UNIVERSITY OF TWENTE.

Model-Based Testing with Labeled Transition Systems

Software Testing and Risk Assessment

Lecture 4b

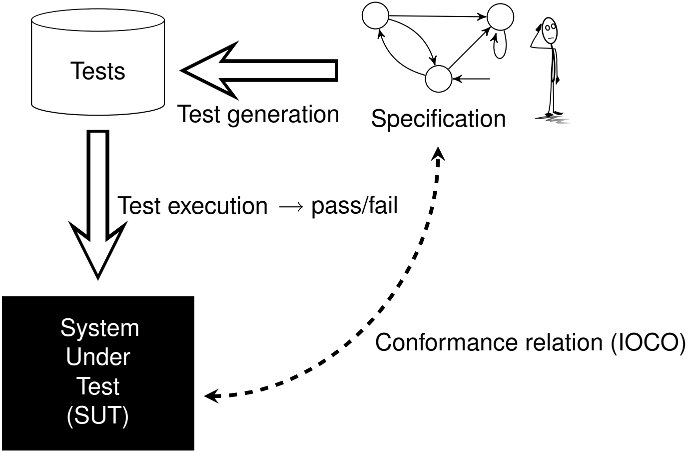
Today

- Project planning
- Model-based testing
- Labeled Transitions Systems
- Quiescence
- Nondeterministic LTS
- Formal definitions
- Determinisation
- Test cases
- Distinguishing and homing test cases

Plan tasks for Software Testing Project

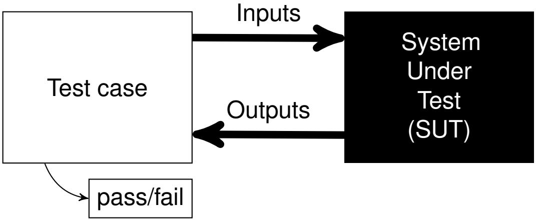
- Last homework on March 26
- SmartDoor assignment on March 31
- Axini lectures on March 24 and April 1
- Project deadline on April 11

Model-Based Testing



Model-based testing in this course

- What kind of testing?
  - Functionality: functional behaviour and interaction of system
  - On system-level
  - Black-box: use only inputs and outputs
  - Test execution:



Model-based testing in this course

- What kind of systems?
  - Systems with a software interface
  - Discrete event-driven systems
  - Reactive, dynamic systems
  - Data-intensive systems

Model-based testing in this course

- What kind of systems?
  - Systems with a software interface
  - Discrete event-driven systems
  - Reactive, dynamic systems
  - Data-intensive systems
- Real-world systems:

ASML wafer machine

Canon industrial printer

Model-based testing in this course

- What kind of model?
  - Lecture 4b & 5: Labeled Transition Systems (LTS)
  - Lecture 6: Symbolic Transition System (STS)
  - Lectures 7: AML model (Axini)

Model-based testing in this course

- What kind of model?
  - Lecture 4b & 5: Labeled Transition Systems (LTS)
  - Lecture 6: Symbolic Transition System (STS)
  - Lectures 7: AML model (Axini)
- Models for testing $\neq$ models for constructing software (e.g. MBSE)
  - What vs. How

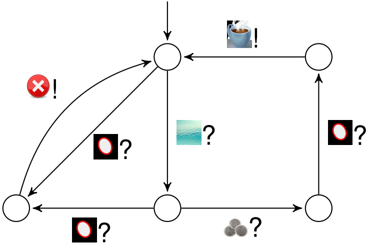
Model-based testing in this course

- What kind of model?
  - Lecture 4b & 5: Labeled Transition Systems (LTS)
  - Lecture 6: Symbolic Transition System (STS)
  - Lectures 7: AML model (Axini)
- Models for testing $\neq$ models for constructing software (e.g. MBSE)
  - What vs. How
- Test models provide abstraction
  - Smaller size $\Longrightarrow$ easier to maintain
  - Same size $\Longrightarrow$ Test larger systems

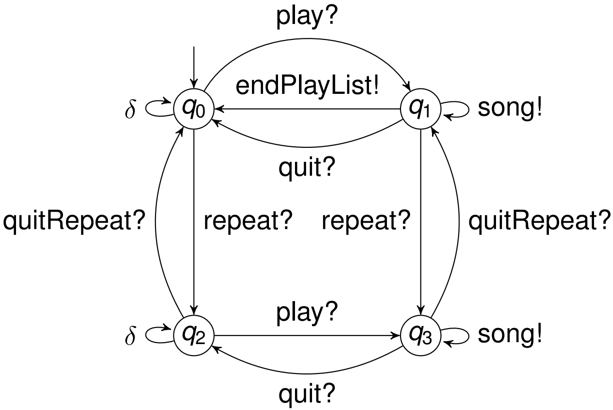
Model-based testing in this course

- What kind of model?
  - Lecture 4b & 5: Labeled Transition Systems (LTS)
  - Lecture 6: Symbolic Transition System (STS)
  - Lectures 7: AML model (Axini)
- Models for testing $\neq$ models for constructing software (e.g. MBSE)
  - What vs. How
- Test models provide abstraction
  - Smaller size $\implies$ easier to maintain
  - Same size $\implies$ Test larger systems
- Test models may be partial
  - Only model relevant/most important part

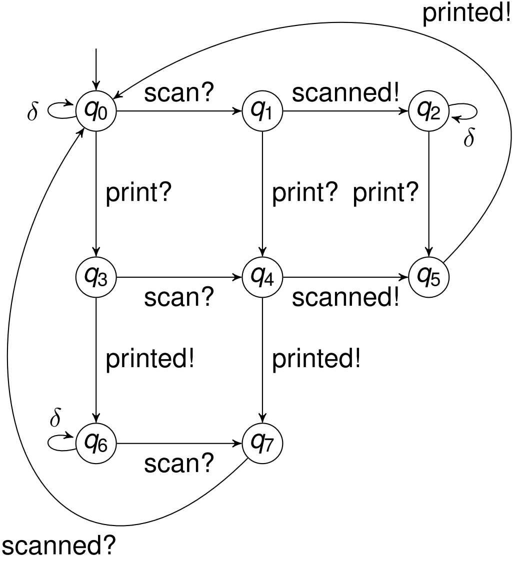
Example LTS: Coffee machine



Example LTS: Music App

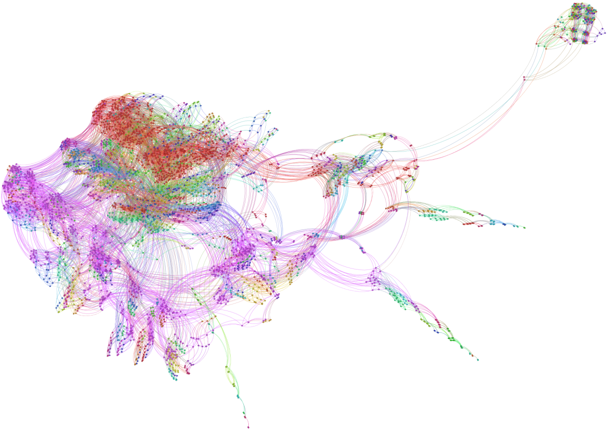


Example LTS: printer/scanner



Size of a model of a real-world system[1]

- A model of a Canon printer



[1] Smeenk, Wouter. "Applying automata learning to complex industrial software." Master's thesis, Radboud University Nijmegen (2012).

Literature

- Model Based Testing with Labelled Transition Systems, Jan Tretmans, 2008.
  `https://doi.org/10.1007/978-3-540-78917-8_1`
  (Main literature used for LTS in this course)
- Tester versus Bug: A Generic Framework for Model-Based Testing via Games, Petra van den Bos and Marielle Stoelinga, 2018. Ignore parts about games, only read: Sections 2, 2.1 and 2.3. `https://doi.org/10.48550/arXiv.1809.03098`
  (This paper only considers deterministic LTS, such that definitions are slightly simpler than in this course.)
- Model-Based Testing. Mark Timmer, Ed Brinksma, Mariëlle Stoelinga (2011).
  `https://doi.org/10.3233/978-1-60750-711-6-1` (Gentle introduction to MBT with slightly different notations/naming)

Labeled Transition Systems

- Labeled Transition System: (Q, L, T, q_0)
  - Q: set of states
  - L: set of labels (for actions)
  - $T \subseteq Q \times L \times Q$: transition relation
  - q_0: initial state

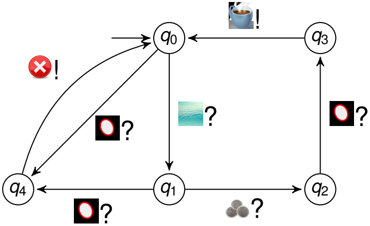
Labeled Transition Systems

- Labeled Transition System: (Q, L, T, q_0)
  - Q: set of states
  - L: set of labels (for actions)
  - $T \subseteq Q \times L \times Q$: transition relation
  - q_0: initial state
- Labeled Transition System with inputs, outputs, and quiescence: (Q, L_I, L_U, T, q_0)
  - Q: set of states
  - L_I: set of input labels (for input actions, e.g. $a?$, $b?$, $c?$)
  - L_U: set of output labels (for output actions, e.g. $x!$, $y!$, $z!$)
  - Special label for quiescence: $\delta \notin L_I \cup L_O$
  - $T \subseteq Q \times (L_I \cup L_O \cup \{\delta\}) \times Q$: transition relation
  - q_0: initial state

Labeled Transition Systems

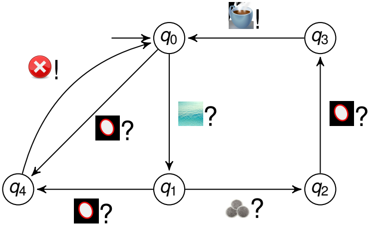
- Labeled Transition System: (Q, L, T, q_0)
  - Q: set of states
  - L: set of labels (for actions)
  - $T \subseteq Q \times L \times Q$: transition relation
  - q_0: initial state
- Labeled Transition System with inputs, outputs, and quiescence: (Q, L_I, L_U, T, q_0)
  - Q: set of states
  - L_I: set of input labels (for input actions, e.g. $a?$, $b?$, $c?$)
  - L_U: set of output labels (for output actions, e.g. $x!$, $y!$, $z!$)
  - Special label for quiescence: $\delta \notin L_I \cup L_O$
  - $T \subseteq Q \times (L_I \cup L_O \cup \{\delta\}) \times Q$: transition relation
  - q_0: initial state
- Note: τ out of scope for this lecture

Example coffee machine



- CM-LTS is the tuple:

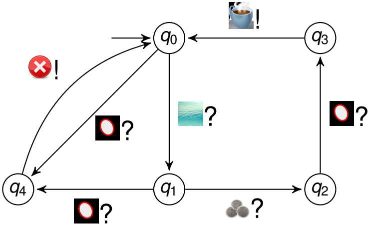
Example coffee machine



- CM-LTS is the tuple:

  $(\{q_0, q_1, q_2, q_3, q_4\},$
  $\{$$?,$$?,$$?\},$
  $\{$$!,$$!\},$

Example coffee machine



- CM-LTS is the tuple:

$(\{q_0, q_1, q_2, q_3, q_4\},$
{?,?,?},
{!,!},
$\{(q_0,$$?,q_1),(q_1,$$?,q_4),(q_4,$$!,q_1), \ldots\},$
$q_0)$

Quiescence

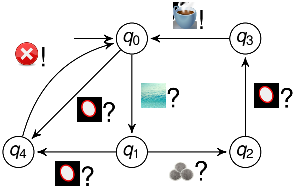
- Quiescence = absence of output
- SUT will not any produce output (until next input arrives)

Quiescence

- Quiescence = absence of output
- SUT will not any produce output (until next input arrives)
  - no matter how long tester waits
  - in practice: set timeout

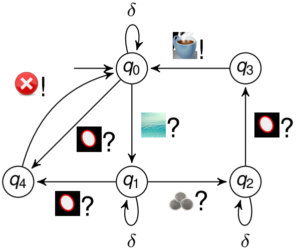
Quiescence

- Quiescence = absence of output
- SUT will not any produce output (until next input arrives)
  - no matter how long tester waits
  - in practice: set timeout
- Q: which states are quiescent?



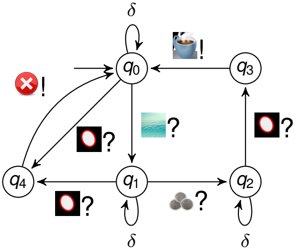
Quiescence

- Quiescence = absence of output
- SUT will not any produce output (until next input arrives)
  - no matter how long tester waits
  - in practice: set timeout
- Q: which states are quiescent?



Quiescence

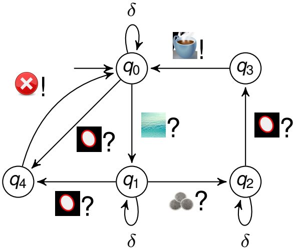
- Quiescence = absence of output
- SUT will not any produce output (until next input arrives)
  - no matter how long tester waits
  - in practice: set timeout
- Q: which states are quiescent?



- Some traces with quiescence (elements of *Straces*(CM-LTS)):

Quiescence

- Quiescence = absence of output
- SUT will not any produce output (until next input arrives)
  - no matter how long tester waits
  - in practice: set timeout
- Q: which states are quiescent?



- Some traces with quiescence (elements of *Straces*(CM-LTS)):
  - ? δ ? ?
  - δ ? δ
  - ? ? δ

Quiescence

- Quiescence = absence of output
- SUT will not any produce output (until next input arrives)

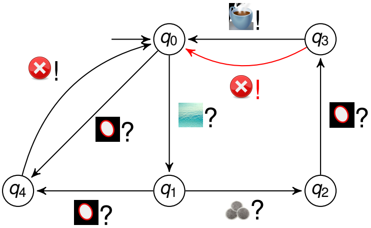
- Formal definition of quiescence:
  - Let $S = (Q, L_I, L_U, T, q_0)$ be an LTS, and $q \in Q$.

Quiescence

- Quiescence = absence of output
- SUT will not any produce output (until next input arrives)

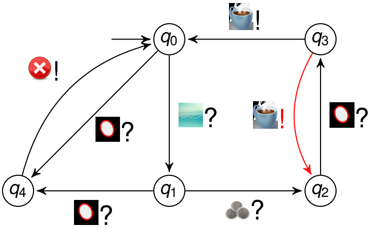
- Formal definition of quiescence:
  - Let $S = (Q, L_I, L_U, T, q_0)$ be an LTS, and $q \in Q$.
  - Then q is quiescent, denoted $\delta(q)$ if:
    $\forall x \in L_U, \forall q' \in Q : (q, x, q') \notin T$

Nondeterministic LTSs

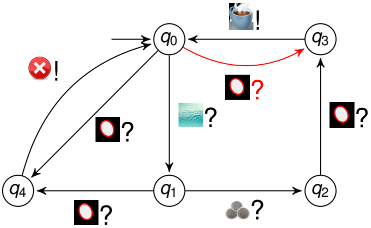


Deterministic, but no control over output

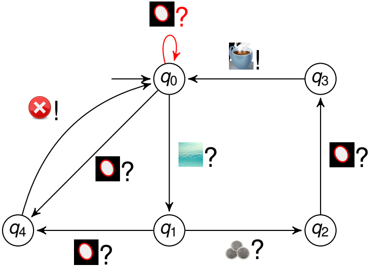
Nondeterministic LTSs



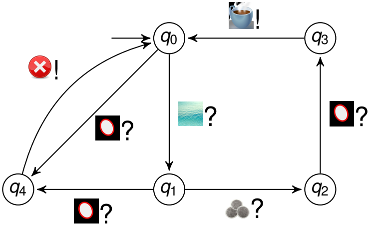
Nondeterministic LTSs



Nondeterministic LTSs

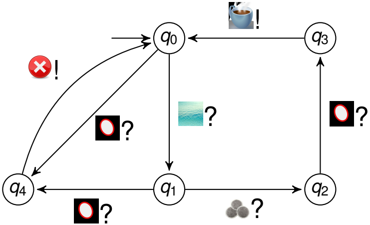


After



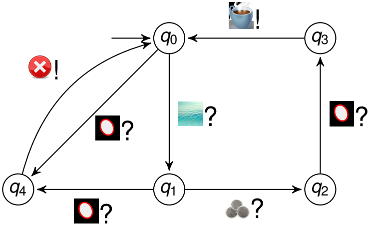
- Reached state after a trace:
  - q_4 *after* ! =

After



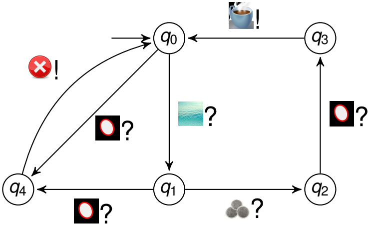
- Reached state after a trace:
  - q_4 *after* ! = $\{q_0\}$

After



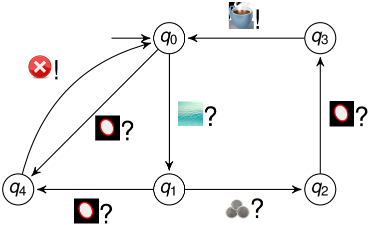
- Reached state after a trace:
  - q_4 *after* ! $= \{q_0\}$
  - q_2 *after* ? $=$

After



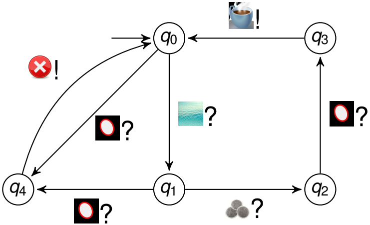
- Reached state after a trace:
  - q_4 *after* ! = $\{q_0\}$
  - q_2 *after* ? = $\{q_3\}$

After



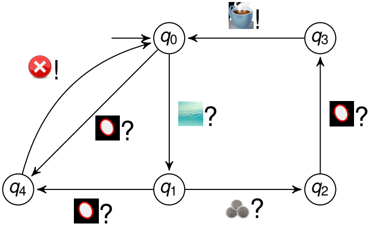
- Reached state after a trace:
  - q_4 *after* ! = $\{q_0\}$
  - q_2 *after* ? = $\{q_3\}$
  - q_0 *after* (? ? ? !) =

After



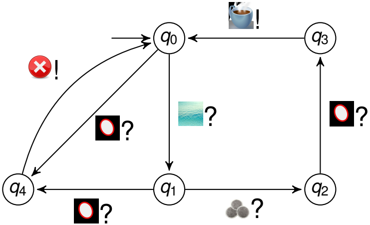
- Reached state after a trace:
  - q_4 *after* ! = $\{q_0\}$
  - q_2 *after* ? = $\{q_3\}$
  - q_0 *after* (? ? ? !) = $\{q_0\}$

After



- Reached state after a trace:
  - q_4 *after* ! = $\{q_0\}$
  - q_2 *after* ? = $\{q_3\}$
  - q_0 *after* (? ? ? !) = $\{q_0\}$
  - q_0 *after* (? δ ? ? !) =

After



- Reached state after a trace:
  - q_4 *after* ! = $\{q_0\}$
  - q_2 *after* ? = $\{q_3\}$
  - q_0 *after* (? ? ? !) = $\{q_0\}$
  - q_0 *after* (? δ ? ? !) = $\{q_0\}$

Formal definition for *after* and *Straces*

- Let $S = (Q, L_I, L_U, T, q_0)$ be an LTS.

Formal definition for *after* and *Straces*

- Let $S = (Q, L_I, L_U, T, q_0)$ be an LTS.
- Let $L = L_I \cup L_U \cup \{\delta\}$.

Formal definition for *after* and *Straces*

- Let $S = (Q, L_I, L_U, T, q_0)$ be an LTS.
- Let $L = L_I \cup L_U \cup \{\delta\}$.
- Let $q \in Q$ be a state, $\mu \in L$ a label, and $\rho \in L^*$ a sequence of labels.

Formal definition for *after* and *Straces*

- Let $S = (Q, L_I, L_U, T, q_0)$ be an LTS.
- Let $L = L_I \cup L_U \cup \{\delta\}$.
- Let $q \in Q$ be a state, $\mu \in L$ a label, and $\rho \in L^*$ a sequence of labels.
- Remember: A state q is quiescent, denoted $\delta(q)$, if: $\forall x \in L_U, \forall q' \in Q : (q, x, q') \notin T$

Formal definition for *after* and *Straces*

- Let $S = (Q, L_I, L_U, T, q_0)$ be an LTS.
- Let $L = L_I \cup L_U \cup \{\delta\}$.
- Let $q \in Q$ be a state, $\mu \in L$ a label, and $\rho \in L^*$ a sequence of labels.
- Remember: A state q is quiescent, denoted $\delta(q)$, if: $\forall x \in L_U, \forall q' \in Q : (q, x, q') \notin T$
- *after* : $Q \times L^* \to \mathbb{P}(Q)$

Formal definition for *after* and *Straces*

- Let $S = (Q, L_I, L_U, T, q_0)$ be an LTS.
- Let $L = L_I \cup L_U \cup \{\delta\}$.
- Let $q \in Q$ be a state, $\mu \in L$ a label, and $\rho \in L^*$ a sequence of labels.
- Remember: A state q is quiescent, denoted $\delta(q)$, if: $\forall x \in L_U, \forall q' \in Q : (q, x, q') \notin T$
- $\mathit{after} : Q \times L^* \to \mathbb{P}(Q)$
- $q \ \mathit{after} \ \epsilon = \{q\}$

Formal definition for *after* and *Straces*

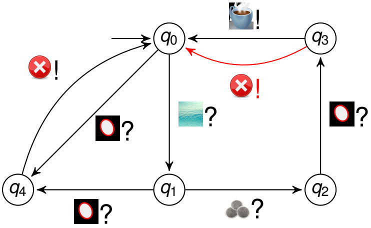
- Let $S = (Q, L_I, L_U, T, q_0)$ be an LTS.
- Let $L = L_I \cup L_U \cup \{\delta\}$.
- Let $q \in Q$ be a state, $\mu \in L$ a label, and $\rho \in L^*$ a sequence of labels.
- Remember: A state q is quiescent, denoted $\delta(q)$, if: $\forall x \in L_U, \forall q' \in Q : (q, x, q') \notin T$
- $\mathit{after} : Q \times L^* \to \mathbb{P}(Q)$
- $q \; \mathit{after} \; \epsilon = \{q\}$
- $q \; \mathit{after} \; \mu\rho = Q' \; \mathit{after} \; \rho$ where $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$
- $\mathit{Straces} : Q \to \mathbb{P}(L^*)$

Formal definition for *after* and *Straces*

- Let $S = (Q, L_I, L_U, T, q_0)$ be an LTS.
- Let $L = L_I \cup L_U \cup \{\delta\}$.
- Let $q \in Q$ be a state, $\mu \in L$ a label, and $\rho \in L^*$ a sequence of labels.
- Remember: A state q is quiescent, denoted $\delta(q)$, if: $\forall x \in L_U, \forall q' \in Q : (q, x, q') \notin T$
- $\mathit{after} : Q \times L^* \to \mathbb{P}(Q)$
- $q \ \mathit{after} \ \epsilon = \{q\}$
- $q \ \mathit{after} \ \mu\rho = Q' \ \mathit{after} \ \rho$ where $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$
- $\mathit{Straces} : Q \to \mathbb{P}(L^*)$
- $\mathit{Straces}(q) = \{\rho' \in L^* \mid q \ \mathit{after} \ \rho' \neq \emptyset\}$

Examples for *after*

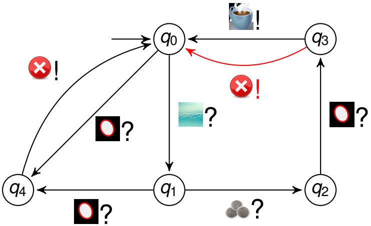
- *after* : $Q \times L^* \to \mathbb{P}(Q)$
- q *after* $\epsilon = \{q\}$
- q *after* $\mu\rho = Q'$ *after* ρ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$
- q_3 *after* ! =



Examples for *after*

- *after* : $Q \times L^* \rightarrow \mathbb{P}(Q)$
- q *after* $\epsilon = \{q\}$
- q *after* $\mu\rho = Q'$ *after* ρ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$

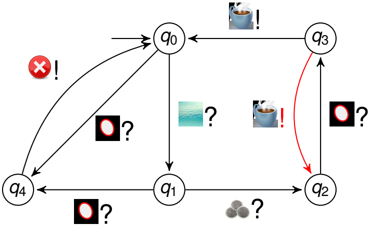
- q_3 *after* ! = $\{q_0\}$



Examples for *after*

- *after* : $Q \times L^* \to \mathbb{P}(Q)$
- q *after* $\epsilon = \{q\}$
- q *after* $\mu\rho = Q'$ *after* ρ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$

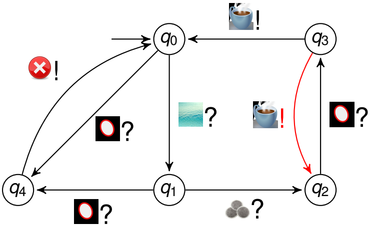
- q_3 *after* ! =



Examples for *after*

- *after* : $Q \times L^* \to \mathbb{P}(Q)$
- q *after* $\epsilon = \{q\}$
- q *after* $\mu\rho = Q'$ *after* ρ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$

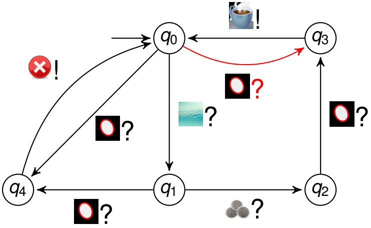
- q_3 *after* ! = $\{q_0, q_2\}$



Examples for *after*

- $after : Q \times L^* \rightarrow \mathbb{P}(Q)$
- q *after* $\epsilon = \{q\}$
- q *after* $\mu\rho = Q'$ *after* ρ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$

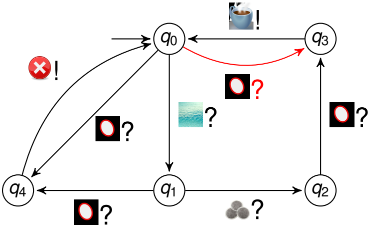
- q_0 *after* ? =



Examples for *after*

- *after* : $Q \times L^* \to \mathbb{P}(Q)$
- q *after* $\epsilon = \{q\}$
- q *after* $\mu\rho = Q'$ *after* ρ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$

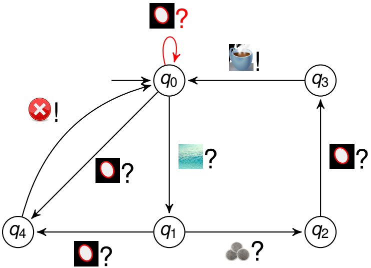
- q_0 *after* ? = $\{q_3, q_4\}$



Examples for *after*

- $after : Q \times L^* \to \mathbb{P}(Q)$
- $q\ after\ \epsilon = \{q\}$
- $q\ after\ \mu\rho = Q'\ after\ \rho$ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$

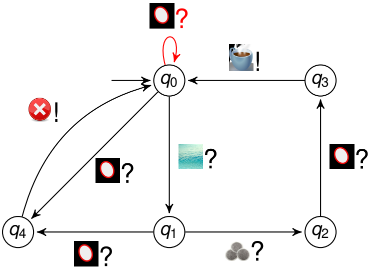
- $q_0\ after\ \delta$ =



Examples for *after*

- *after* : $Q \times L^* \rightarrow \mathbb{P}(Q)$
- q *after* $\epsilon = \{q\}$
- q *after* $\mu\rho = Q'$ *after* ρ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$

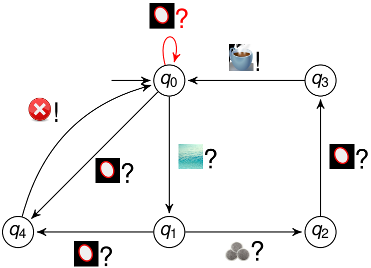
- q_0 *after* δ = $\{q_0\}$



Examples for *after*

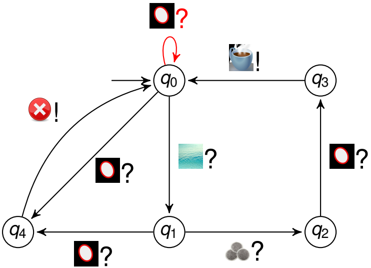
- *after* : $Q \times L^* \to \mathbb{P}(Q)$
- q *after* $\epsilon = \{q\}$
- q *after* $\mu\rho = Q'$ *after* ρ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$

- q_3 *after* δ =



Examples for *after*

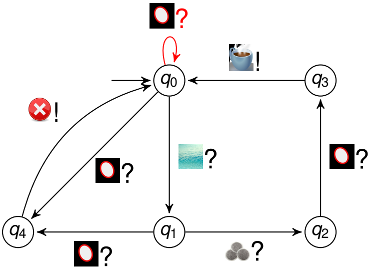
- *after* : $Q \times L^* \rightarrow \mathbb{P}(Q)$
- q *after* $\epsilon = \{q\}$
- q *after* $\mu\rho = Q'$ *after* ρ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$
- q_3 *after* δ = $\emptyset$



Examples for *after*

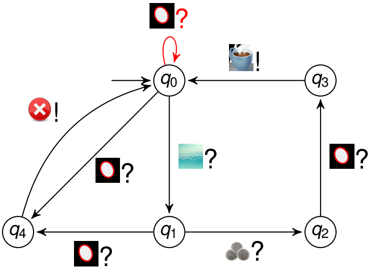
- *after* : $Q \times L^* \rightarrow \mathbb{P}(Q)$
- q *after* $\epsilon = \{q\}$
- q *after* $\mu\rho = Q'$ *after* ρ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$

- *Straces*(q_0) =



Examples for *after*

- $\textit{after} : Q \times L^* \to \mathbb{P}(Q)$
- $q \textit{ after } \epsilon = \{q\}$
- $q \textit{ after } \mu\rho = Q' \textit{ after } \rho$ where
  $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$
- *Straces*(q_0) = {?, ?, δ?, δ?, ??,
  ????!???!, ...}



Formal definitions (continued)

- $in(q) = \{a \in L_I \mid q \text{ after } a \neq \emptyset\}$

Formal definitions (continued)

- $in(q) = \{a \in L_I \mid q \text{ after } a \neq \emptyset\}$
- $in(Q') = \bigcup_{q \in Q'} in(q)$

Formal definitions (continued)

- $in(q) = \{a \in L_I \mid q \text{ after } a \neq \emptyset\}$
- $in(Q') = \bigcup_{q \in Q'} in(q)$
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \text{ after } x \neq \emptyset\}$

Formal definitions (continued)

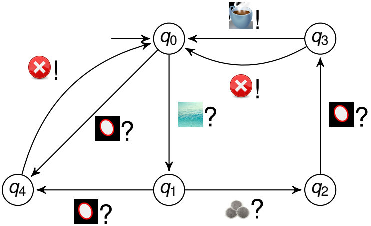
- $in(q) = \{a \in L_I \mid q \text{ after } a \neq \emptyset\}$
- $in(Q') = \bigcup_{q \in Q'} in(q)$
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \text{ after } x \neq \emptyset\}$
- $out(Q') = \bigcup_{q \in Q'} out(q)$

Formal definitions (continued)

- $in(q) = \{a \in L_I \mid q \text{ after } a \neq \emptyset\}$
- $in(Q') = \bigcup_{q \in Q'} in(q)$
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \text{ after } x \neq \emptyset\}$
- $out(Q') = \bigcup_{q \in Q'} out(q)$
- LTS S is deterministic if $\forall q \in Q, \forall \rho \in Straces(q) : |q \text{ after } \rho| \leq 1$

Examples for definitions

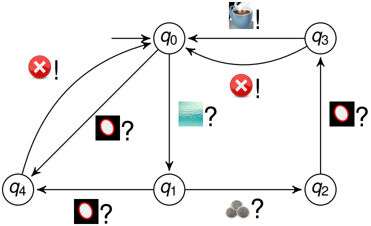
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \text{ after } x \neq \emptyset\}$
- LTS S is deterministic if $\forall q \in Q, \forall \rho \in Straces(q) : |q \text{ after } \rho| \leq 1$



- $out(q_3) =$

Examples for definitions

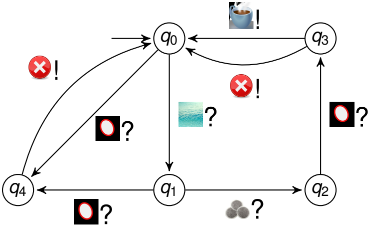
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \text{ after } x \neq \emptyset\}$
- LTS S is deterministic if $\forall q \in Q, \forall \rho \in Straces(q) : |q \text{ after } \rho| \leq 1$



- $out(q_3) = \{$$!,$ $!\}$

Examples for definitions

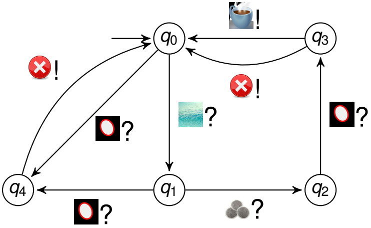
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \textit{ after } x \neq \emptyset\}$
- LTS S is deterministic if $\forall q \in Q, \forall \rho \in Straces(q) : |q \textit{ after } \rho| \leq 1$



- Deterministic?

Examples for definitions

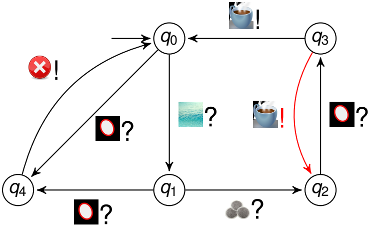
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \text{ after } x \neq \emptyset\}$
- LTS S is deterministic if $\forall q \in Q, \forall \rho \in Straces(q) : |q \text{ after } \rho| \leq 1$



- Deterministic? Yes

Examples for definitions

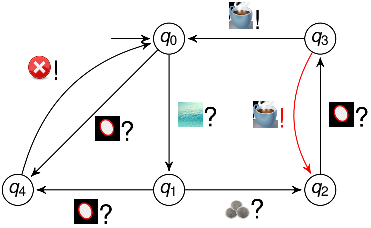
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \textit{ after } x \neq \emptyset\}$
- LTS S is deterministic if $\forall q \in Q, \forall \rho \in Straces(q) : |q \textit{ after } \rho| \leq 1$



- $out(q_3) =$

Examples for definitions

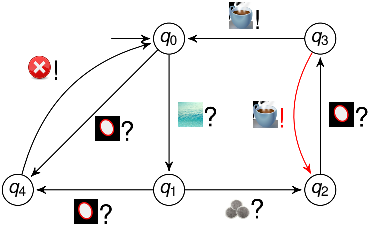
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \text{ after } x \neq \emptyset\}$
- LTS S is deterministic if $\forall q \in Q, \forall \rho \in Straces(q) : |q \text{ after } \rho| \leq 1$



- $out(q_3) =$ {!}

Examples for definitions

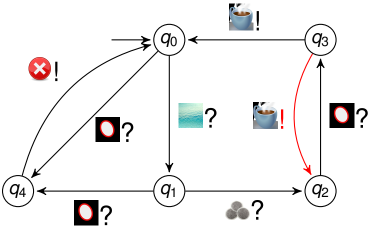
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \text{ } after \text{ } x \neq \emptyset\}$
- LTS S is deterministic if $\forall q \in Q, \forall \rho \in Straces(q) : |q \text{ } after \text{ } \rho| \leq 1$



- Deterministic?

Examples for definitions

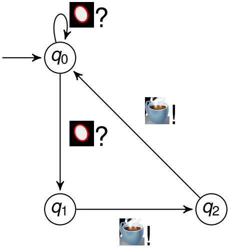
- $out(q) = \{x \in L_U \cup \{\delta\} \mid q \text{ after } x \neq \emptyset\}$
- LTS S is deterministic if $\forall q \in Q, \forall \rho \in Straces(q) : |q \text{ after } \rho| \leq 1$



- Deterministic? No

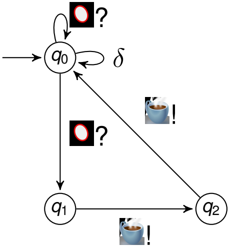
Determinisation of nondeterministic LTS

- Nondeterministic LTS:



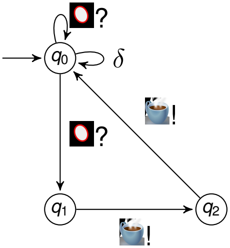
Determinisation of nondeterministic LTS

- Nondeterministic LTS:



Determinisation of nondeterministic LTS

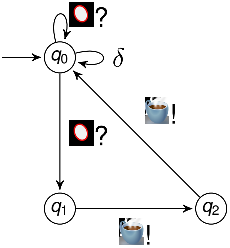
- Nondeterministic LTS:



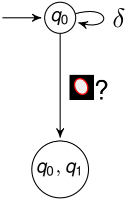
- Deterministic LTS

Determinisation of nondeterministic LTS

- Nondeterministic LTS:

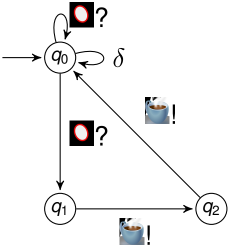


- Deterministic LTS

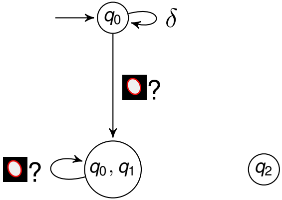


Determinisation of nondeterministic LTS

- Nondeterministic LTS:

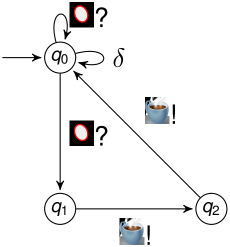


- Deterministic LTS

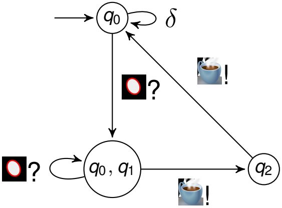


Determinisation of nondeterministic LTS

- Nondeterministic LTS:

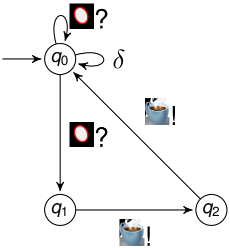


- Deterministic LTS with explicit δ

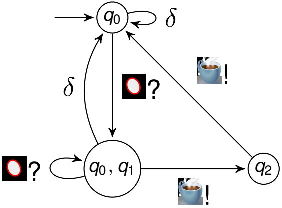


Determinisation of nondeterministic LTS

- Nondeterministic LTS:

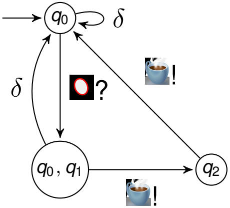


- Deterministic LTS



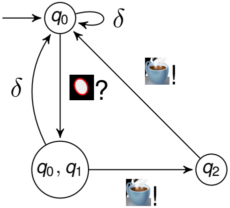
Q: does δ still denote quiescence?

Determinisation



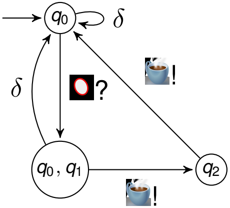
- Resulting LTS_{det} has:
  - Sets of states as states: $Q_{det} = \{\{q_0\}, \{q_0, q_1\}, \{q_2\}\}$

Determinisation



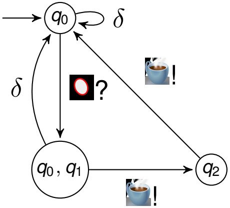
- Resulting LTS_{det} has:
  - Sets of states as states: $Q_{det} = \{\{q_0\}, \{q_0, q_1\}, \{q_2\}\}$
    - Algorithm potentially creates $2^n - 1$ states

Determinisation



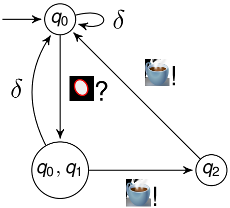
- Resulting LTS_{det} has:
  - Sets of states as states: $Q_{det} = \{\{q_0\}, \{q_0, q_1\}, \{q_2\}\}$
    - Algorithm potentially creates $2^n - 1$ states
  - δ in transition relation: $(\{q_0, q_1\}, \delta, \{q_0\}) \in T_{det}$

Determinisation



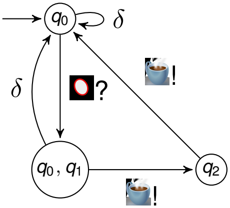
- Resulting LTS_{det} has:
  - Sets of states as states: $Q_{det} = \{\{q_0\}, \{q_0, q_1\}, \{q_2\}\}$
    - Algorithm potentially creates $2^n - 1$ states
  - δ in transition relation: $(\{q_0, q_1\}, \delta, \{q_0\}) \in T_{det}$
- define after_{det}:
  - $\mathit{after}_{det} : Q_{det} \times L^* \to \mathbb{P}(Q)$
  - $q\ \mathit{after}_{det}\ \epsilon = \{q\}$ (here ϵ denotes the sequence of 0 labels)

Determinisation



- Resulting LTS_{det} has:
  - Sets of states as states: $Q_{det} = \{\{q_0\}, \{q_0, q_1\}, \{q_2\}\}$
    - Algorithm potentially creates $2^n - 1$ states
  - δ in transition relation: $(\{q_0, q_1\}, \delta, \{q_0\}) \in T_{det}$
- define after_{det}:
  - $\mathit{after}_{det} : Q_{det} \times L^* \to \mathbb{P}(Q)$
  - $q\ \mathit{after}_{det}\ \epsilon = \{q\}$ (here ϵ denotes the sequence of 0 labels)
  - $q\ \mathit{after}_{det}\ \mu\rho = Q'\ \mathit{after}_{det}\ \rho$ where
    $Q' = \{q' \in Q \mid (q, \mu, q') \in T\} \cup \{q \mid \delta(q) \wedge \mu = \delta\}$

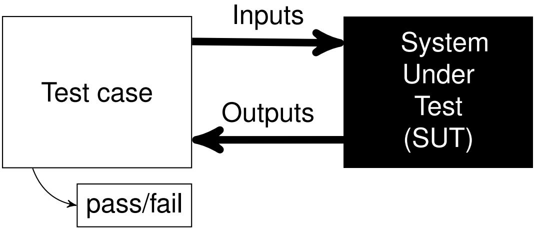
Determinisation



- Resulting LTS_{det} has:
  - Sets of states as states: $Q_{det} = \{\{q_0\}, \{q_0, q_1\}, \{q_2\}\}$
    - Algorithm potentially creates $2^n - 1$ states
  - δ in transition relation: $(\{q_0, q_1\}, \delta, \{q_0\}) \in T_{det}$
- define after_{det}:
  - $\mathit{after}_{det} : Q_{det} \times L^* \to \mathbb{P}(Q)$
  - $q \ \mathit{after}_{det}\ \epsilon = \{q\}$ (here ϵ denotes the sequence of 0 labels)
  - $q \ \mathit{after}_{det}\ \mu\rho = \{q' \in Q \mid (q, \mu, q') \in T\} \ \mathit{after}_{det}\ \rho$

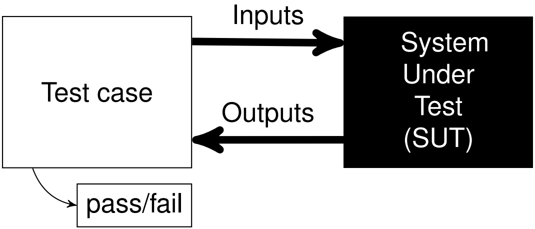
Test cases

Towards a formal definition for test cases



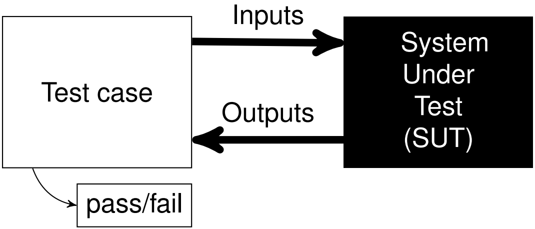
- A test case specifies:

Towards a formal definition for test cases



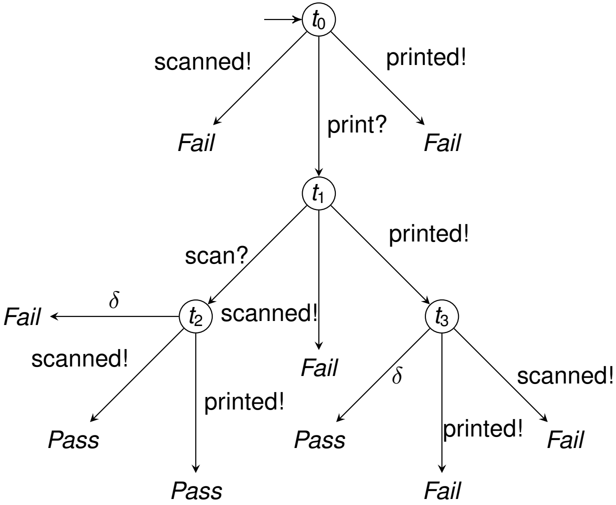
- A test case specifies:
  - which inputs should be provided to the SUT
  - which outputs are expected from the SUT
  - a verdict (pass or fail)

Towards a formal definition for test cases



- A test case specifies:
  - which inputs should be provided to the SUT
  - which outputs are expected from the SUT
  - a verdict (pass or fail)
  - a finite execution of the SUT
    - When the test case stops, it returns its verdict

Example test case for printer



Formal definition of test cases

- A test case for an LTS S is an LTS $t = (Q^t, L_I, L_U, T^t, q_0^t)$ s.t.:
  - t uses the same labels as S
  - There are two special states *Pass*, *Fail* $\in Q^t$
  - States *Pass* and *Fail* have self-loops for all outputs (incl. δ):
    $\forall x \in L_U \cup \{\delta\} : \textit{Pass after } x = \{\textit{Pass}\}$ and $\textit{Fail after } x = \{\textit{Fail}\}$
  - States *Pass* and *Fail* have no transition for inputs:
    $\forall a \in L_I : \textit{Pass after } a = \textit{Fail after } a = \emptyset$
  - t has no cycles except those in *Pass* and *Fail*
  - t is deterministic
  - Every state enables all outputs L_U, and either one input or δ: $\forall q \in Q^t : (|in(q)| = 0 \wedge out(q) = L_U \cup \{\delta\}) \vee (out(q) = L_U \wedge |in(q)| = 1)$

Formal definition of test cases

- A test case for an LTS S is an LTS $t = (Q^t, L_I, L_U, T^t, q_0^t)$ s.t.:
  - t uses the same labels as S
  - There are two special states *Pass*, *Fail* $\in Q^t$
  - States *Pass* and *Fail* have self-loops for all outputs (incl. δ): $\forall x \in L_U \cup \{\delta\} : \textit{Pass after } x = \{\textit{Pass}\}$ and $\textit{Fail after } x = \{\textit{Fail}\}$
  - States *Pass* and *Fail* have no transition for inputs: $\forall a \in L_I : \textit{Pass after } a = \textit{Fail after } a = \emptyset$
  - t has no cycles except those in *Pass* and *Fail*
  - t is deterministic
  - Every state enables all outputs L_U, and either one input or δ: $\forall q \in Q^t : (|in(q)| = 0 \wedge out(q) = L_U \cup \{\delta\}) \vee (out(q) = L_U \wedge |in(q)| = 1)$

Q1: why self-loops for modelling that the test case stops?

Formal definition of test cases

- A test case for an LTS S is an LTS $t = (Q^t, L_I, L_U, T^t, q_0^t)$ s.t.:
  - t uses the same labels as S
  - There are two special states *Pass*, *Fail* $\in Q^t$
  - States *Pass* and *Fail* have self-loops for all outputs (incl. δ): $\forall x \in L_U \cup \{\delta\} : \mathit{Pass}\ \mathit{after}\ x = \{\mathit{Pass}\}$ and $\mathit{Fail}\ \mathit{after}\ x = \{\mathit{Fail}\}$
  - States *Pass* and *Fail* have no transition for inputs: $\forall a \in L_I : \mathit{Pass}\ \mathit{after}\ a = \mathit{Fail}\ \mathit{after}\ a = \emptyset$
  - t has no cycles except those in *Pass* and *Fail*
  - t is deterministic
  - Every state enables all outputs L_U, and either one input or δ: $\forall q \in Q^t : (|in(q)| = 0 \wedge out(q) = L_U \cup \{\delta\}) \vee (out(q) = L_U \wedge |in(q)| = 1)$

Q1: why self-loops for modelling that the test case stops?

- Self-loops omitted in graphical representations of test cases

Formal definition of test cases

- A test case for an LTS S is an LTS $t = (Q^t, L_I, L_U, T^t, q_0^t)$ s.t.:
  - t uses the same labels as S
  - There are two special states *Pass*, *Fail* $\in Q^t$
  - States *Pass* and *Fail* have self-loops for all outputs (incl. δ): $\forall x \in L_U \cup \{\delta\} : \mathit{Pass\ after\ x} = \{\mathit{Pass}\}$ and $\mathit{Fail\ after\ x} = \{\mathit{Fail}\}$
  - States *Pass* and *Fail* have no transition for inputs: $\forall a \in L_I : \mathit{Pass\ after\ a} = \mathit{Fail\ after\ a} = \emptyset$
  - t has no cycles except those in *Pass* and *Fail*
  - t is deterministic
  - Every state enables all outputs L_U, and either one input or δ: $\forall q \in Q^t : (|in(q)| = 0 \wedge out(q) = L_U \cup \{\delta\}) \vee (out(q) = L_U \wedge |in(q)| = 1)$

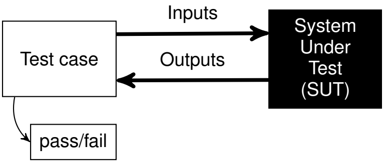
Q1: why self-loops for modelling that the test case stops?

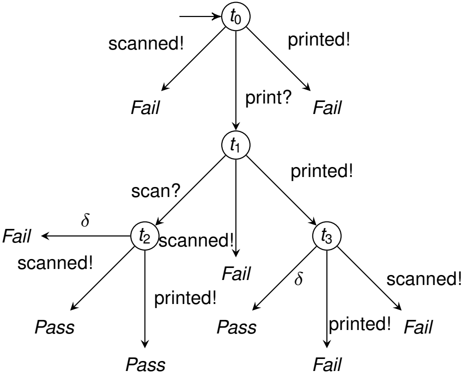
- Self-loops omitted in graphical representations of test cases

Q2: why is a test case a tree (and not a trace)?

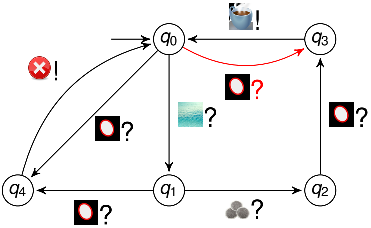
Test execution

- Provide input, or
- wait for output, or
- stop with verdict



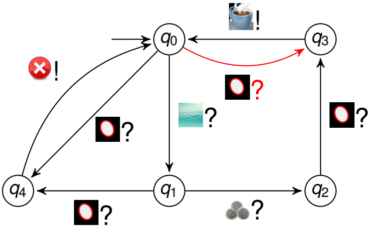


Test cases for nondeterministic LTSs

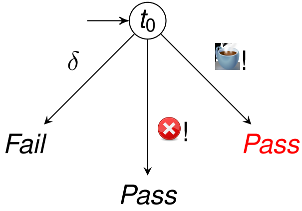


- Observe chosen ?-transition (after SUT executed ?)

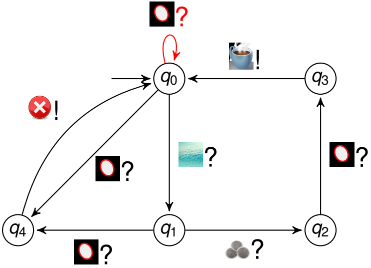
Test cases for nondeterministic LTSs



- Observe chosen ?-transition (after SUT executed ?)

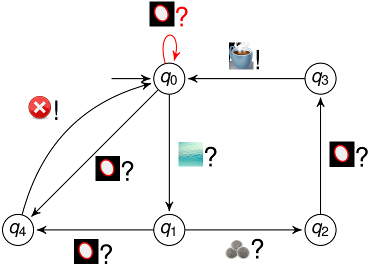


Test cases for nondeterministic LTSs

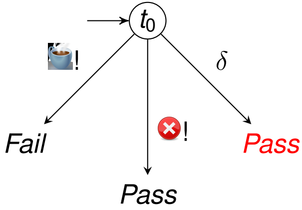


- Observe chosen ?-transition (after SUT executed ?)

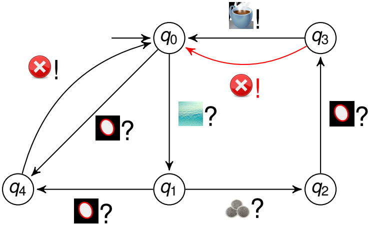
Test cases for nondeterministic LTSs



- Observe chosen ?-transition (after SUT executed ?)

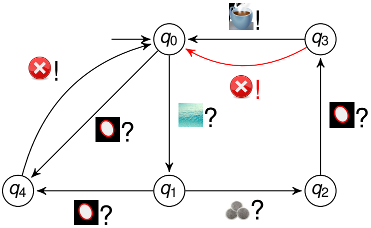


Test cases for nondeterministic LTSs

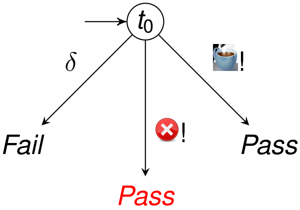


- Test case to observe !, after the SUT executed ? ? ?

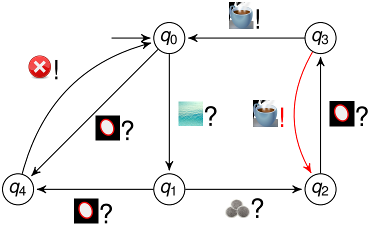
Test cases for nondeterministic LTSs



- Test case to observe !, after the SUT executed ???

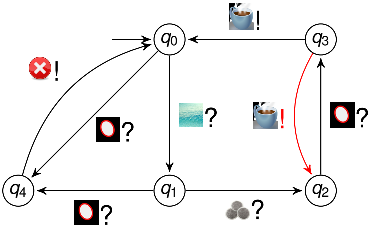


Test cases for nondeterministic LTSs

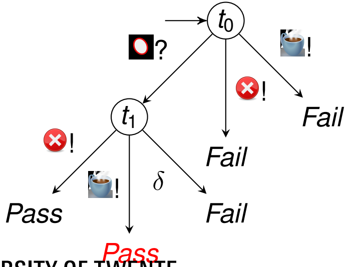


- Observe chosen !-transition, after SUT executed ??? !

Test cases for nondeterministic LTSs



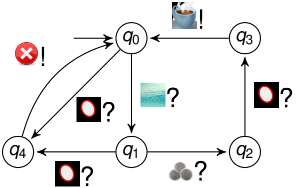
- Observe chosen !-transition, after SUT executed ? ? ? !



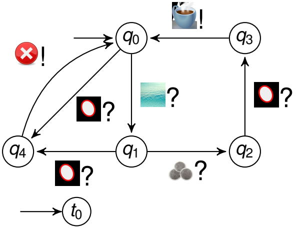
Special test cases

- Distinguishing test case: a test case that distinguishes two start states
- Homing test case: a test case that transitions to a particular state from any starting state

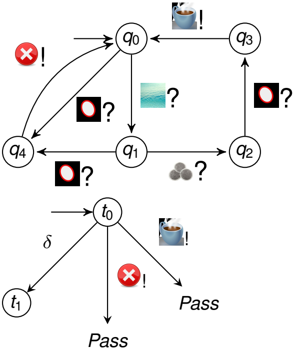
Homing test case for q_0



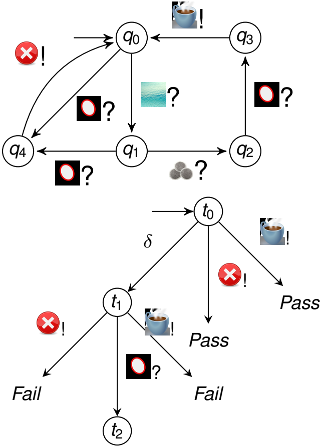
Homing test case for q_0



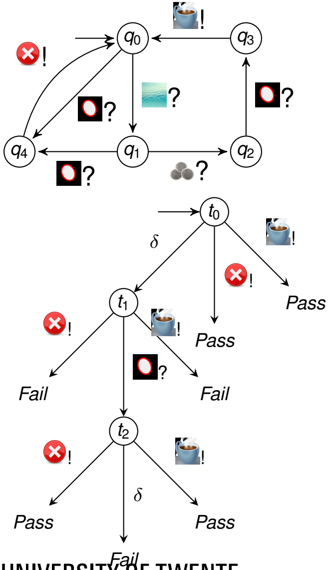
Homing test case for q_0



Homing test case for q_0



Homing test case for q_0



Next times

- Tutorial next Tuesday: exercises prepare for exam
- Lecture next Thursday: ioco & test generation algorithms