolic Transition Systems: definition

A Symbolic Transition System (STS) with inputs and outputs is a tuple $(\mathcal{L}, l_0, \mathcal{V}_I, \mathit{ini}, \mathcal{V}_P, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L}$ is a finite set of *locations*,
- $l_0 \in \mathcal{L}$ is the *initial location*,

Symbolic Transition Systems: definition

A Symbolic Transition System (STS) with inputs and outputs is a tuple $(\mathcal{L}, l_0, \mathcal{V}_l, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L}$ is a finite set of *locations*,
- $l_0 \in \mathcal{L}$ is the *initial location*,
- $\mathcal{V}_l$ is a finite set of *location variables*,

Symbolic Transition Systems: definition

A Symbolic Transition System (STS) with inputs and outputs is a tuple $(\mathcal{L}, l_0, \mathcal{V}_l, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L}$ is a finite set of *locations*,
- $l_0 \in \mathcal{L}$ is the *initial location*,
- $\mathcal{V}_l$ is a finite set of *location variables*,
- $ini \in \mathcal{T}(\emptyset)^{\mathcal{V}_l}$ is the *initialization*,

Symbolic Transition Systems: definition

A Symbolic Transition System (STS) with inputs and outputs is a tuple $(\mathcal{L}, l_0, \mathcal{V}_l, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L}$ is a finite set of *locations*,
- $l_0 \in \mathcal{L}$ is the *initial location*,
- $\mathcal{V}_l$ is a finite set of *location variables*,
- $ini \in \mathcal{T}(\emptyset)^{\mathcal{V}_l}$ is the *initialization*,
- $\mathcal{V}_p$ is a finite set of *gate parameters* such that $\mathcal{V}_p \cap \mathcal{V}_l = \emptyset$,

Symbolic Transition Systems: definition

A Symbolic Transition System (STS) with inputs and outputs is a tuple $(\mathcal{L}, l_0, \mathcal{V}_l, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L}$ is a finite set of *locations*,
- $l_0 \in \mathcal{L}$ is the *initial location*,
- $\mathcal{V}_l$ is a finite set of *location variables*,
- $ini \in \mathcal{T}(\emptyset)^{\mathcal{V}_l}$ is the *initialization*,
- $\mathcal{V}_p$ is a finite set of *gate parameters* such that $\mathcal{V}_p \cap \mathcal{V}_l = \emptyset$,
- Γ_I is a finite set of input *gates*,

Symbolic Transition Systems: definition

A Symbolic Transition System (STS) with inputs and outputs is a tuple $(\mathcal{L}, l_0, \mathcal{V}_l, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

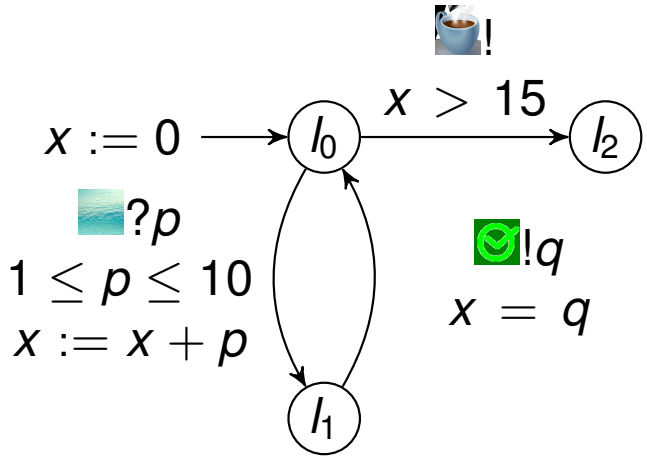
- $\mathcal{L}$ is a finite set of *locations*,
- $l_0 \in \mathcal{L}$ is the *initial location*,
- $\mathcal{V}_l$ is a finite set of *location variables*,
- $ini \in \mathcal{T}(\emptyset)^{\mathcal{V}_l}$ is the *initialization*,
- $\mathcal{V}_p$ is a finite set of *gate parameters* such that $\mathcal{V}_p \cap \mathcal{V}_l = \emptyset$,
- Γ_I is a finite set of input *gates*,
- Γ_O is a finite set of output *gates* such that $\Gamma_I \cap \Gamma_O = \emptyset$,

Symbolic Transition Systems: definition

A Symbolic Transition System (STS) with inputs and outputs is a tuple $(\mathcal{L}, l_0, \mathcal{V}_l, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L}$ is a finite set of *locations*,
- $l_0 \in \mathcal{L}$ is the *initial location*,
- $\mathcal{V}_l$ is a finite set of *location variables*,
- $ini \in \mathcal{T}(\emptyset)^{\mathcal{V}_l}$ is the *initialization*,
- $\mathcal{V}_p$ is a finite set of *gate parameters* such that $\mathcal{V}_p \cap \mathcal{V}_l = \emptyset$,
- Γ_I is a finite set of input *gates*,
- Γ_O is a finite set of output *gates* such that $\Gamma_I \cap \Gamma_O = \emptyset$,
- $\mathcal{R} \subseteq \mathcal{L} \times (\Gamma_I \cup \Gamma_O) \times \mathcal{V}_p^* \times \mathcal{T}_{\texttt{Bool}}(\mathcal{V}_l \cup \mathcal{V}_p) \times \mathcal{T}(\mathcal{V}_l \cup \mathcal{V}_p)^{\mathcal{V}_l} \times \mathcal{L}$ is the *switch relation* with a finite number of elements.

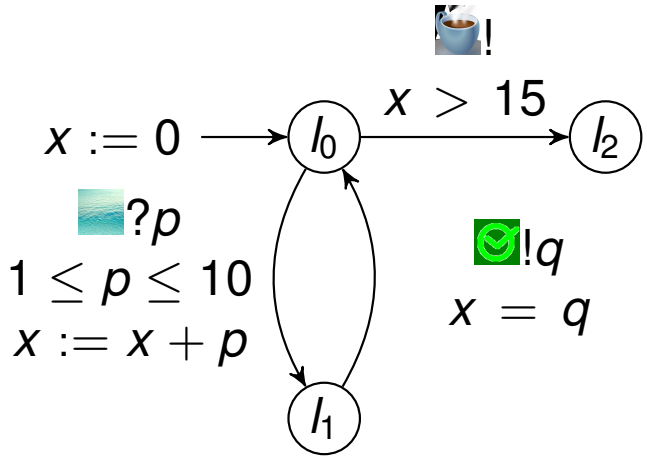
Using the STS definition



This is a graphical representation of $(\mathcal{L}, l_0, \mathcal{V}_I, \mathit{ini}, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L} = \{l_0, l_1, l_2\}$ is its finite set of *locations*,

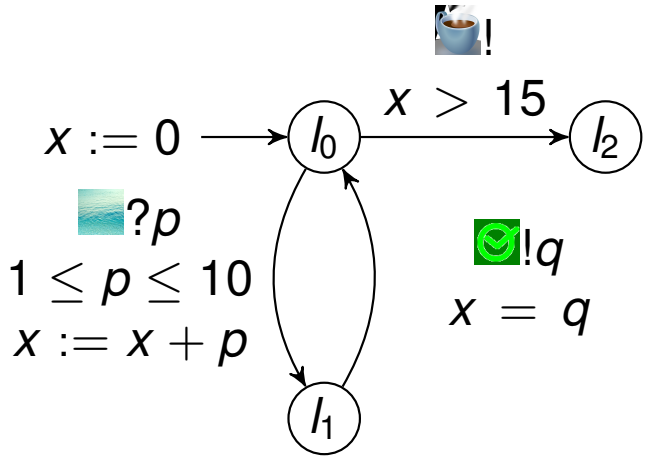
Using the STS definition



This is a graphical representation of $(\mathcal{L}, l_0, \mathcal{V}_I, \mathit{ini}, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L} = \{l_0, l_1, l_2\}$ is its finite set of *locations*,
- $l_0 \in \mathcal{L}$ is its *initial location*,

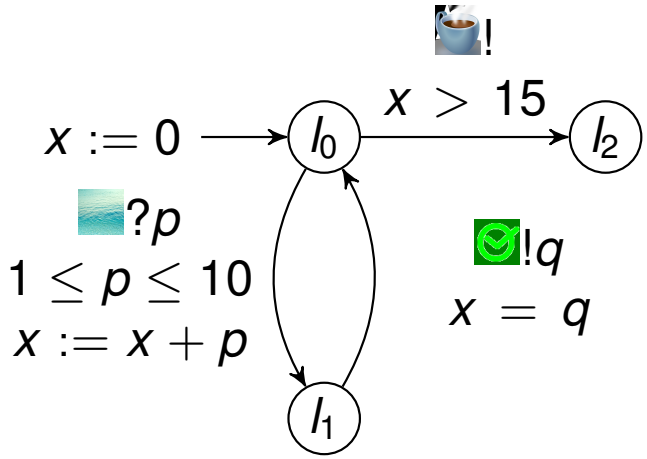
Using the STS definition



This is a graphical representation of $(\mathcal{L}, l_0, \mathcal{V}_l, \mathit{ini}, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L} = \{l_0, l_1, l_2\}$ is its finite set of *locations*,
- $l_0 \in \mathcal{L}$ is its *initial location*,
- $\mathcal{V}_l = \{x\}$ is its finite set of *location variables*,

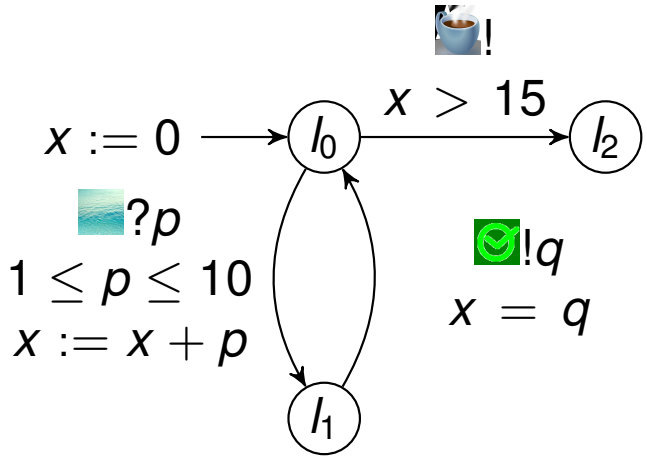
Using the STS definition



This is a graphical representation of $(\mathcal{L}, l_0, \mathcal{V}_l, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L} = \{l_0, l_1, l_2\}$ is its finite set of *locations*,
- $l_0 \in \mathcal{L}$ is its *initial location*,
- $\mathcal{V}_l = \{x\}$ is its finite set of *location variables*,
- $ini \in \mathcal{T}(\emptyset)^{\mathcal{V}_l} = \{x := 0\}$ is its *initialization*,

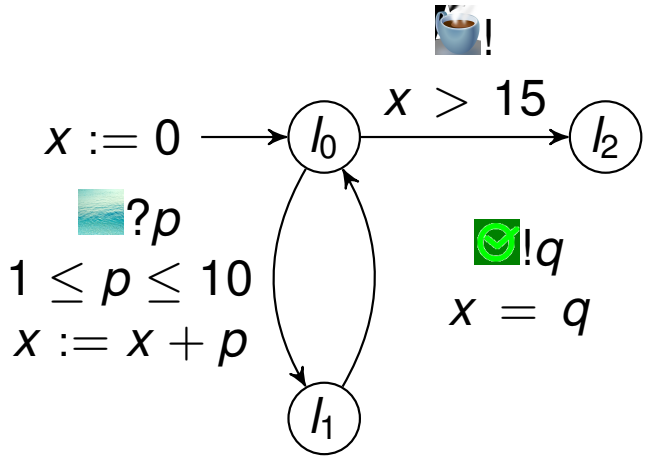
Using the STS definition



This is a graphical representation of $(\mathcal{L}, l_0, \mathcal{V}_I, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L} = \{l_0, l_1, l_2\}$ is its finite set of *locations*,
- $l_0 \in \mathcal{L}$ is its *initial location*,
- $\mathcal{V}_I = \{x\}$ is its finite set of *location variables*,
- $ini \in \mathcal{T}(\emptyset)^{\mathcal{V}_I} = \{x := 0\}$ is its *initialization*,
- $\mathcal{V}_p = \{p, q\}$ is its finite set of *gate parameters* s.t. $\mathcal{V}_p \cap \mathcal{V}_I = \emptyset$,

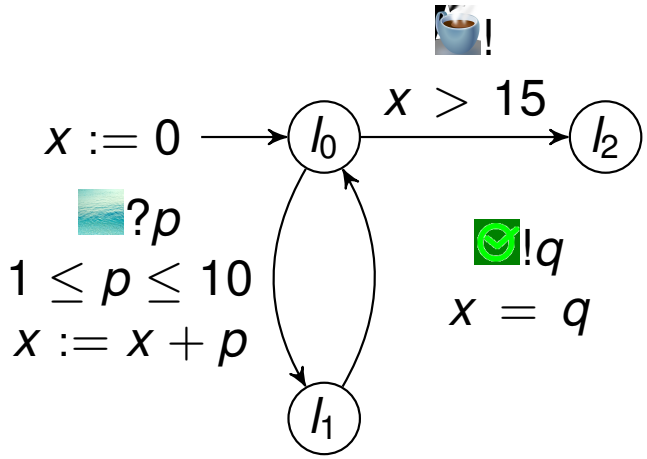
Using the STS definition



This is a graphical representation of $(\mathcal{L}, l_0, \mathcal{V}_l, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L} = \{l_0, l_1, l_2\}$ is its finite set of *locations*,
- $l_0 \in \mathcal{L}$ is its *initial location*,
- $\mathcal{V}_l = \{x\}$ is its finite set of *location variables*,
- $ini \in \mathcal{T}(\emptyset)^{\mathcal{V}_l} = \{x := 0\}$ is its *initialization*,
- $\mathcal{V}_p = \{p, q\}$ is its finite set of *gate parameters* s.t. $\mathcal{V}_p \cap \mathcal{V}_l = \emptyset$,
- $\Gamma_I = \{?\}$ is its finite set of input *gates*,

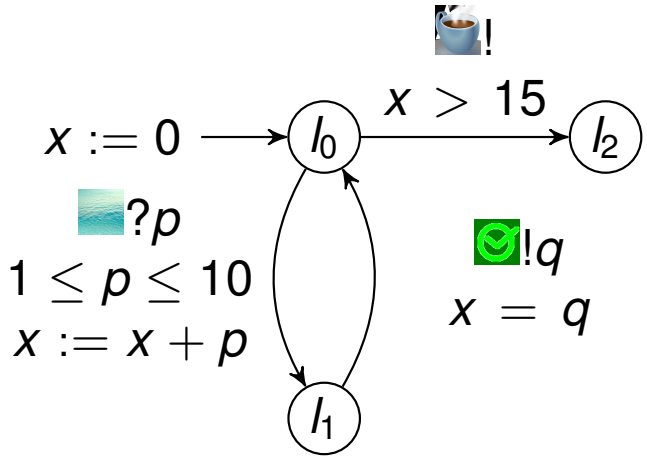
Using the STS definition



This is a graphical representation of $(\mathcal{L}, l_0, \mathcal{V}_l, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

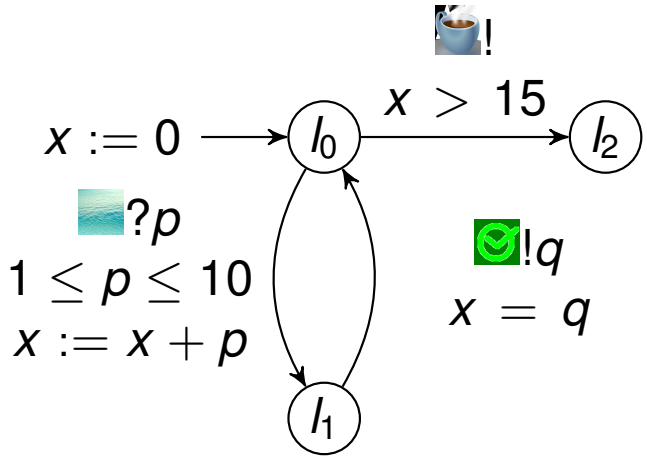
- $\mathcal{L} = \{l_0, l_1, l_2\}$ is its finite set of *locations*,
- $l_0 \in \mathcal{L}$ is its *initial location*,
- $\mathcal{V}_l = \{x\}$ is its finite set of *location variables*,
- $ini \in \mathcal{T}(\emptyset)^{\mathcal{V}_l} = \{x := 0\}$ is its *initialization*,
- $\mathcal{V}_p = \{p, q\}$ is its finite set of *gate parameters* s.t. $\mathcal{V}_p \cap \mathcal{V}_l = \emptyset$,
- $\Gamma_I = \{?\}$ is its finite set of input *gates*,
- $\Gamma_O = \{!, !\}$ is its finite set of output *gates* s.t. $\Gamma_I \cap \Gamma_O = \emptyset$, and
- (see next slide)

Using the STS definition



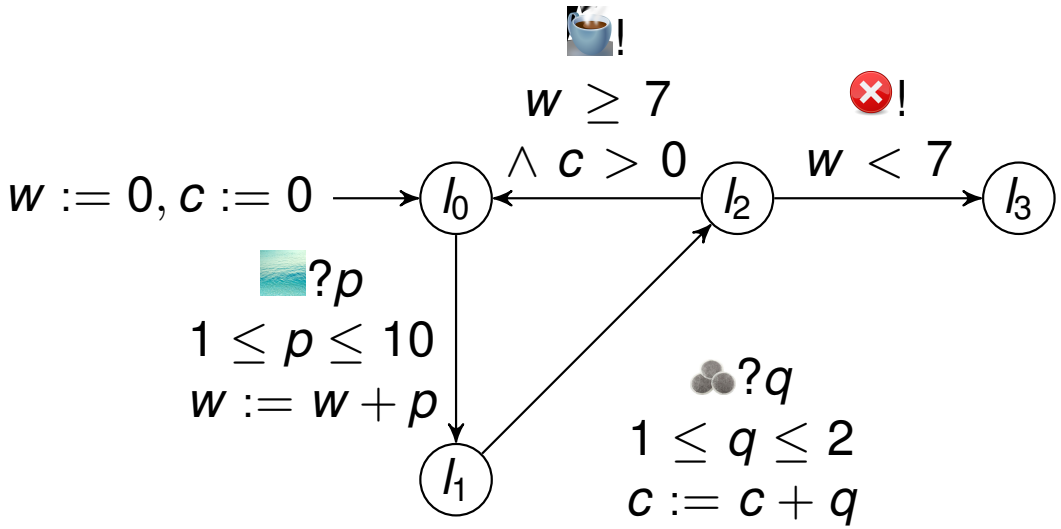
- $\mathcal{R} = \{r_1, r_2, r_3\} \subseteq \mathcal{L} \times (\Gamma_I \cup \Gamma_O) \times \mathcal{V}_p^* \times \mathcal{T}_{\texttt{Bool}}(\mathcal{V}_I \cup \mathcal{V}_p) \times \mathcal{T}(\mathcal{V}_I \cup \mathcal{V}_p)^{\mathcal{V}_I} \times \mathcal{L}$ is its *switch relation* with 3 (i.e. a finite number of) elements, namely:

Using the STS definition



- $\mathcal{R} = \{r_1, r_2, r_3\} \subseteq \mathcal{L} \times (\Gamma_I \cup \Gamma_O) \times \mathcal{V}_p^* \times \mathcal{T}_{\texttt{Bool}}(\mathcal{V}_I \cup \mathcal{V}_p) \times \mathcal{T}(\mathcal{V}_I \cup \mathcal{V}_p)^{\mathcal{V}_I} \times \mathcal{L}$ is its *switch relation* with 3 (i.e. a finite number of) elements, namely:
  - $r_0 = (l_0, ?, p, 1 \leq p \leq 10, \{x := x + p\}, l_1)$
  - $r_1 = (l_1, !, q, x = q, \{x := x\}, l_0)$
  - $r_2 = (l_0, !, \epsilon, x > 15, \{x := x\}, l_2)$

Example STS 1: formal definition?

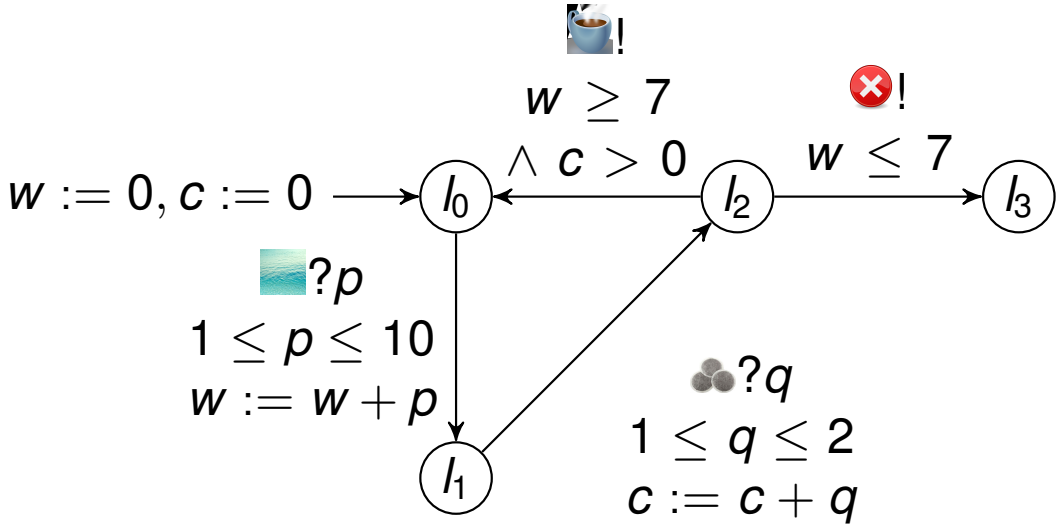


An STS is a tuple $(\mathcal{L}, l_0, \mathcal{V}_l, \mathit{ini}, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$ where:

- $\mathcal{L}$ is a finite set of *locations*,
- $l_0 \in \mathcal{L}$ is the *initial location*,
- $\mathcal{V}_l$ is a finite set of *location variables*,
- $\mathit{ini} \in \mathcal{T}(\emptyset)^{\mathcal{V}_l}$ is the *initialization*,
- $\mathcal{V}_p$ is a finite set of *gate parameters* such that $\mathcal{V}_p \cap \mathcal{V}_l = \emptyset$,
- Γ_I is a finite set of input *gates*,
- Γ_O is a finite set of output *gates*,
- $\mathcal{R} \subseteq \mathcal{L} \times (\Gamma_I \cup \Gamma_O) \times \mathcal{V}_p^* \times \mathcal{T}_{\mathtt{Bool}}(\mathcal{V}_l \cup \mathcal{V}_p) \times \mathcal{T}(\mathcal{V}_l \cup \mathcal{V}_p)^{\mathcal{V}_l} \times \mathcal{L}$ is the *switch relation*

(Answer on board)

Example STS 2: nondeterminism



Some definition details

- Assignments on switches happen simultaneously
  - e.g. $x := y, y := x$

Some definition details

- Assignments on switches happen simultaneously
  - e.g. $x := y, y := x$
- If a location variable is not assigned by a switch its value stays the same
  - e.g. $c := c$

Some definition details

- Assignments on switches happen simultaneously
  - e.g. $x := y, y := x$
- If a location variable is not assigned by a switch its value stays the same
  - e.g. $c := c$
- Scope of variables:
  - Location variables: global to whole STS
  - Gate parameters: local to guard and assignments of switch

Some definition details

- Assignments on switches happen simultaneously
  - e.g. $x := y, y := x$
- If a location variable is not assigned by a switch its value stays the same
  - e.g. $c := c$
- Scope of variables:
  - Location variables: global to whole STS
  - Gate parameters: local to guard and assignments of switch
- For any switch $(l_1, \lambda, p_0 \ldots p_k, \phi, \psi, l_2) \in \mathcal{R}$, we require that:
  - $p_0 \ldots p_k$ is a sequence of distinct variables

Some definition details

- Assignments on switches happen simultaneously
  - e.g. $x := y, y := x$
- If a location variable is not assigned by a switch its value stays the same
  - e.g. $c := c$
- Scope of variables:
  - Location variables: global to whole STS
  - Gate parameters: local to guard and assignments of switch
- For any switch $(l_1, \lambda, p_0 \ldots p_k, \phi, \psi, l_2) \in \mathcal{R}$, we require that:
  - $p_0 \ldots p_k$ is a sequence of distinct variables
  - $\mathtt{type}_g(\lambda) = \mathtt{type}_t(p_0 \ldots p_k)$
    - e.g. $\mathtt{type}_g($  $?) = \mathbb{Z}$ and $\mathtt{type}_t(x) = \mathtt{type}_g(x + p) = \mathbb{Z}$

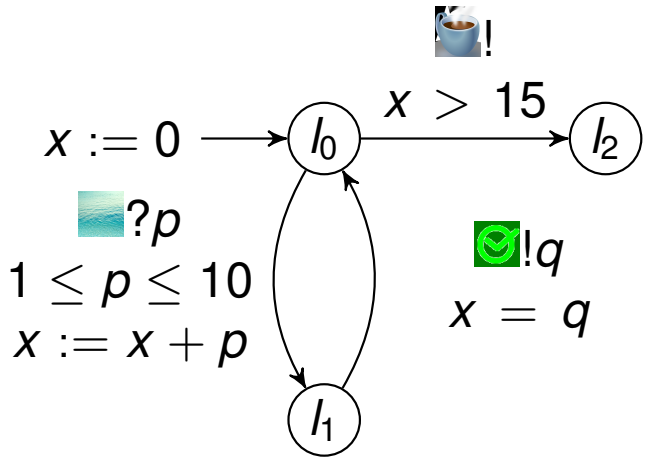
Some definition details

- Assignments on switches happen simultaneously
  - e.g. $x := y, y := x$
- If a location variable is not assigned by a switch its value stays the same
  - e.g. $c := c$
- Scope of variables:
  - Location variables: global to whole STS
  - Gate parameters: local to guard and assignments of switch
- For any switch $(l_1, \lambda, p_0 \ldots p_k, \phi, \psi, l_2) \in \mathcal{R}$, we require that:
  - $p_0 \ldots p_k$ is a sequence of distinct variables
  - $\mathtt{type}_g(\lambda) = \mathtt{type}_t(p_0 \ldots p_k)$
    - e.g. $\mathtt{type}_g($  $?) = \mathbb{Z}$ and $\mathtt{type}_t(x) = \mathtt{type}_g(x + p) = \mathbb{Z}$
  - $\phi \in \mathcal{T}_{\mathtt{Bool}}(\mathcal{V}_l \cup \{p_0, \ldots, p_k\})$

Some definition details

- Assignments on switches happen simultaneously
  - e.g. $x := y, y := x$
- If a location variable is not assigned by a switch its value stays the same
  - e.g. $c := c$
- Scope of variables:
  - Location variables: global to whole STS
  - Gate parameters: local to guard and assignments of switch
- For any switch $(l_1, \lambda, p_0 \ldots p_k, \phi, \psi, l_2) \in \mathcal{R}$, we require that:
  - $p_0 \ldots p_k$ is a sequence of distinct variables
  - $\mathtt{type}_g(\lambda) = \mathtt{type}_t(p_0 \ldots p_k)$
    - e.g. $\mathtt{type}_g($  $?) = \mathbb{Z}$ and $\mathtt{type}_t(x) = \mathtt{type}_g(x + p) = \mathbb{Z}$
  - $\phi \in \mathcal{T}_{\mathtt{Bool}}(\mathcal{V}_l \cup \{p_0, \ldots, p_k\})$
  - $\psi \in \mathcal{T}(\mathcal{V}_l \cup \{p_0, \ldots, p_k\})^{\mathcal{V}_l}$

Q: how to violate the rules?

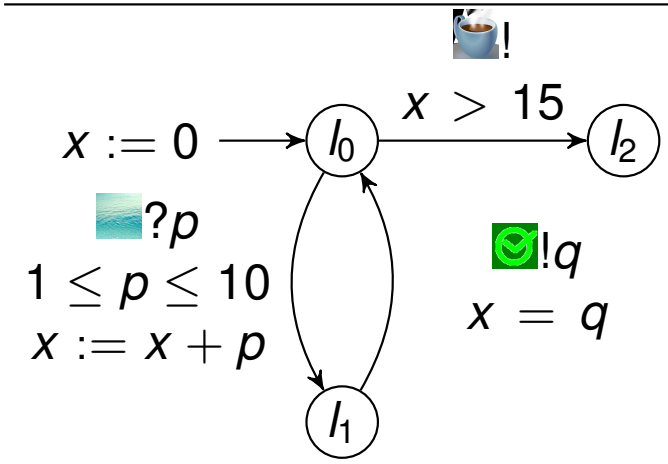


- For any switch $(l_1, \lambda, p_0 \ldots p_k, \phi, \psi, l_2) \in \mathcal{R}$, we require that:
  - $p_0 \ldots p_k$ is a sequence of distinct variables
  - $\mathtt{type}_g(\lambda) = \mathtt{type}_t(p_0 \ldots p_k)$
  - $\phi \in \mathcal{T}_{\mathtt{Bool}}(\mathcal{V}_l \cup \{p_0, \ldots, p_k\})$
  - $\psi \in \mathcal{T}(\mathcal{V}_l \cup \{p_0, \ldots, p_k\})^{\mathcal{V}_l}$

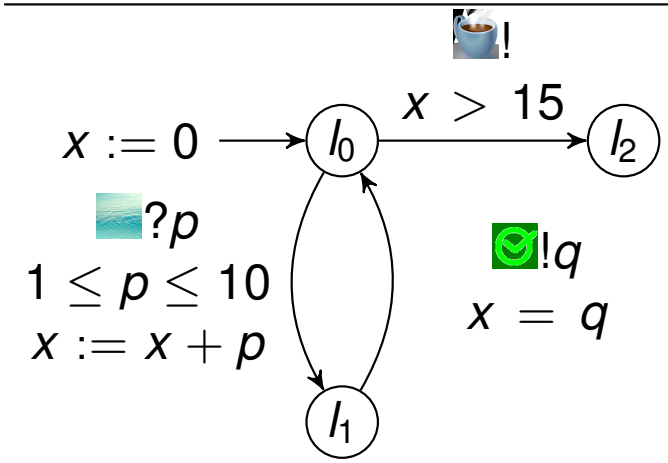
(Answer on board)

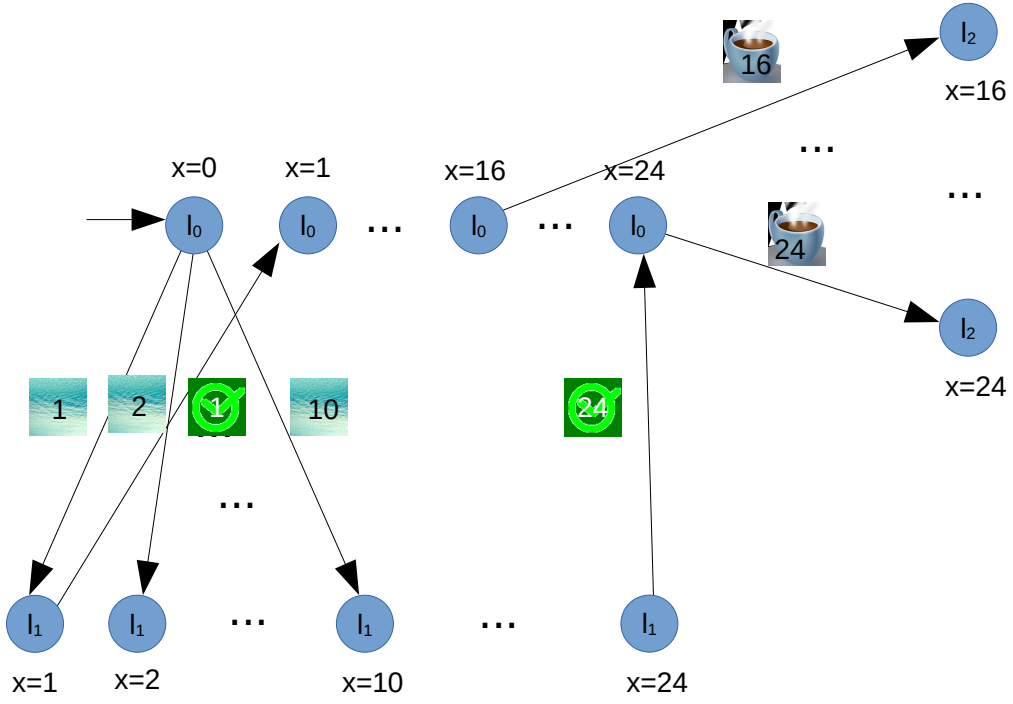
Relation between STS and LTS

Interpretation: the LTS of an STS (an example)



Interpretation: the LTS of an STS (an example)





Interpretation: the LTS of an STS (definition)

- Let S be an STS. Then its *interpretation* is the LTS $S_{intrp} = (Q, L_I, L_U, T, q_0)$ where:

Interpretation: the LTS of an STS (definition)

- Let S be an STS. Then its *interpretation* is the LTS $S_{intrp} = (Q, L_I, L_U, T, q_0)$ where:
  - $Q \subseteq \mathcal{L} \times \mathcal{U}^{\mathcal{V}_I}$ is a set of states
  - $L_I, L_U \subseteq \Gamma \times \mathcal{U}^*$ are sets of labels called gate values such that $\texttt{type}_g(\lambda) = \texttt{type}_v(\bar{w})$

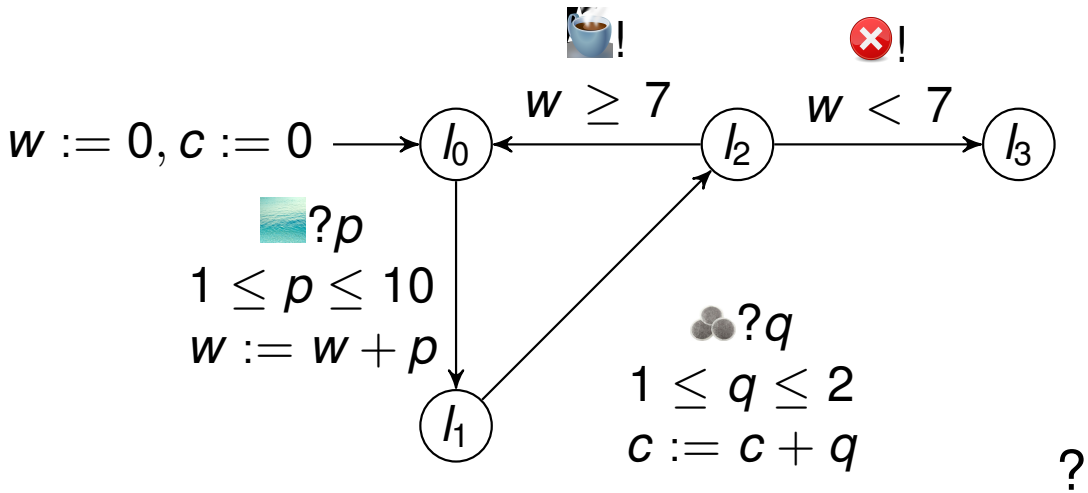
Interpretation: the LTS of an STS (definition)

- Let S be an STS. Then its *interpretation* is the LTS $S_{intrp} = (Q, L_I, L_U, T, q_0)$ where:
  - $Q \subseteq \mathcal{L} \times \mathcal{U}^{\mathcal{V}_l}$ is a set of states
  - $L_I, L_U \subseteq \Gamma \times \mathcal{U}^*$ are sets of labels called gate values such that $\texttt{type}_g(\lambda) = \texttt{type}_v(\bar{w})$
- Hence, interpretation S_{intrp} has states $(l, \alpha) \in Q$ where:
  - $l \in \mathcal{L}$ is a location
  - $\alpha \in \mathcal{U}^{\mathcal{V}_l}$ is an assignment of values to location variables

Interpretation: the LTS of an STS (definition)

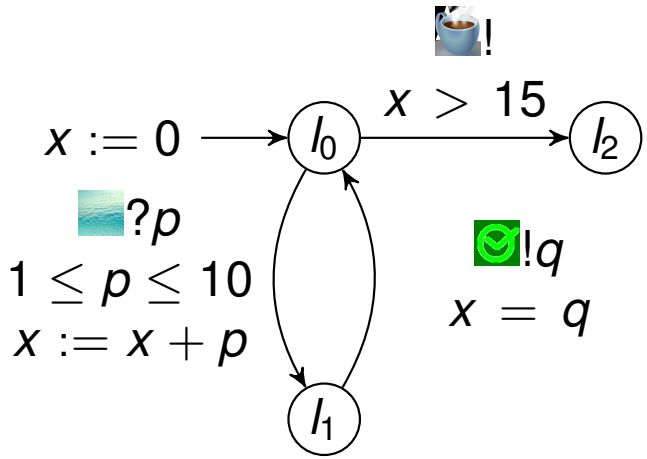
- Let S be an STS. Then its *interpretation* is the LTS $S_{intrp} = (Q, L_I, L_U, T, q_0)$ where:
  - $Q \subseteq \mathcal{L} \times \mathcal{U}^{\mathcal{V}_l}$ is a set of states
  - $L_I, L_U \subseteq \Gamma \times \mathcal{U}^*$ are sets of labels called gate values such that $\texttt{type}_g(\lambda) = \texttt{type}_v(\bar{w})$
- Hence, interpretation S_{intrp} has states $(l, \alpha) \in Q$ where:
  - $l \in \mathcal{L}$ is a location
  - $\alpha \in \mathcal{U}^{\mathcal{V}_l}$ is an assignment of values to location variables
- Hence, interpretation S_{intrp} has labels $(\lambda, \bar{w}) \in L$ where:
  - $\lambda \in \Gamma$ is a gate
  - $\bar{w} \in \mathcal{U}^*$ is a sequence of values

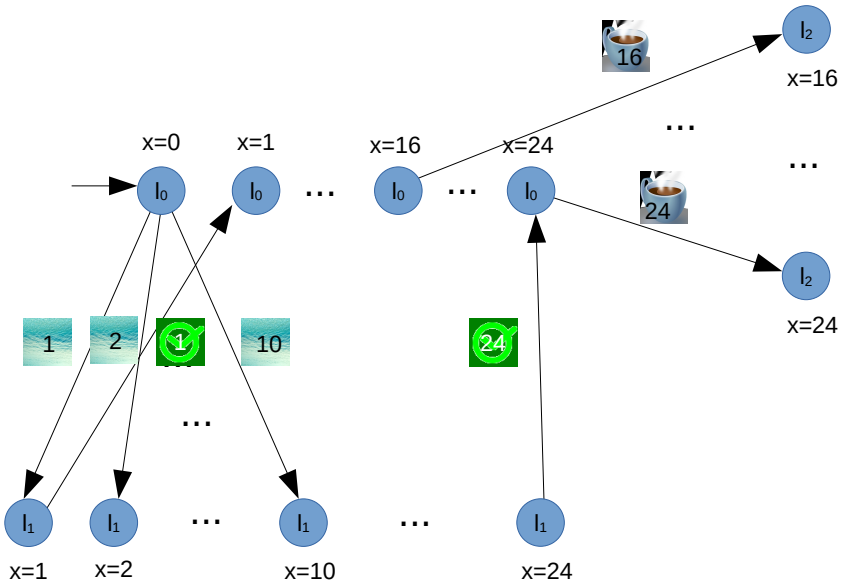
Q: what is the interpretation of:



(Answer on the board)

Interpretation





- Q: how many states does an interpretation have?
- Q: and how many labels?

On-the-fly test generation and execution

On-the-fly test generation

- Algorithm *onTheFlyGen*:
  - **Input:** STS $S = (\mathcal{L}, l_0, \mathcal{V}_I, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$, and current states $Q' = \{(l_0, ini)\}$
  - **Output:** a verdict *Pass* or *Fail*

On-the-fly test generation

- Algorithm *onTheFlyGen*:
  - **Input:** STS $S = (\mathcal{L}, l_0, \mathcal{V}_I, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$, and current states $Q' = \{(l_0, ini)\}$
  - **Output:** a verdict *Pass* or *Fail*
- *onTheFlyGen*(S, Q') = take either of the following steps:

End test case	return verdict *Pass*

On-the-fly test generation

- Algorithm *onTheFlyGen*:
  - **Input:** STS $S = (\mathcal{L}, l_0, \mathcal{V}_I, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$, and current states $Q' = \{(l_0, ini)\}$
  - **Output:** a verdict *Pass* or *Fail*
- $onTheFlyGen(S, Q') =$ take either of the following steps:

End test case	return verdict *Pass*
Observe output $(x!, \bar{w})$	if $(x!, \bar{w}) \in out(Q')$: $onTheFlyGen(S, Q' \ after \ (x!, \bar{w}))$ else: return *Fail*

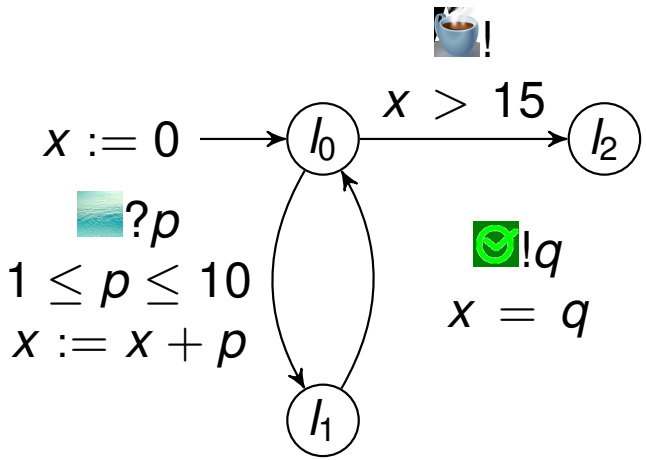
On-the-fly test generation

- Algorithm *onTheFlyGen*:
  - **Input:** STS $S = (\mathcal{L}, l_0, \mathcal{V}_I, ini, \mathcal{V}_p, \Gamma_I, \Gamma_O, \mathcal{R})$, and current states $Q' = \{(l_0, ini)\}$
  - **Output:** a verdict *Pass* or *Fail*
- $onTheFlyGen(S, Q') =$ take either of the following steps:

End test case	return verdict *Pass*
Observe output $(x!, \bar{w})$	if $(x!, \bar{w}) \in out(Q')$: $onTheFlyGen(S, Q' \ after \ (x!, \bar{w}))$ else: return *Fail*
Choose input $(a?, \bar{w})$ where $Q' \ after \ (a?, \bar{w}) \neq \emptyset$	if SUT provides output $(x!, \bar{w}_2)$ alread handle output observation (as abov else: supply $(a?, \bar{w})$ and continue with $onTheFlyGen(S, Q' \ after \ (a?, \bar{w}))$

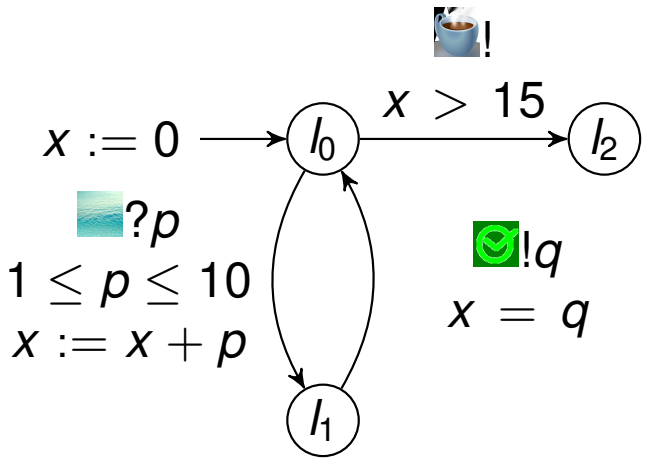
On-the-fly test generation example 1

- Choose input (?,10)

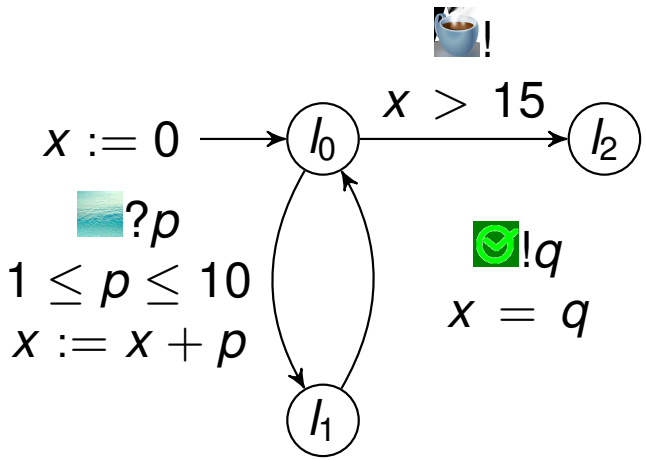


On-the-fly test generation example 1

- Choose input (?,10)
- SUT accepts input

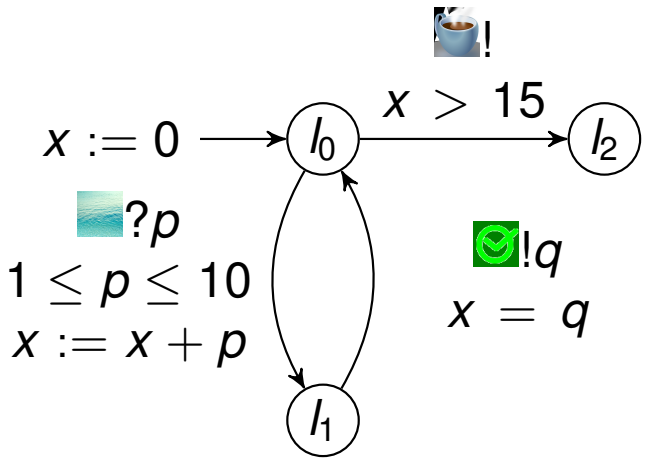


On-the-fly test generation example 1



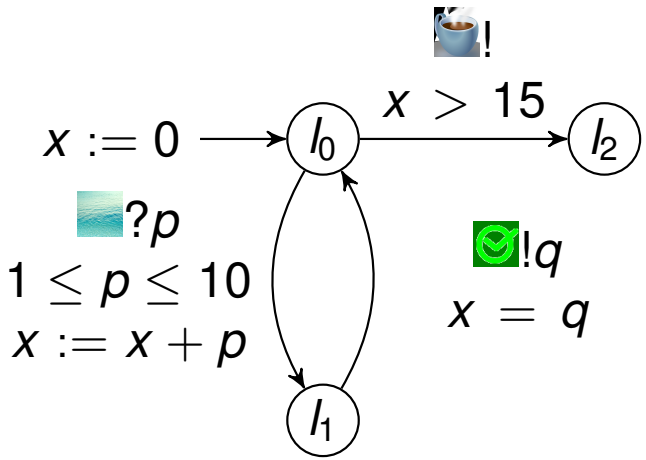
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$

On-the-fly test generation example 1



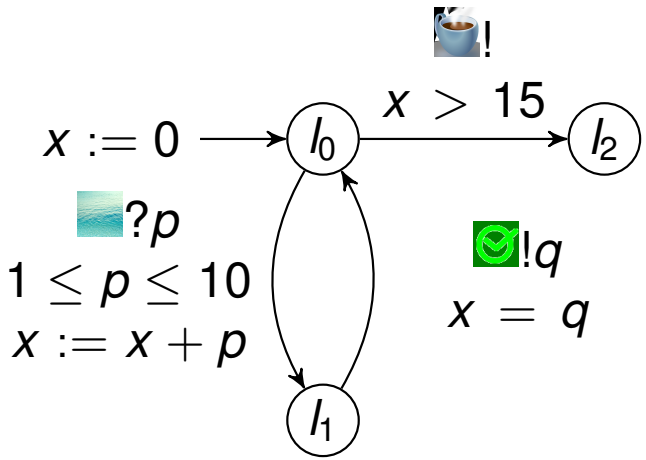
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output

On-the-fly test generation example 1



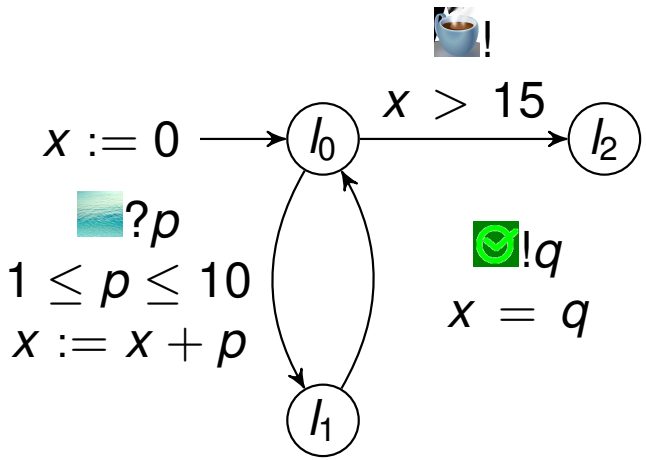
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)

On-the-fly test generation example 1



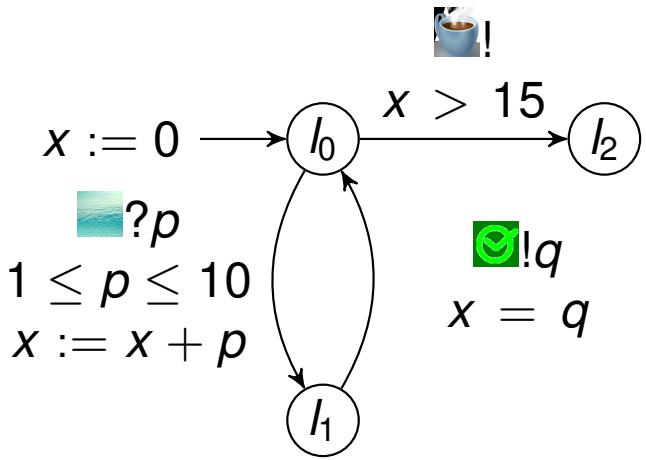
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$

On-the-fly test generation example 1



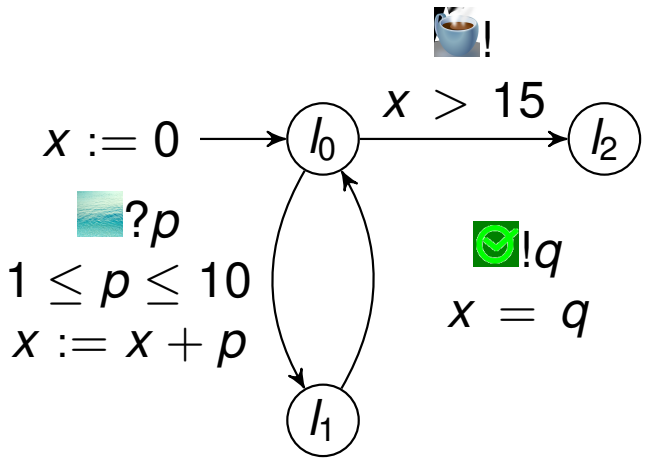
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)

On-the-fly test generation example 1



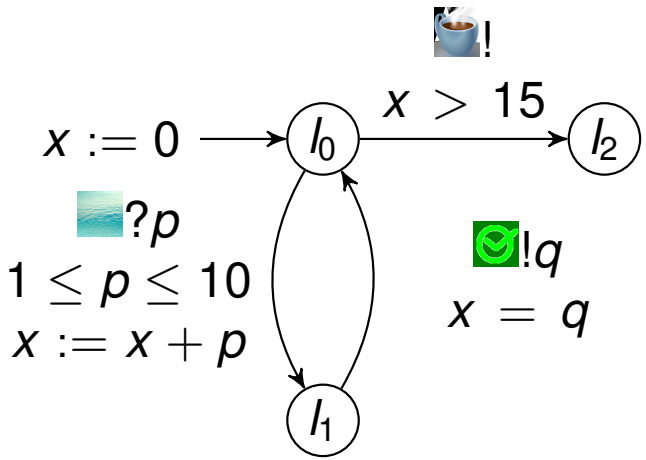
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input

On-the-fly test generation example 1



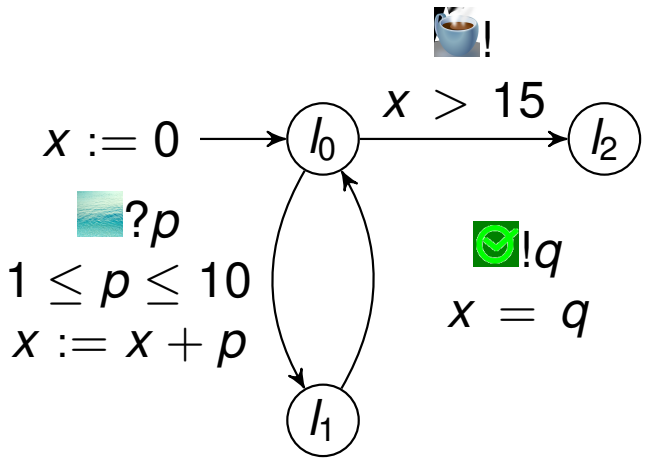
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$

On-the-fly test generation example 1



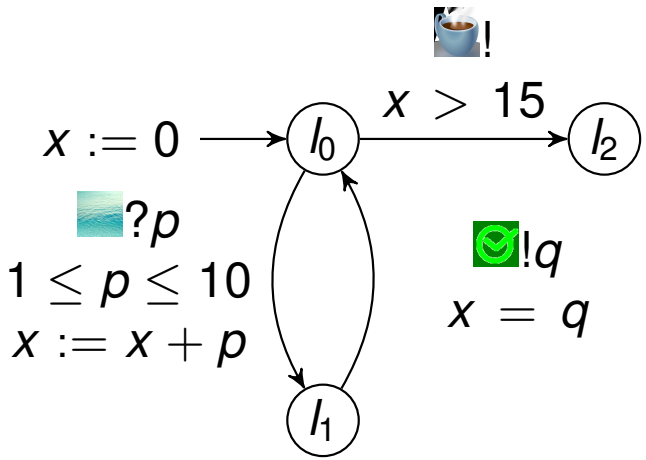
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$
- Observe output

On-the-fly test generation example 1



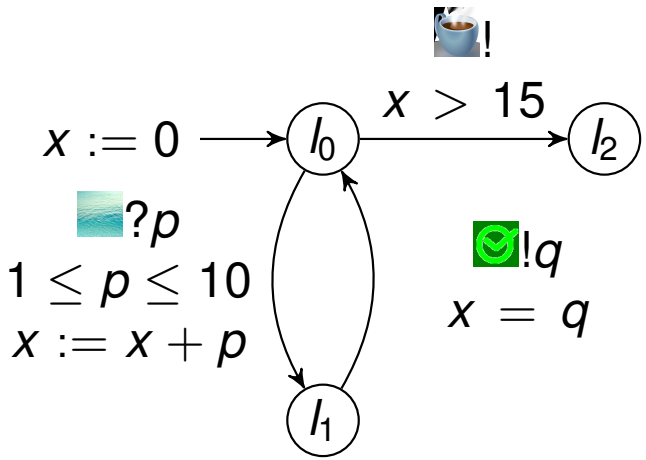
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$
- Observe output
- SUT provides (!, 20)

On-the-fly test generation example 1



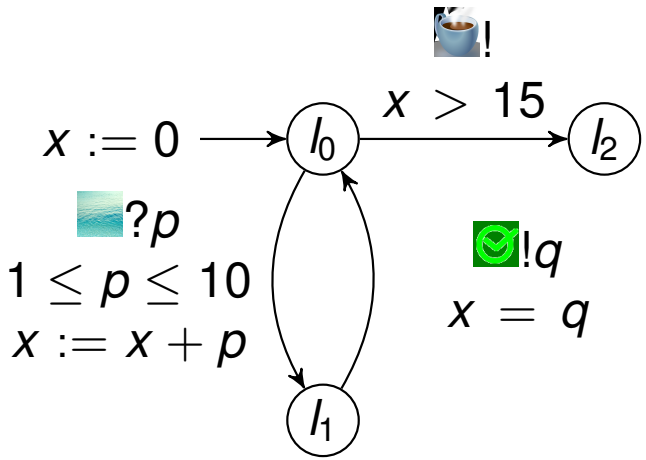
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$
- Observe output
- SUT provides (!, 20)
- Current states: $\{(l_0, x := 20)\}$

On-the-fly test generation example 1



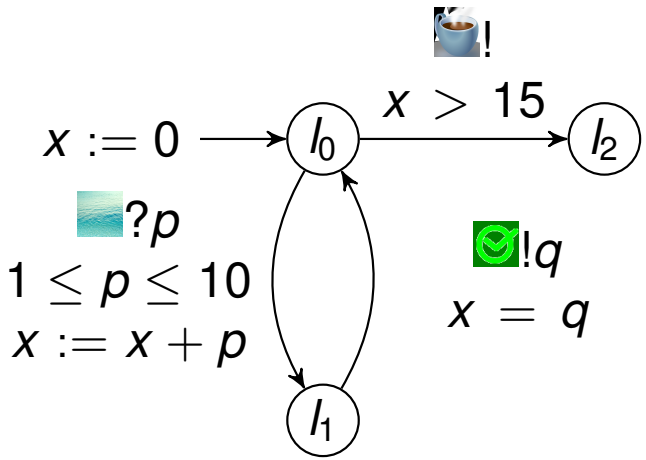
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$
- Observe output
- SUT provides (!, 20)
- Current states: $\{(l_0, x := 20)\}$
- Observe output

On-the-fly test generation example 1



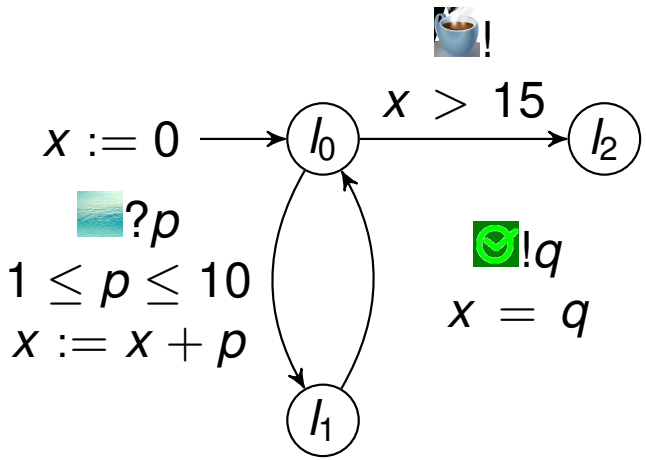
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$
- Observe output
- SUT provides (!, 20)
- Current states: $\{(l_0, x := 20)\}$
- Observe output
- SUT provides (!,)

On-the-fly test generation example 1



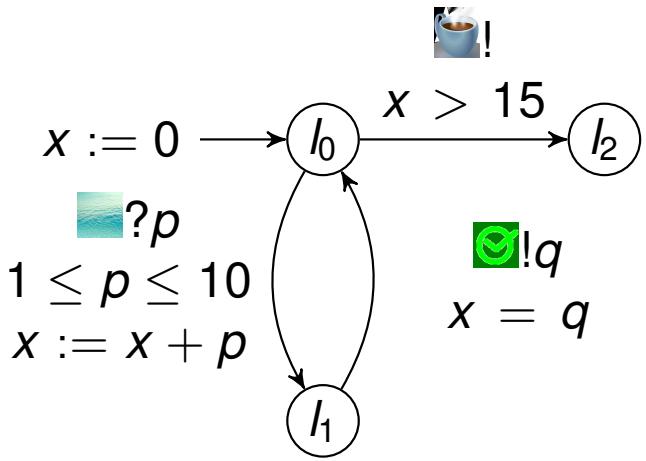
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$
- Observe output
- SUT provides (!, 20)
- Current states: $\{(l_0, x := 20)\}$
- Observe output
- SUT provides (!,)
- Current states: $\{(l_2, x := 20)\}$

On-the-fly test generation example 1



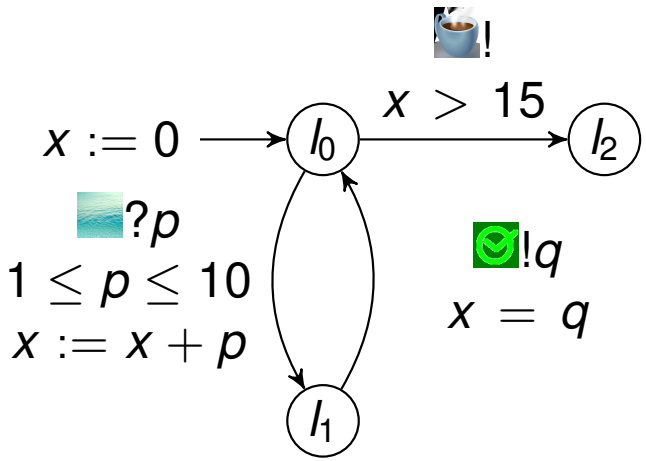
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$
- Observe output
- SUT provides (!, 20)
- Current states: $\{(l_0, x := 20)\}$
- Observe output
- SUT provides (!,)
- Current states: $\{(l_2, x := 20)\}$
- *Pass*

On-the-fly test generation example 2



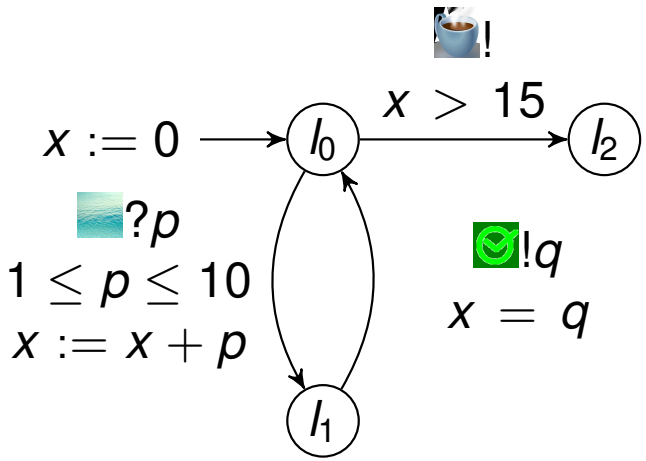
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$
- Observe output

On-the-fly test generation example 2



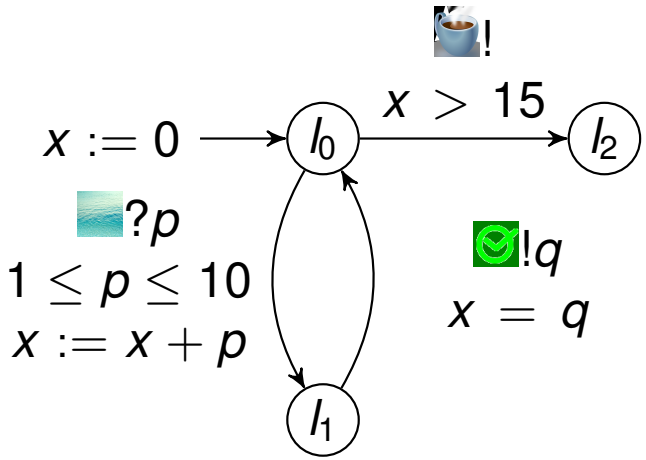
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$
- Observe output
- SUT provides (!, 10)

On-the-fly test generation example 2



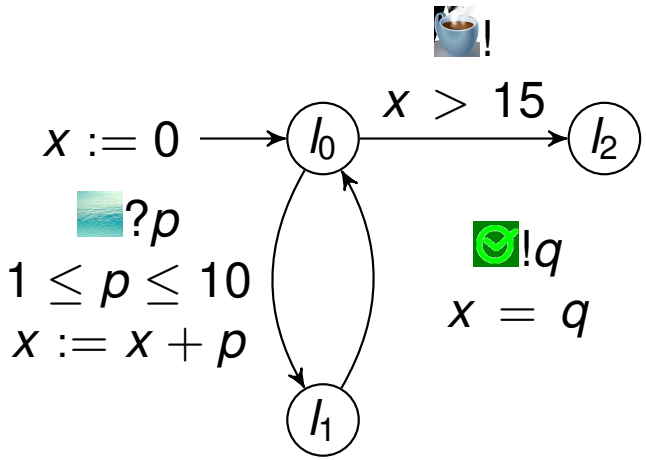
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 10)\}$
- Observe output
- SUT provides (!, 10)
- Current states: $\{(l_0, x := 10)\}$
- Choose input (?,10)
- SUT accepts input
- Current states: $\{(l_1, x := 20)\}$
- Observe output
- SUT provides (!, 10)
- Fail!

On-the-fly test generation example 3



- Observe output
- SUT provides (!,)

On-the-fly test generation example 3

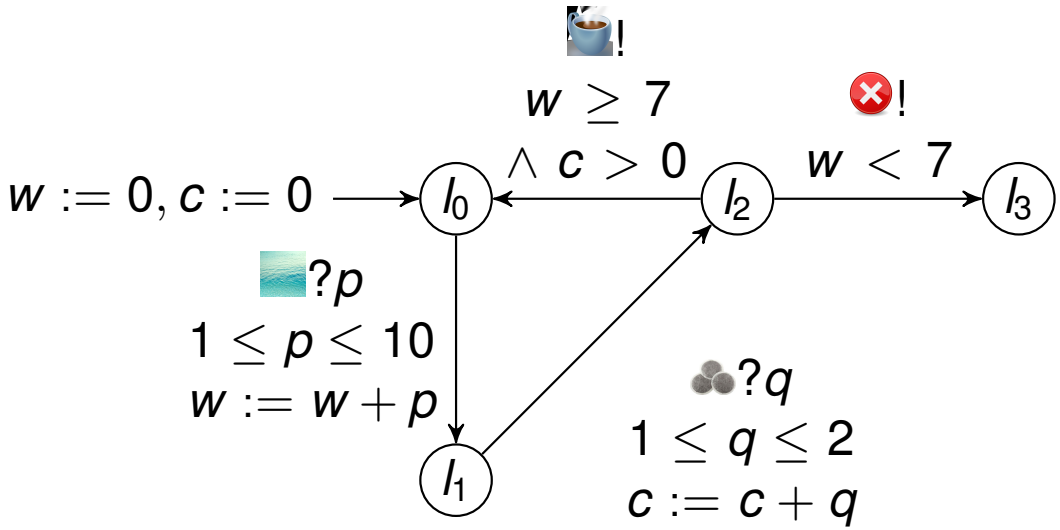


- Observe output
- SUT provides (!,)
- Fail!

Test trace instead of test case

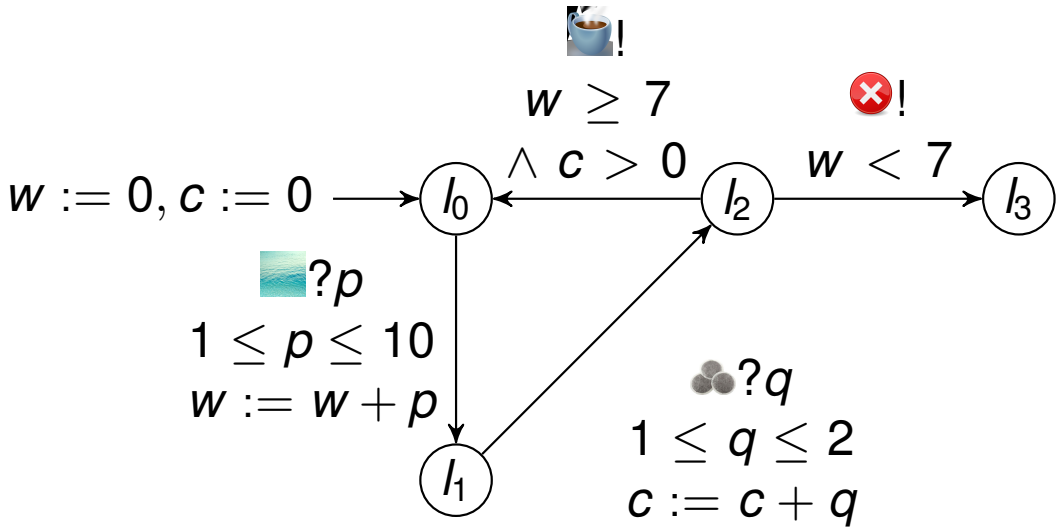
- Apply test trace
- Deviation?
  - Fail or Inconclusive (try again)
- Otherwise: Pass

Test trace example



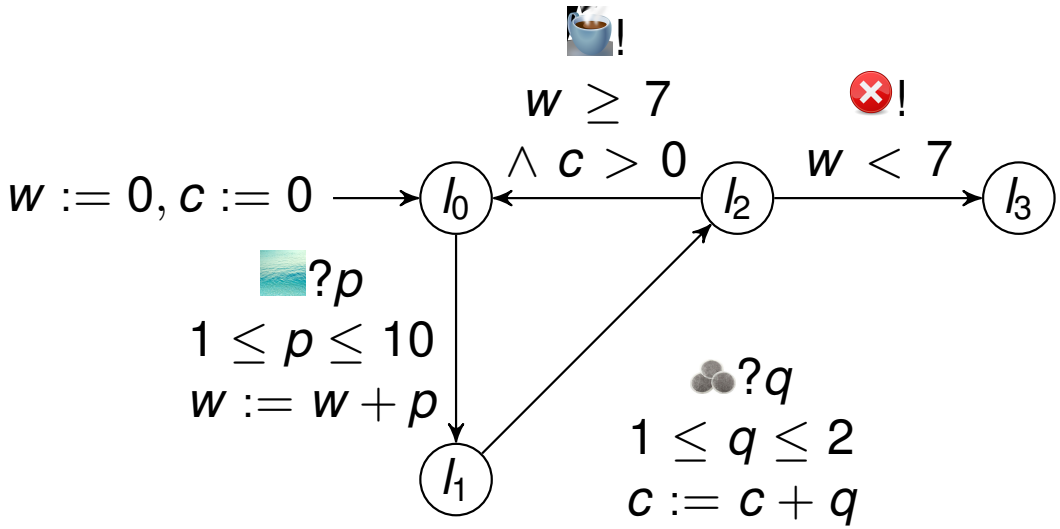
- Test trace: (?, 7) (?, 1) (!,)
- Test execution: (?, 7) (?, 1) (!,)
- Verdict:

Test trace example



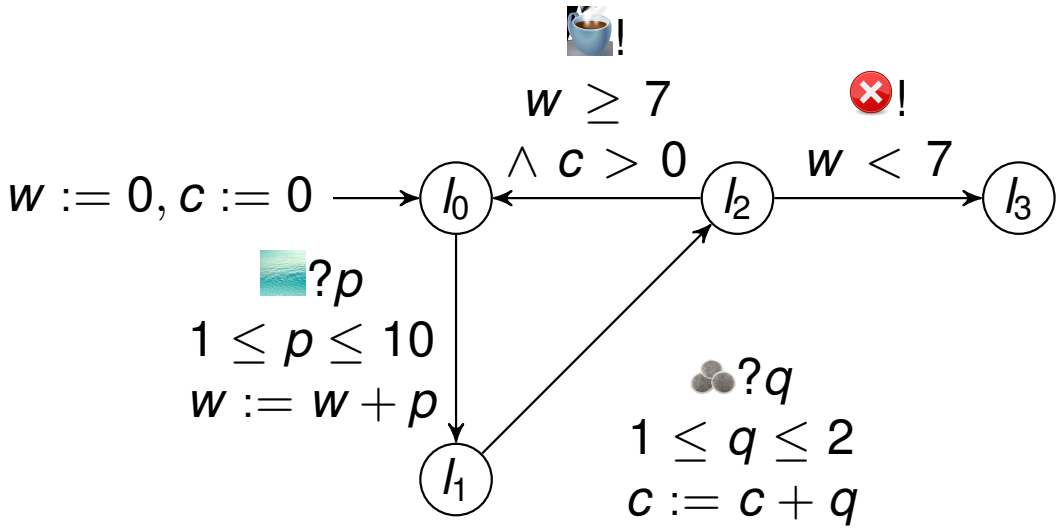
- Test trace: (?, 7) (?, 1) (!,)
- Test execution: (?, 7) (?, 1) (!,)
- Verdict: *Pass*

Test trace example



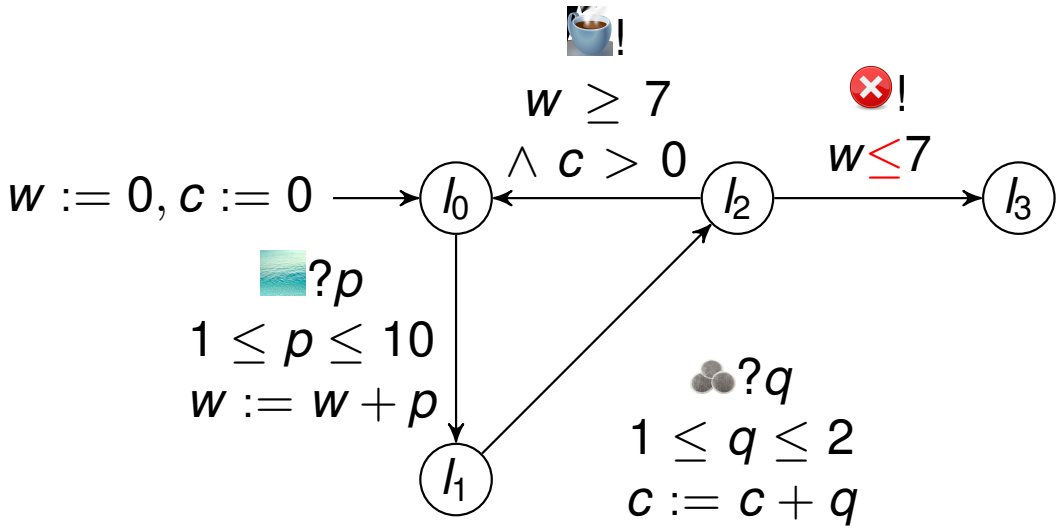
- Test trace: (?, 7) (?, 1) (!,)
- Test execution: (?, 7) (?, 1) (!,)
- Verdict:

Test trace example



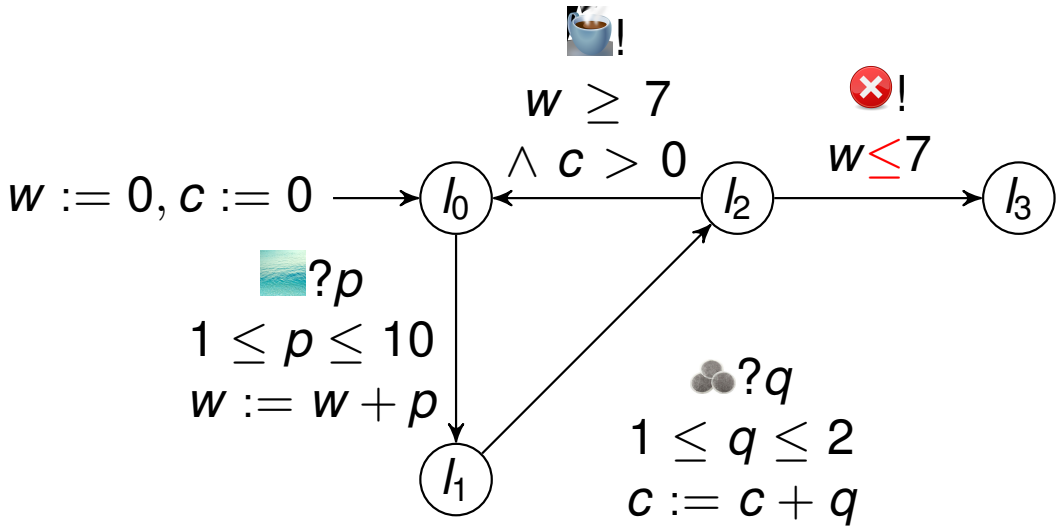
- Test trace: (?, 7) (?, 1) (!,)
- Test execution: (?, 7) (?, 1) (!,)
- Verdict: *Fail*

Test trace example



- Test trace: (?, 7) (?, 1) (!,)
- Test execution: (?, 7) (?, 1) (!,)
- Verdict:

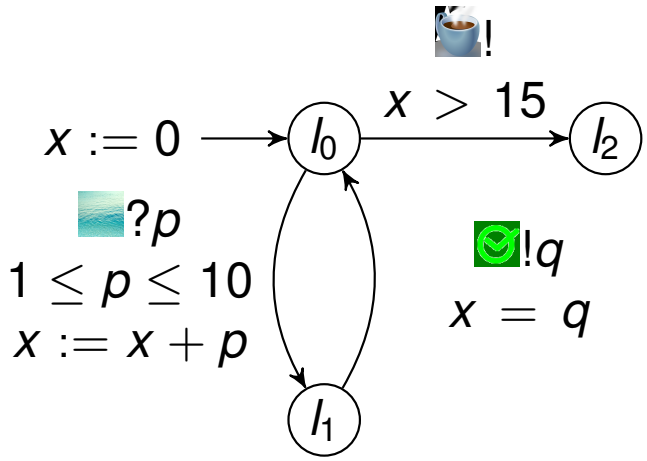
Test trace example



- Test trace: (?, 7) (?, 1) (!,)
- Test execution: (?, 7) (?, 1) (!,)
- Verdict: Inconclusive

Switch coverage test generation and execution

Test generation for STS[1]

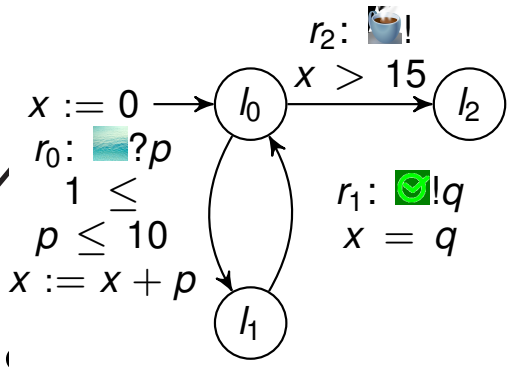


- Symbolic test generation
  - (?, p) (!, q) ...
- On-the-fly parameter value generation during test execution
  - (?, 6) (!, 6) ...

[1] Petra van den Bos, Jan Tretmans. Coverage-Based Testing with Symbolic Transition Systems. https://doi.org/10.1007/978-3-030-31157-5_5

Switch coverage test generation

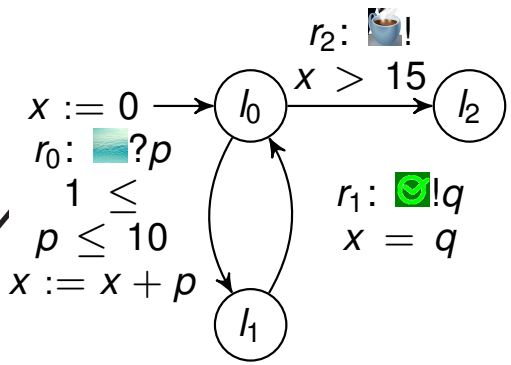
- Obtain symbolic test case using Symbolic Execution



$(l_0, \{x := 0\}, \mathit{True})$

Switch coverage test generation

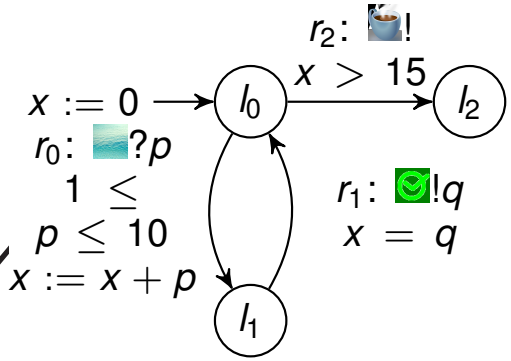
- Obtain symbolic test case using Symbolic Execution



$$(l_0, \{x := 0\}, \mathit{True})$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p\}, 1 \leq p \leq 10 \wedge \mathit{True})$$

Switch coverage test generation

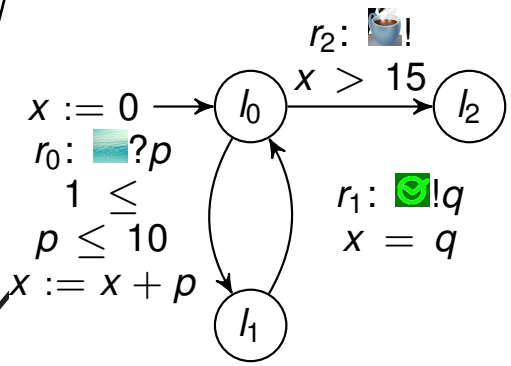
- Obtain symbolic test case using Symbolic Execution



$$(l_0, \{x := 0\}, \mathit{True})$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p\}, 1 \leq p \leq 10 \wedge \mathit{True})$$
$$\downarrow r_1$$
$$(l_0, \{x := 0 + p\}, q = 0 + p \wedge 1 \leq p \leq 10)$$

Switch coverage test generation

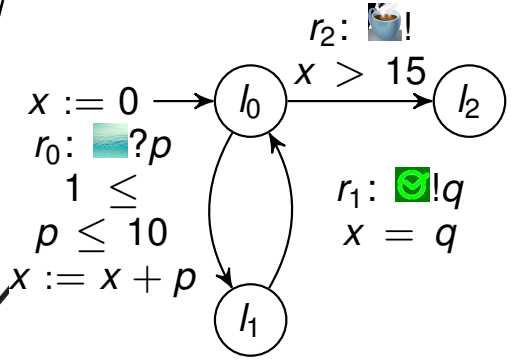
- Obtain symbolic test case using Symbolic Execution



$$(l_0, \{x := 0\}, \mathit{True})$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p\}, 1 \leq p \leq 10 \wedge \mathit{True})$$
$$\downarrow r_1$$
$$(l_0, \{x := 0 + p\}, q = 0 + p \wedge 1 \leq p \leq 10)$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p + p\}, 1 \leq p \leq 10 \wedge q = 0 + p \wedge 1 \leq p \leq 10)$$

Switch coverage test generation

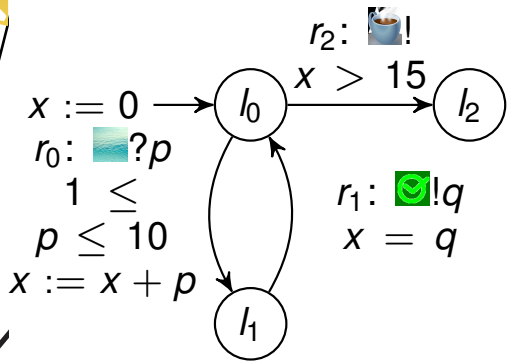
- Obtain symbolic test case using Symbolic Execution



$$(l_0, \{x := 0\}, \mathit{True})$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p\}, 1 \leq p \leq 10 \wedge \mathit{True})$$
$$\downarrow r_1$$
$$(l_0, \{x := 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'' \wedge 1 \leq p'' \leq 10)$$

Switch coverage test generation

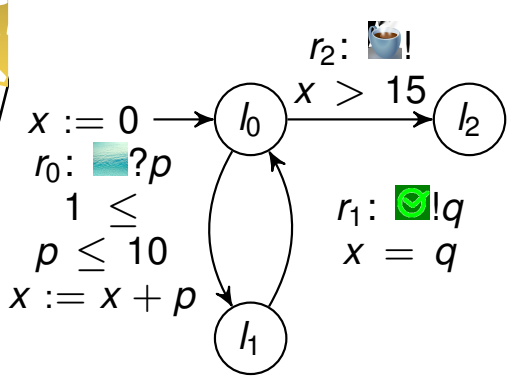
- Obtain symbolic test case using Symbolic Execution



$$(l_0, \{x := 0\}, \mathit{True})$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p\}, 1 \leq p \leq 10 \wedge \mathit{True})$$
$$\downarrow r_1$$
$$(l_0, \{x := 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'' \wedge 1 \leq p'' \leq 10)$$
$$\downarrow r_1$$
$$(l_0, \{x := 0 + p''' + p'\}, q = 0 + p''' + p' \wedge 1 \leq p' \leq 10 \wedge q'' = 0 + p''' \wedge 1 \leq p''' \leq$$

Switch coverage test generation

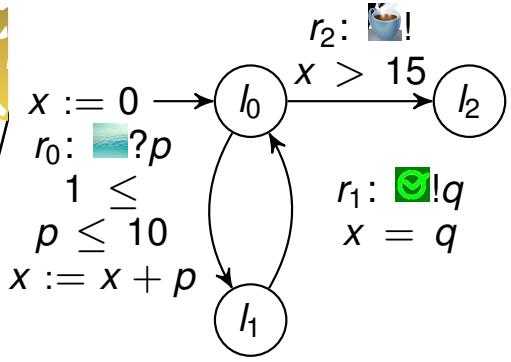
- Obtain symbolic test case using Symbolic Execution



$$(l_0, \{x := 0\}, \mathit{True})$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p\}, 1 \leq p \leq 10 \wedge \mathit{True})$$
$$\downarrow r_1$$
$$(l_0, \{x := 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'' \wedge 1 \leq p'' \leq 10)$$
$$\downarrow r_1$$
$$(l_0, \{x := 0 + p''' + p'\}, q = 0 + p''' + p' \wedge 1 \leq p' \leq 10 \wedge q'' = 0 + p''' \wedge 1 \leq p''' \leq$$
$$\swarrow r_0$$
$$(l_0, \{x := 0 + p'''' + p'' + p\},$$
$$1 \leq p \leq 10 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10)$$

Switch coverage test generation

- Obtain symbolic test case using Symbolic Execution



$$(l_0, \{x := 0\}, \mathit{True})$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p\}, 1 \leq p \leq 10 \wedge \mathit{True})$$
$$\downarrow r_1$$
$$(l_0, \{x := 0 + p'\}, q = 0 + p' \wedge 1 \leq p' \leq 10)$$
$$\downarrow r_0$$
$$(l_1, \{x := 0 + p'' + p\}, 1 \leq p \leq 10 \wedge q' = 0 + p'' \wedge 1 \leq p'' \leq 10)$$
$$\downarrow r_1$$
$$(l_0, \{x := 0 + p''' + p'\}, q = 0 + p''' + p' \wedge 1 \leq p' \leq 10 \wedge q'' = 0 + p''' \wedge 1 \leq p''' \leq$$

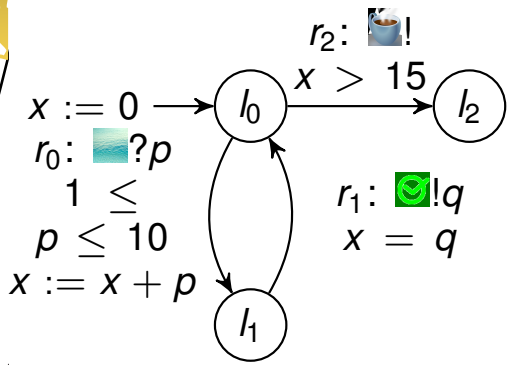
r_0 / r_2

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

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

Switch coverage test generation

- Obtain symbolic test case using Symbolic Execution with 100% switch coverage



$$(l_0, \{x := 0\}, \mathit{True})$$

$$\downarrow r_0$$

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

$$\downarrow r_1$$

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

$$\downarrow r_0$$

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

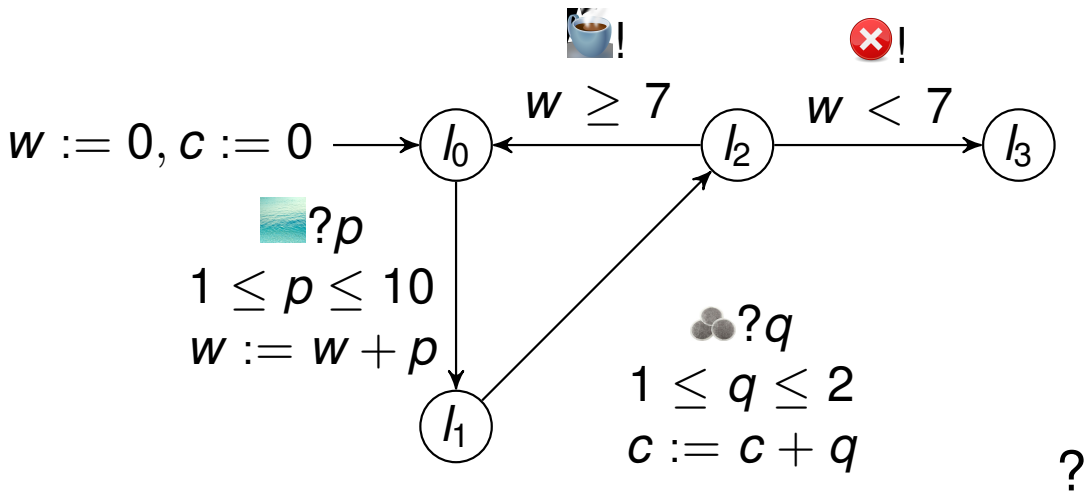
$$(l_0, \{x := 0 + p''' + p'\}, q = 0 + p''' + p' \wedge 1 \leq p' \leq 10 \wedge q'' = 0 + p''' \wedge 1 \leq p''' \leq$$

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

$$\downarrow r_2$$

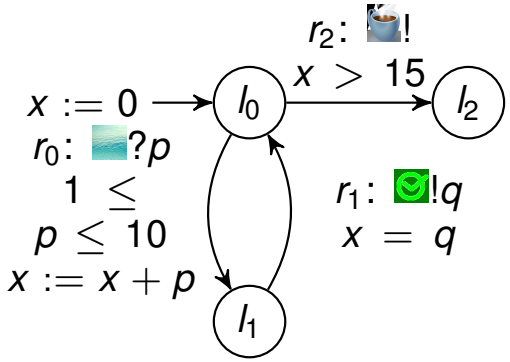
$$(l_2, \{x := 0 + p'''' + p''\},$$
$$0 + p'''' + p'' > 15 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p''''$$

Q: what is the symbolic execution graph of:



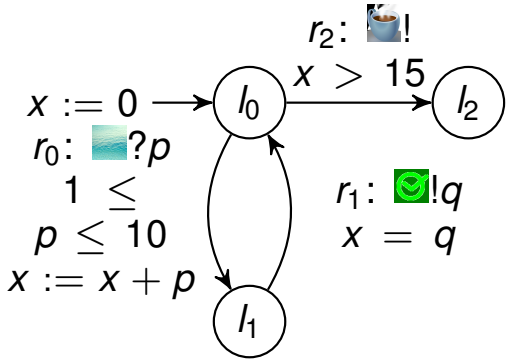
(Answer on board)

Switch coverage test case



- A symbolic test case consists of
  - Sequence of switches: $r_0 r_1 r_0 r_1 r_2$
  - Path condition: $0 + p'''' + p'' > 15 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10$

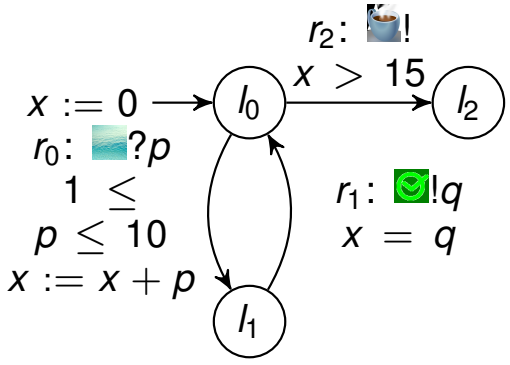
Switch coverage test case



- A symbolic test case consists of
  - Sequence of switches: $r_0 r_1 r_0 r_1 r_2$
  - Path condition: $0 + p'''' + p'' > 15 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10$

- SMT-solver: SAT, $p'''' = 6, q''' = 6, p'' = 10, q' = 16$

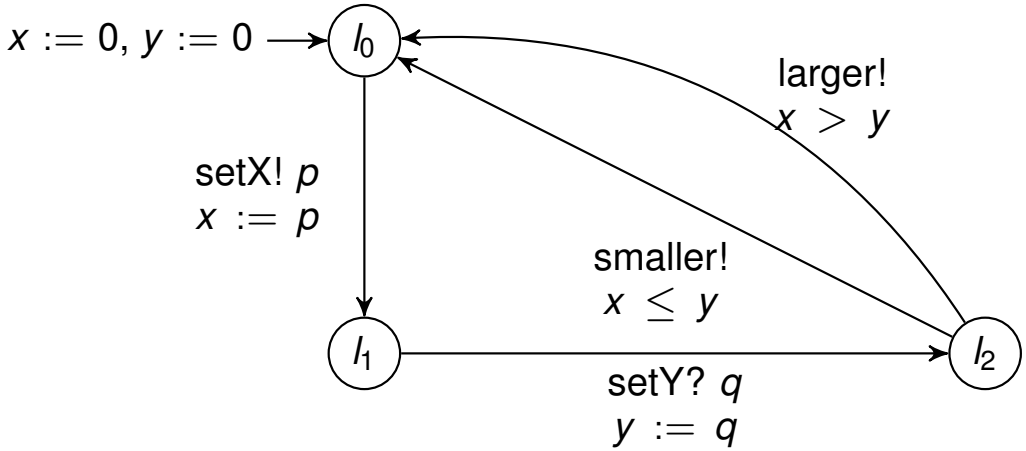
Switch coverage test case



- A symbolic test case consists of
  - Sequence of switches: $r_0 r_1 r_0 r_1 r_2$
  - Path condition: $0 + p'''' + p'' > 15 \wedge q' = 0 + p'''' + p'' \wedge 1 \leq p'' \leq 10 \wedge q''' = 0 + p'''' \wedge 1 \leq p'''' \leq 10$

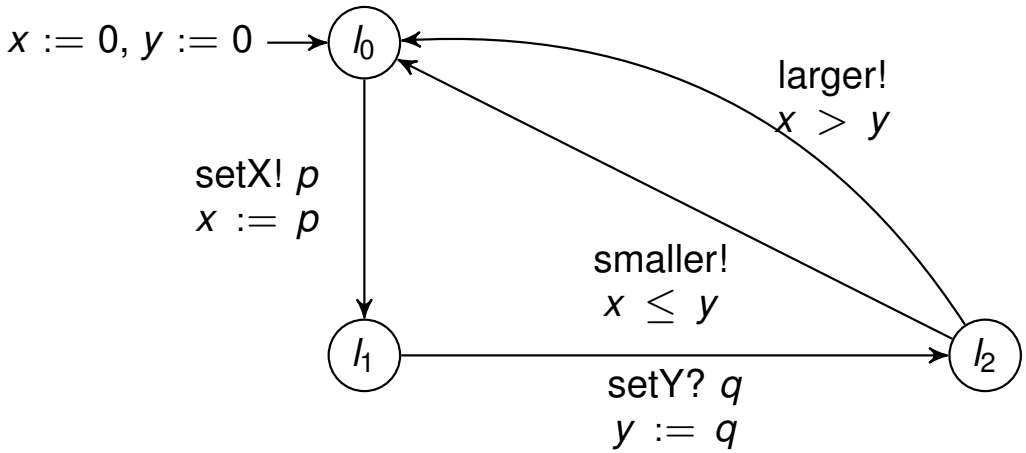
- SMT-solver: SAT, $p'''' = 6, q''' = 6, p'' = 10, q' = 16$
- Test trace: (?,6) (!, 6) (?, 10) (!, 16) (!,)

Generate input values at latest moment (1)



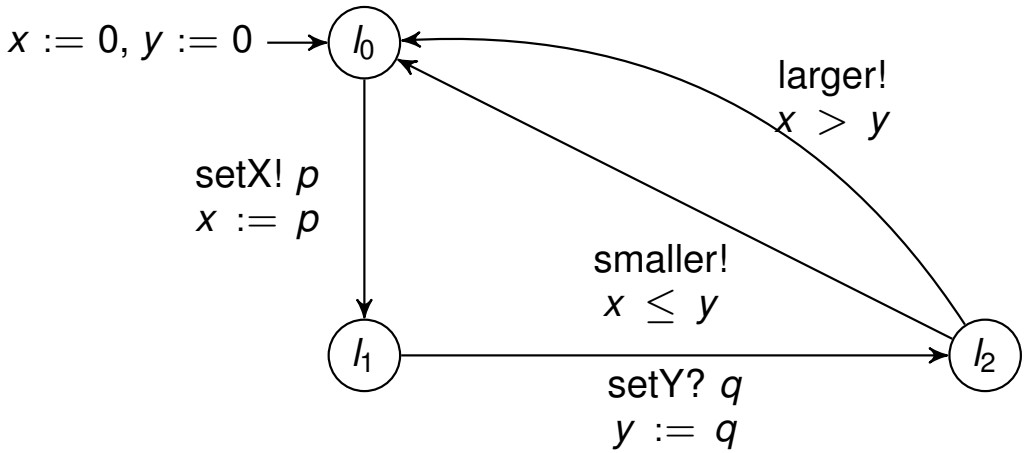
- Goal: cover smaller! switch

Generate input values at latest moment (1)



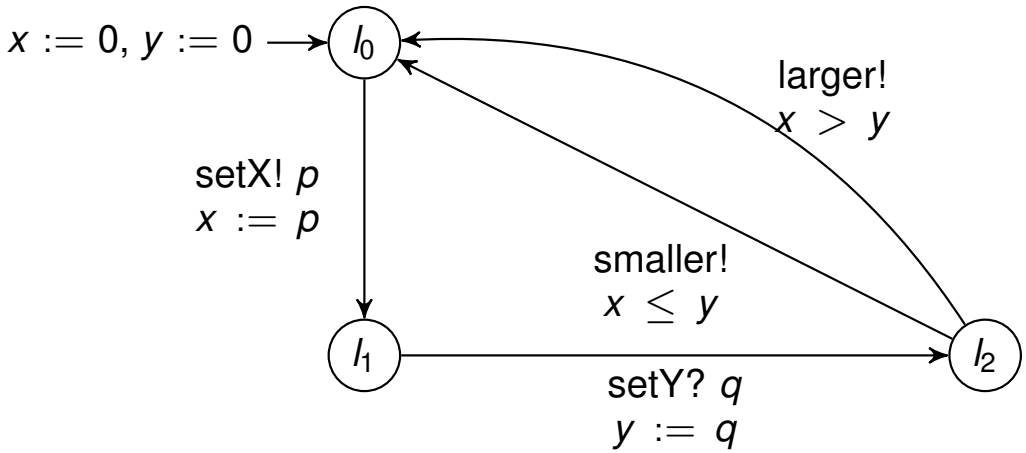
- Goal: cover smaller! switch
- In advance:
  - SMT-Solver returns for $p'' \leq q'$: SAT, $p'' = 0, q' = 0$

Generate input values at latest moment (1)



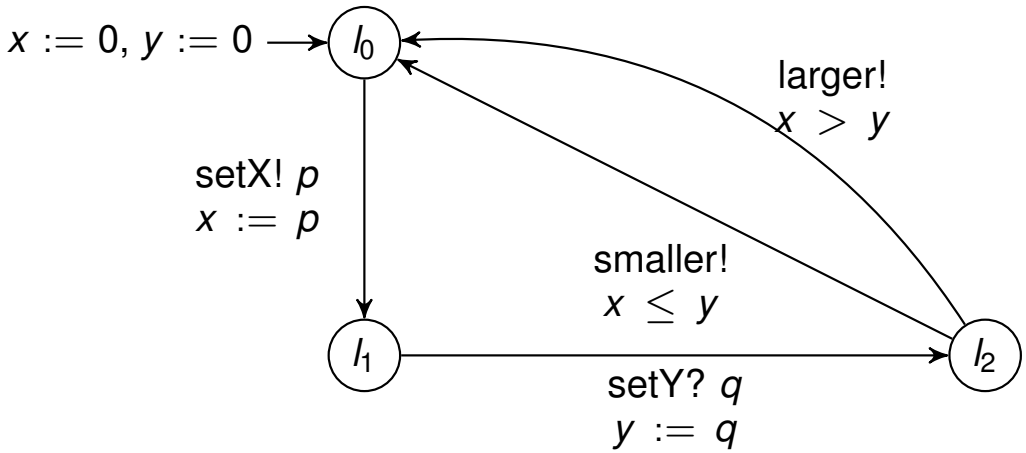
- Goal: cover smaller! switch
- In advance:
  - SMT-Solver returns for $p'' \leq q'$: SAT, $p'' = 0, q' = 0$
  - Test execution: (setX!, 5) (setY?, 0) ends in

Generate input values at latest moment (1)



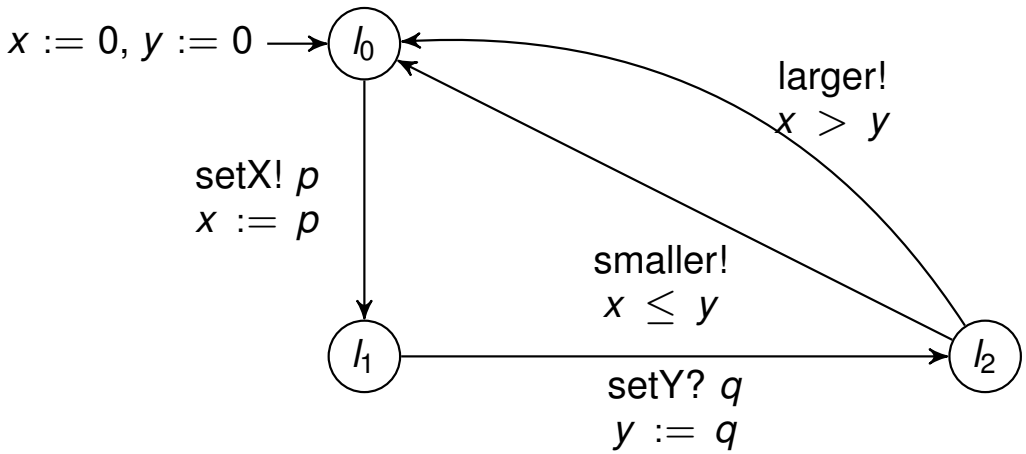
- Goal: cover smaller! switch
- In advance:
  - SMT-Solver returns for $p'' \leq q'$: SAT, $p'' = 0, q' = 0$
  - Test execution: (setX!, 5) (setY?, 0) ends in $(l_0, x := 5, y := 0)$

Generate input values at latest moment (1)



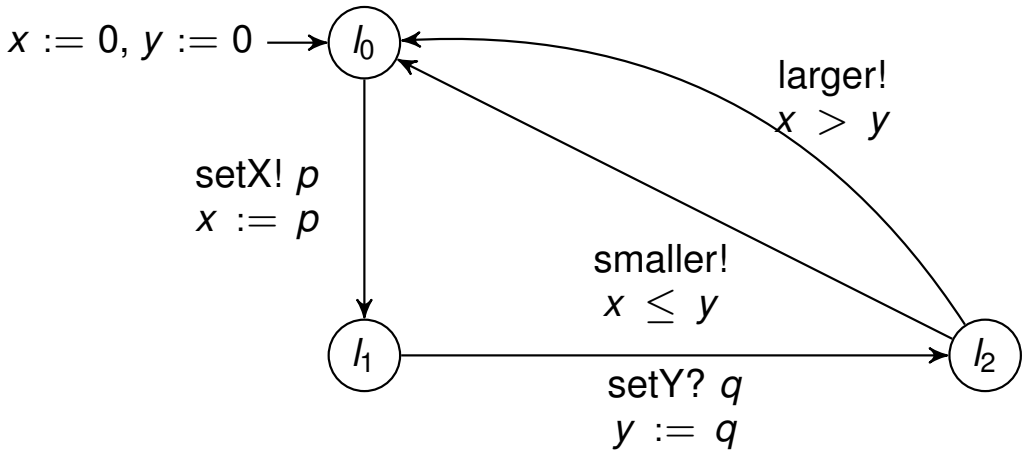
- Goal: cover smaller! switch
- In advance:
  - SMT-Solver returns for $p'' \leq q'$: SAT, $p'' = 0, q' = 0$
  - Test execution: (setX!, 5) (setY?, 0) ends in $(l_0, x := 5, y := 0)$
  - smaller! switch **not** covered

Generate input values at latest moment (1)



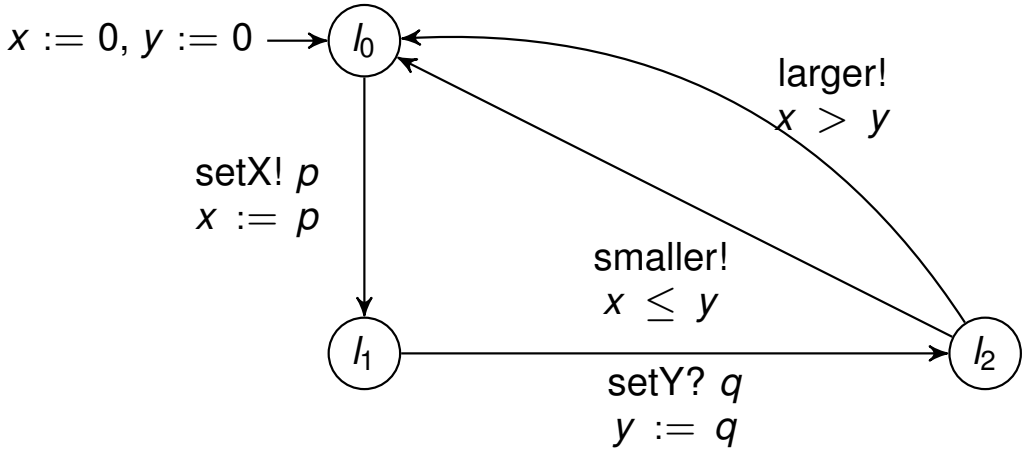
- Goal: cover smaller! switch
- In advance:
  - SMT-Solver returns for $p'' \leq q'$: SAT, $p'' = 0, q' = 0$
  - Test execution: (setX!, 5) (setY?, 0) ends in $(l_0, x := 5, y := 0)$
  - smaller! switch **not** covered
- Latest moment:
  - Test execution: (setX!, 5) ends in $(l_1, x := 5, y := 0)$

Generate input values at latest moment (1)



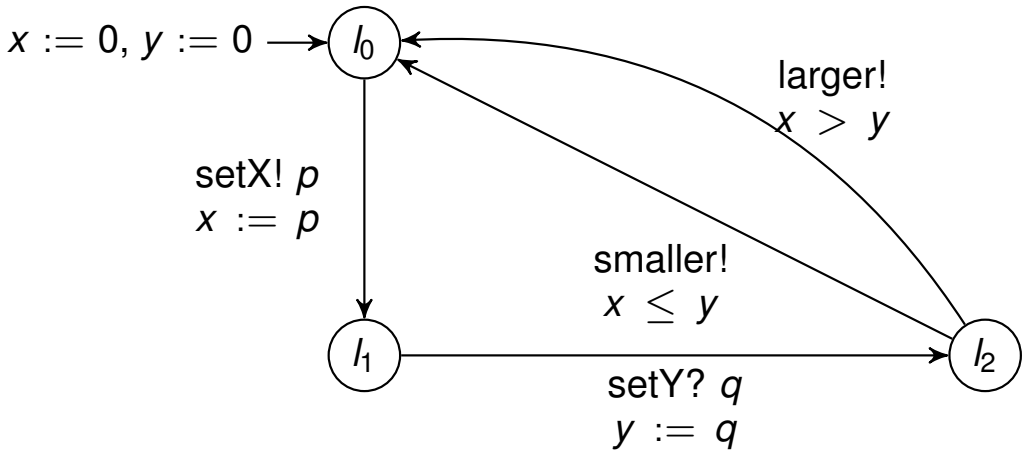
- Goal: cover smaller! switch
- In advance:
  - SMT-Solver returns for $p'' \leq q'$: SAT, $p'' = 0, q' = 0$
  - Test execution: (setX!, 5) (setY?, 0) ends in $(l_0, x := 5, y := 0)$
  - smaller! switch **not** covered
- Latest moment:
  - Test execution: (setX!, 5) ends in $(l_1, x := 5, y := 0)$
  - SMT-Solver returns for $5 \leq q'$: SAT, $q' = 5$

Generate input values at latest moment (1)



- Goal: cover smaller! switch
- In advance:
  - SMT-Solver returns for $p'' \leq q'$: SAT, $p'' = 0, q' = 0$
  - Test execution: (setX!, 5) (setY?, 0) ends in $(l_0, x := 5, y := 0)$
  - smaller! switch **not** covered
- Latest moment:
  - Test execution: (setX!, 5) ends in $(l_1, x := 5, y := 0)$
  - SMT-Solver returns for $5 \leq q'$: SAT, $q' = 5$
  - Test execution continues: (setY?, 5) ends in

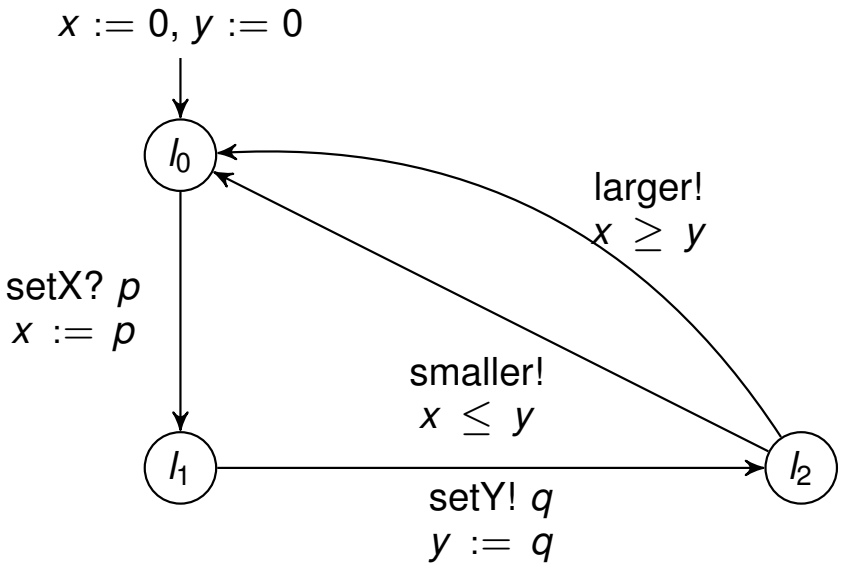
Generate input values at latest moment (1)



- Goal: cover smaller! switch
- In advance:
  - SMT-Solver returns for $p'' \leq q'$: SAT, $p'' = 0, q' = 0$
  - Test execution: (setX!, 5) (setY?, 0) ends in $(l_0, x := 5, y := 0)$
  - smaller! switch **not** covered
- Latest moment:
  - Test execution: (setX!, 5) ends in $(l_1, x := 5, y := 0)$
  - SMT-Solver returns for $5 \leq q'$: SAT, $q' = 5$
  - Test execution continues: (setY?, 5) ends in $(l_0, x := 5, y := 5)$
  - smaller! switch covered!

Generate input values at latest moment (2)

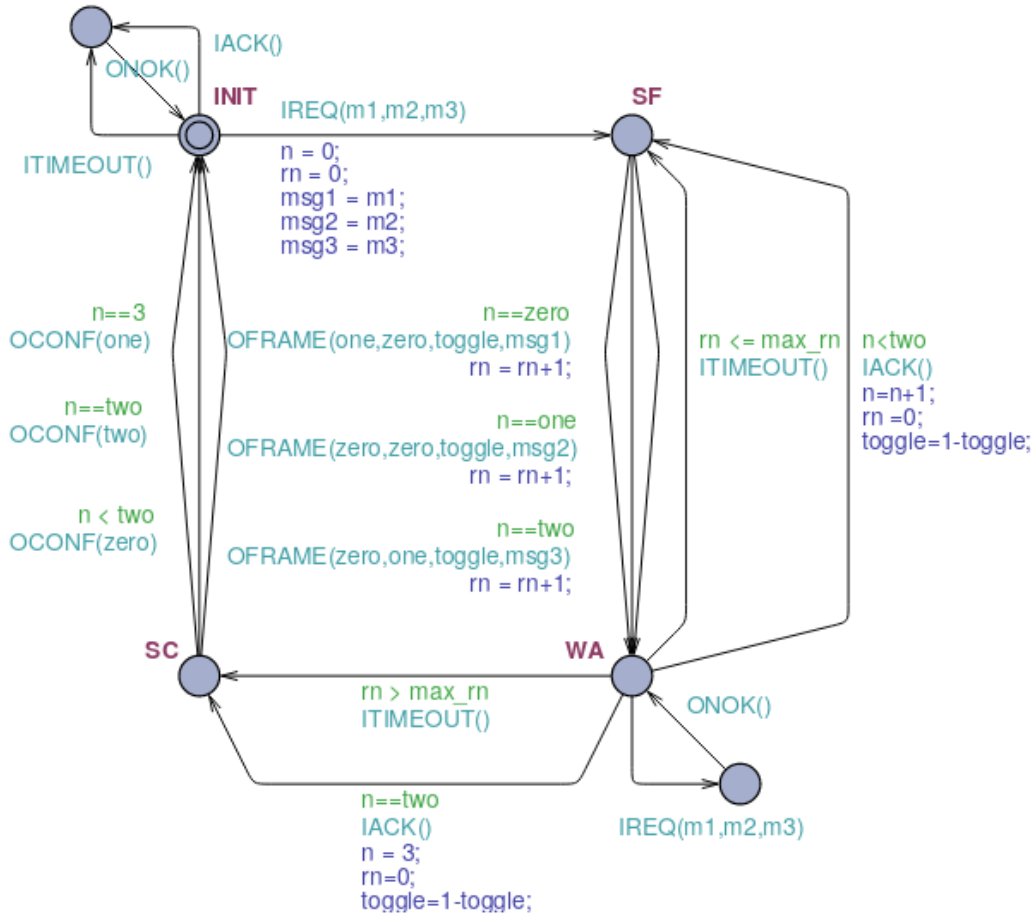
- Late parameter value choice
  - no guaranteed coverage for some models



Case study: Bounded Retransmission Protocol

Case study: Bounded Retransmission Protocol

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



Experimental setup

- Test mutant implementations against given specification
- Compare:
  - Switch coverage testing
  - Random testing by TorXakis
- Count nr of switches before obtaining Fail

Experimental setup

- Test mutant implementations against given specification
- Compare:
  - Switch coverage testing
  - 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

Experimental setup

- Test mutant implementations against given specification
- Compare:
  - Switch coverage testing
  - 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	
Average	42.7	328.7	7.7 times more!
Geometric mean	21.5	64.9	3.0 times more!

Master thesis?

- Example topics:
  - Modelling
  - Applying model-based testing
  - Test generation algorithms
  - Behaviour-driven development
  - Testing in software engineering
- Interested? Send me an email!

Next times

- Next week:
  - Monday: lecture by Axini
  - Wednesday: tutorial on STS, deadline homework 6
  - Thursday: online Q&A at 16:00-17:00
- Week after next week:
  - Monday: online Q&A at 9:30-10:30, deadline SmartDoor assignment
  - Tuesday: lecture by Axini
  - Thursday: tutorial on project
- Recommendation:
  - Work on AMP-part of project from March 24
  - Finish other project parts before March 24