Test Generation – Finite State Models

Andrea Polini

Advanced Topics on Software Engineering – Software Testing
MSc in Computer Science
University of Camerino

Models in the Design Phase

Design Phase

- Between the requirements phase and the implementation phase "*The last you start the first you finish*"
- Produce models in order to clarify requirements and to better formalize them
- Models can be the source of test set derivation strategies

Various modeling notations for behavioral specification of a software system have been proposed. Which to use depends on the system you are developing, and the aspects you would like to highlight:

- Finite State Machines
- Petri Nets
- Statecharts
- Message sequence charts

Finite State Machines

FSM

A finite state machine is a six-tuple $<\mathscr{X}, \mathscr{Y}, \mathscr{Q}, q_0, \delta, \mathscr{O}>$ where:

- $\mathscr{X}$: finite set of input symbols
- $\mathscr{Y}$: finite set of output symbols
- $\mathscr{Q}$: finite set of states
- $q_0 \in \mathscr{Q}$: initial state
- δ: transition function ($\mathscr{Q} \times \mathscr{X} \rightarrow \mathscr{Q}$)
- $\mathscr{O}$: output function ($\mathscr{Q} \times \mathscr{X} \rightarrow \mathscr{Y}$)

Many possible extensions:

- Transition and output functions can consider strings
- Definiton of the set of accepting states $\mathscr{F} \subseteq \mathscr{Q}$
- Non determinism

Properties of FSM

Useful properties/concepts for test generation

- Completely specified (input enabled)
  - $\forall(q_i \in \mathscr{Q}, a \in \mathscr{X}).\exists q_j \in \mathscr{Q}.\delta(q_i, a) = q_j$
- Strongly connected
  - $\forall(q_i, q_j) \in \mathscr{Q} \times \mathscr{Q}.\exists s \in X^*.\delta^*(q_i, s) = q_j$
- V-equivalence (distinguishable)
  - Let M_1 and M_2 two FSMs. Let $\mathscr{V}$ denote a set of non-empty string on the input alphabet $\mathscr{X}$, and $q_i \in \mathscr{Q}_1$ and $q_j \in \mathscr{Q}_2$. q_i and q_j are considered *$\mathscr{V} - equivalent$* if $\mathscr{O}_1(q_i, s) = \mathscr{O}_2(q_j, s)$. If q_i and q_j are *$\mathscr{V} - equivalent$* given any set $\mathscr{V} \subseteq \mathscr{X}^+$ than they are said to be *equivalent* ($q_i \equiv q_j$). If states are not equivalent they are said to be *distinguishable*.

Properties of FSM....cntd

Useful properties/concepts for test generation...cntd

- Machine equivalence
  - M_1 and M_2 are said to be *equivalent* if $\forall q_i \in \mathcal{Q}_1 . \exists q_j \in \mathcal{Q}_2 . q_i \equiv q_j$ and viceversa.
- k-equivalence
  - Let M_1 and M_2 two FSMs and $q_i \in \mathcal{Q}_1$ and $q_j \in \mathcal{Q}_1$ and $k \in \mathbb{N}$. q_i and q_j are said to be $\mathcal{K} - equivalent$ if they are $\mathcal{V} - equivalent$ for $\mathcal{V} = \{s \in X^+ | \mid s \mid \leq k\}$
- Minimal machine
  - an FSM is considered *minimal* if the number of its states is less than or equal to any other *equivalent* FSM

Conformance Testing

Conformance Testing

Relates to testing of communication protocols. It aims at assessing that an implementation of a protocol conform to its specification. Protocols generally specify:

- Control rules (FSM)
- Data rules

Developed techniques are equally applicable when the specification is refined into an FSM

The Testing Problem

FSM and Testing

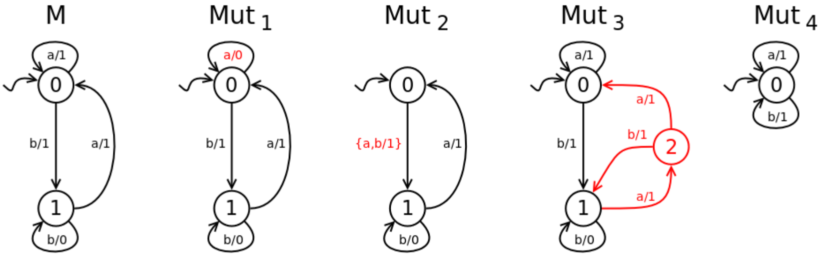
- Reset inputs ($\mathscr{X} = \mathscr{X} \cup \{Re\}$, and $\mathscr{Y} = \mathscr{Y} \cup \{null\}$)
- Testing based on requirements checks if the implementation conforms to the machine on a given requirement.
- The testing problem is reconducted to an equivalence (nevertheless finite experiments). Is the SUT (IUT) equivalent to the machine defined during design?
- Fault model for FSM – given a fault model the challenge is to generate a test set T from a design M_d where any fault in M_i of the type in the fault model is guaranteed to be revealed when tested against T
  - Operation error (refers to issues with $\mathscr{O}$)
  - Transfer error (refers to issues with δ)
  - Extra-state error (refers to issues with $\mathscr{Q}$ and δ)
  - Missing-state error (refers to issues with $\mathscr{Q}$ and δ)

Mutation of FSMs

Mutant

A mutant of an FMS M_d is an FSM obtained by introducing one or more errors one or more times.

- Equivalent mutants: mutants that could not be distinguishable from the originating machine



The Testing Problem

Fault coverage

Techniques to measure the goodness of a test set in relation to the number of errors that it reveals in a given implementation M_i.

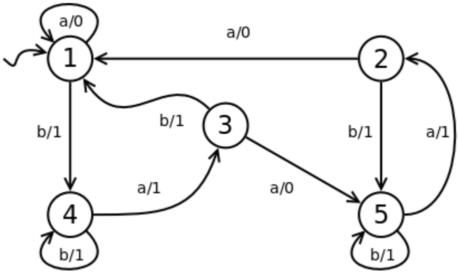
- N_t: total number of first order mutants of the machine M used for generating tests.
- N_e: Number of mutants that are equivalent to M
- N_f: Number of mutants that are distinguished by test set T generated using some test generation method.
- N_l: Number of mutants that are not distinguished by T

The fault coverage of a test suite T with respect to a design M is denoted by $FC(T, M)$ and computed as follows:

$$\begin{aligned} FC(T, M) &= \text{Number of mutants not distinguished by T /} \\ &\quad\ \text{Number of mutants that are not equivalent to M} \\ &= (N_t - N_e - N_f)/(N_t - N_e) \end{aligned}$$

Characterization Set

Let $M = <\mathscr{X}, \mathscr{Y}, \mathscr{Q}, q_1, \delta, \mathscr{O}>$ an FSM that is minimal and complete. A characterization set for M, denoted as $\mathscr{W}$, is a finite set of input sequences that distinguish the behaviour of any pair of states in M.



K-equivalence partitions

The notion of $\mathscr{K} - $ *equivalence* leads to the notion of $\mathscr{K} - $ *equivalence partitions*.

Given an FSM a $\mathscr{K} - $ *equivalence partition* of $\mathscr{Q}$, denoted by $\mathscr{P}_k$, is a collection of n finite sets of states denoted as $\Sigma_{k_1}, \Sigma_{k_2}, ..., \Sigma_{k_n}$ such that:

- $\cup_{i=1...n}\Sigma_{K_i} = \mathscr{Q}$
- States in Σ_{k_j}, for $1 \leq j \leq n$ are $\mathscr{K} - $ *equivalent*
- if $q_l \in \Sigma_{k_i}$ and $q_m \in \Sigma_{k_j}$, for $i \neq j$, then q_l and q_m must be $\mathscr{K} - $ *distinguishable*

$\mathscr{K} - $ *equivalence* partitions can be derived using an iterative approach for increasing number of $\mathscr{K}$

K-equivalence partitions

The notion of $\mathscr{K} -$ *equivalence* leads to the notion of $\mathscr{K} -$ *equivalence partitions*.

Given an FSM a $\mathscr{K} -$ *equivalence partition* of $\mathscr{Q}$, denoted by $\mathscr{P}_k$, is a collection of n finite sets of states denoted as $\Sigma_{k_1}, \Sigma_{k_2}, ..., \Sigma_{k_n}$ such that:

- $\cup_{i=1...n} \Sigma_{K_i} = \mathscr{Q}$
- States in Σ_{k_j}, for $1 \leq j \leq n$ are $\mathscr{K} -$ *equivalent*
- if $q_l \in \Sigma_{k_i}$ and $q_m \in \Sigma_{k_j}$, for $i \neq j$, then q_l and q_m must be $\mathscr{K} -$ *distinguishable*

$\mathscr{K} -$ *equivalence* partitions can be derived using an iterative approach for increasing number of $\mathscr{K}$

Let's use the intuition

Let's build K-equivalence partitions for the previous FSM

How to derive $\mathscr{W}$ from K-equivalence partitions

1. Let M an FSM for which $P = \{P_1, P_2, ..., P_n\}$ is the set of k-equivalence partition. $\mathscr{W} = \emptyset$
2. Repeat the steps (a) through (d) given below for each pair of states (q_i, q_j), $i \neq j$, in M
   - (a) Find r ($1 \leq r < n$ such that the states in pair (q_i, q_j) belong to the same group in P_r but not in P_{r+1}. If such an r is found then move to step (b) otherwise we find an $\eta \in \mathscr{X}$ such that $\mathscr{O}(q_i, \eta) \neq \mathscr{O}(q_j, \eta)$, set $\mathscr{W} = \mathscr{W} \cup \{\eta\}$ and continue with the next available pair of states. The length of the minimal distinguishing sequence for (q_i, q_j) is $r + 1$.
   - (b) Initialize $z = \epsilon$. Let $p_1 = q_i$ and $p_2 = q_j$ be the current pair of states. Execute steps (i) through (iii) given below for $m = r, r - 1, ..., 1$
     - (i) Find an input symbol η in P_m such that $\mathscr{G}(p_1, \eta) \neq \mathscr{G}(p_2, \eta)$. In case there is more than one symbol that satisfy the condition in this step, then select one arbitrarily.
     - (ii) set $z = z\eta$
     - (iii) set $p_1 = \delta(p_1, \eta)$ and $p_2 = \delta(p_2, \eta)$
   - (c) Find an $\eta \in \mathscr{X}$ such that $\mathscr{O}(p_1, \eta) \neq \mathscr{O}(p_2, \eta)$. Set $z = z\eta$
   - (d) The distinguishing sequence for the pair (q_i, q_j) is the sequence z. Set $\mathscr{W} = \mathscr{W} \cup \{z\}$

Example

- Termination of the $\mathscr{W}$ – *procedure* guarantees the generation of distinguishing sequence for each pair.

S_i	S_i	x	$\mathscr{O}(S_i, x)$	$\mathscr{O}(S_j, x)$
1	2	baaa	1	0
1	3	aa	0	1
1	4	a	0	1
1	5	a	0	1
2	3	aa	0	1
2	4	a	0	1
2	5	a	0	1
3	4	a	0	1
3	5	a	0	1
4	5	aaa	1	0

Example

- Termination of the $\mathscr{W} -$ *procedure* guarantees the generation of distinguishing sequence for each pair.

S_i	S_i	x	$\mathscr{O}(S_i, x)$	$\mathscr{O}(S_j, x)$
1	2	baaa	1	0
1	3	aa	0	1
1	4	a	0	1
1	5	a	0	1
2	3	aa	0	1
2	4	a	0	1
2	5	a	0	1
3	4	a	0	1
3	5	a	0	1
4	5	aaa	1	0

The W-Method

The W-Method aims at deriving a test set to check the implementation (Implementation Under Test - IUT) of an FSM model

Assumptions

- M is completely specified, minimal, connected, and deterministic
- M starts in a fixed initial states
- M can be reset to the initial state. A `null` output is generated by the reset
- M and IUT have the same input alphabet

The W-Method

The W-Method aims at deriving a test set to check the implementation (Implementation Under Test - IUT) of an FSM model

Assumptions

- M is completely specified, minimal, connected, and deterministic
- M starts in a fixed initial states
- M can be reset to the initial state. A `null` output is generated by the reset
- M and IUT have the same input alphabet

W-Method steps

Given an FSM $\mathscr{M} =< \mathscr{X}, \mathscr{Y}, \mathscr{Q}, q_0, \delta, \mathscr{O} >$ the W-method consists of the following steps:

1. Estimate the maximum number of states in the correct design
2. Construct the characterization set $\mathscr{W}$ for the given machine $\mathscr{M}$
3. Construct the testing tree for $\mathscr{M}$ and determine the transition cover set $\mathscr{P}$
4. Construct set $\mathscr{Z}$
5. $\mathscr{P} \cdot \mathscr{Z}$ is the desired test set

Computation of the transition cover set

$\mathscr{P}$ - transition cover set

Let q_i and $q_j, i \neq j$ be two states of $\mathscr{M}$. $\mathscr{P}$ consists of sequences $s \cdot x$ s.t. $\delta(q_0, s) = q_i \wedge \delta(q_i, x) = q_j$ for $s \in \mathscr{X}^* \wedge x \in \mathscr{X}$. The set can be constructed using the testing tree for $\mathscr{M}$.

Testing tree

The testing tree for an FSM $\mathscr{M}$ can be constructed as follows:

1. State q_0 is the root of the tree
2. Suppose that the testing tree has been constructed till level k. The $(k+1)^{th}$ level is built as follows:
   - Select a node n at level k. If n appears at any level from 1 to $k-1$ then n is a leaf node. Otherwise expand it by adding branch from node n to a new node m if $\delta(n, x) = m$ for $x \in \mathscr{X}$. This branch is labeled as x.

Computation of the transition cover set

$\mathscr{P}$ - transition cover set

Let q_i and $q_j, i \neq j$ be two states of $\mathscr{M}$. $\mathscr{P}$ consists of sequences $s \cdot x$ s.t. $\delta(q_0, s) = q_i \wedge \delta(q_i, x) = q_j$ for $s \in \mathscr{X}^* \wedge x \in \mathscr{X}$. The set can be constructed using the testing tree for $\mathscr{M}$.

Testing tree

The testing tree for an FSM $\mathscr{M}$ can be constructed as follows:

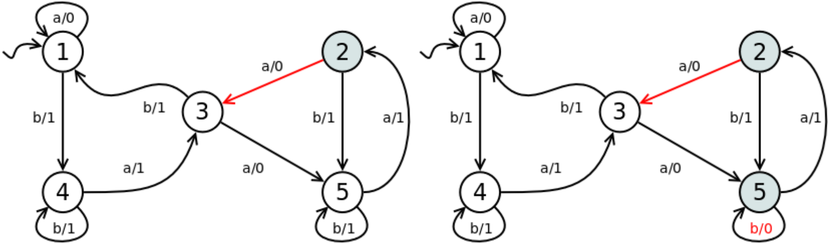
1. State q_0 is the root of the tree
2. Suppose that the testing tree has been constructed till level k. The $(k+1)^{th}$ level is built as follows:
   - Select a node n at level k. If n appears at any level from 1 to $k-1$ then n is a leaf node. Otherwise expand it by adding branch from node n to a new node m if $\delta(n, x) = m$ for $x \in \mathscr{X}$. This branch is labeled as x.

Constructing $\mathscr{Z}$

The set $\mathscr{Z}$

Suppose number of states estimates to be m for the IUT, and n in the specification $m > n$. We compute $\mathscr{Z}$ as:
$\mathscr{Z} = (\mathscr{X}^0 \cdot \mathscr{W}) \cup (\mathscr{X} \cdot \mathscr{W}) \cup (\mathscr{X}^1 \cdot \mathscr{W}) \cdots \cup (\mathscr{X}^{m-1-n} \cdot \mathscr{W}) \cup (\mathscr{X}^{m-n} \cdot \mathscr{W})$

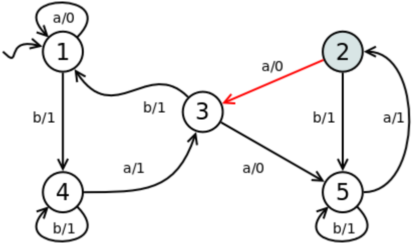
Deriving a test set – $\mathscr{P} \cdot \mathscr{Z}$



Try sequences:

- *baaaaaa*
- *baaba*

Deriving a test set – $\mathscr{P} \cdot \mathscr{Z}$



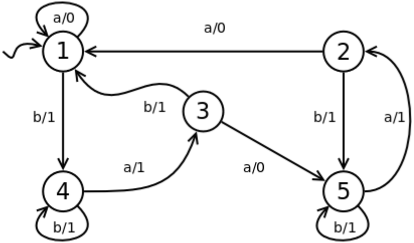
Try sequences:

- *baaaaaa*
- *baaba*

$\mathscr{W}$-method fault detection rationale

- A test case generated by the $\mathscr{W}$ − *method* is of the form $r \cdot s$ where $r \in \mathscr{P}$ and $s \in \mathscr{W}$
  - Why can we detect operation errors?
  - Why can we detect transfer errors?

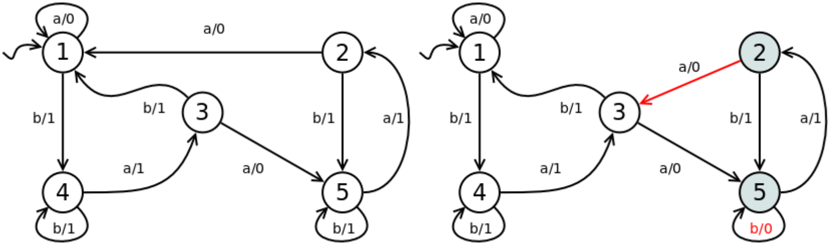
$$\mathscr{P} = \{\epsilon, a, b, bb, ba, bab, baa, baab, baaa, baaab, baaaa\}$$
$$\mathscr{W} = \{a, aa, aaa, baaa\}$$



$\mathscr{W}$-method fault detection rationale

- A test case generated by the $\mathscr{W}$ − *method* is of the form $r \cdot s$ where $r \in \mathscr{P}$ and $s \in \mathscr{W}$
  - Why can we detect operation errors?
  - Why can we detect transfer errors?

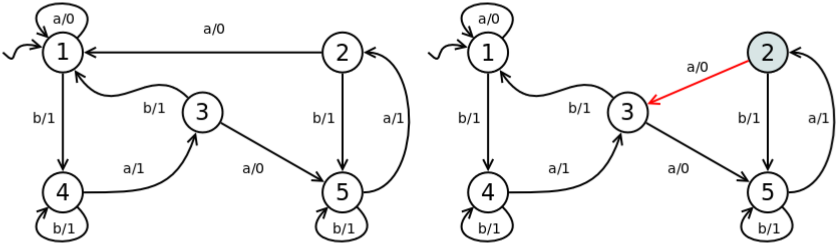
$$\mathscr{P} = \{\epsilon, a, b, bb, ba, bab, baa, baab, baaa, baaab, baaaa\}$$
$$\mathscr{W} = \{a, aa, aaa, baaa\}$$



$\mathscr{W}$-method fault detection rationale

- A test case generated by the $\mathscr{W} - method$ is of the form $r \cdot s$ where $r \in \mathscr{P}$ and $s \in \mathscr{W}$
  - Why can we detect operation errors?
  - Why can we detect transfer errors?

$$\mathscr{P} = \{\epsilon, a, b, bb, ba, bab, baa, baab, baaa, baaab, baaaa\}$$
$$\mathscr{W} = \{a, aa, aaa, baaa\}$$



The partial $\mathscr{W} - method$ (aka $Wp - method$)

$Wp - method$

Main characteristics:

- It considers minimal, complete and connected FSM
- is inspired by the $\mathscr{W} - method$ it generates smaller test sets
- uses a derivation phase split in two phases that make use of state identification sets $\mathscr{W}_i$ instead of characterization set $\mathscr{W}$
- uses the state cover set ($\mathscr{S}$) to derive the test set.

Identification Set and State Cover Set

Identification Set

The Identification Set is associated to each state $q \in \mathcal{Q}$ of an FSM.

An Identification set for state $q_i \in \mathcal{Q}$, where $|\mathcal{Q}| = n$, is denoted by $\mathscr{W}_i$ and has the following properties:

1. $\mathscr{W}_i \subseteq \mathscr{W}$ per $1 < i \leq n$
2. $\exists j, s.1 \leq j \leq n \wedge s \in \mathscr{W}_i \wedge \mathscr{O}(q_i, s) \neq \mathscr{O}(q_j, s)$
3. No subset of $\mathscr{W}_i$ satisfies property 2.

State Cover Set

The state cover set is a nonempty set of sequences ($\mathscr{S} \subseteq \mathscr{X}^*$ s.t.:

- $\forall q_i \in \mathcal{Q}\ \exists r \in \mathscr{S} s.t. \delta(q_0, r) = q_i$

From the definition it is evident that $\mathscr{S} \subseteq \mathscr{P}$

Identification Set and State Cover Set

Identification Set

The Identification Set is associated to each state $q \in \mathscr{Q}$ of an FSM.

An Identification set for state $q_i \in \mathscr{Q}$, where $|\mathscr{Q}| = n$, is denoted by $\mathscr{W}_i$ and has the following properties:

1. $\mathscr{W}_i \subseteq \mathscr{W}$ per $1 < i \leq n$
2. $\exists j, s.1 \leq j \leq n \wedge s \in \mathscr{W}_i \wedge \mathscr{O}(q_i, s) \neq \mathscr{O}(q_j, s)$
3. No subset of $\mathscr{W}_i$ satisfies property 2.

State Cover Set

The state cover set is a nonempty set of sequences ($\mathscr{S} \subseteq \mathscr{X}^*$ s.t.:

- $\forall q_i \in \mathscr{Q} \; \exists r \in \mathscr{S} s.t. \delta(q_0, r) = q_i$

From the definition it is evident that $\mathscr{S} \subseteq \mathscr{P}$

Exercise

Compute the State cover set and the identification set for the usual automaton

The $\mathscr{W}p$ procedure (assuming $m = n$)

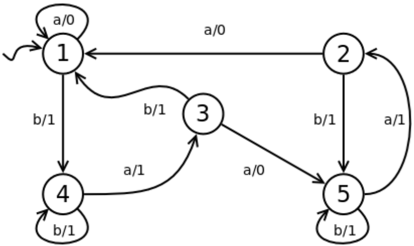
The test set derived using the $\mathscr{W}p - method$ is given by the union to two test sets $\mathscr{T}_1$, $\mathscr{T}_2$ calculated according to the following procedure:

1. Compute sets $\mathscr{P}$, $\mathscr{S}$, $\mathscr{W}$, and $\mathscr{W}_i$
2. $\mathscr{T}_1 = \mathscr{S} \cdot \mathscr{W}$
3. Let $\mathcal{W} = \{\mathscr{W}_1, \mathscr{W}_2, \ldots, \mathscr{W}_n\}$
4. Let $\mathcal{R} = \{r_1, r_2, \ldots, r_k\}$ where $\mathcal{R} = \mathscr{P} - \mathscr{S}$ and $r_j \in \mathcal{R}$ is s.t. $\delta(q_0, r_j) = q_i$
5. $\mathscr{T}_2 = \mathcal{R} \otimes \mathcal{W} = \cup_{j=1}^{K}(\{r_j\} \cdot \mathscr{W}_i)$ where $\mathscr{W}_i \in \mathcal{W}$ is the state identification set for state q_i ($\otimes$ is the partial string concatenation operator)

$\mathscr{W}p$ – *method* rationale

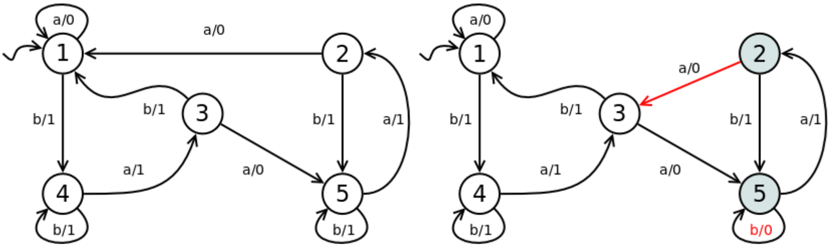
- Phase 1: test are of the form *uv* where $u \in \mathscr{S}$ and $v \in \mathscr{W}$. Reach each state than check if it is distinguishable from another one
- Phase 2: test covers all the missing transitions and then check if the reached state is different from the one specified in the model

$\mathscr{W}p - method$ in practice



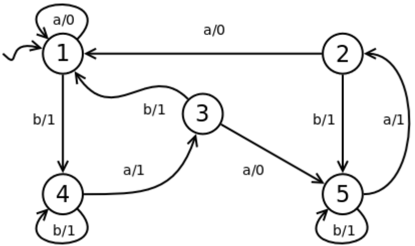
$\mathscr{W} = \{a, aa, aaa, baaa\}$
$\mathscr{P} = \{\epsilon, a, b, bb, ba, bab, baa, baab, baaa, baaab, baaaa\}$
$\mathscr{S} = \{\epsilon, b, ba, baa, baaa\}$
$\mathscr{W}_1 = \{baaa, aa, a\}$, $\mathscr{W}_2 = \{baaa, aa, a\}$, $\mathscr{W}_3 = \{aa, a\}$
$\mathscr{W}_4 = \{aaa, a\}$, $\mathscr{W}_5 = \{aaa, a\}$

$\mathscr{W}p - method$ in practice



$\mathscr{W} = \{a, aa, aaa, baaa\}$
$\mathscr{P} = \{\epsilon, a, b, bb, ba, bab, baa, baab, baaa, baaab, baaaa\}$
$\mathscr{S} = \{\epsilon, b, ba, baa, baaa\}$
$\mathscr{W}_1 = \{baaa, aa, a\}$, $\mathscr{W}_2 = \{baaa, aa, a\}$, $\mathscr{W}_3 = \{aa, a\}$
$\mathscr{W}_4 = \{aaa, a\}$, $\mathscr{W}_5 = \{aaa, a\}$

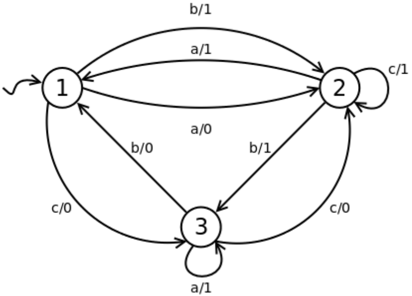
$\mathscr{W}p - method$ in practice



$\mathscr{W} = \{a, aa, aaa, baaa\}$
$\mathscr{P} = \{\epsilon, a, b, bb, ba, bab, baa, baab, baaa, baaab, baaaa\}$
$\mathscr{S} = \{\epsilon, b, ba, baa, baaa\}$
$\mathscr{W}_1 = \{baaa, aa, a\}$, $\mathscr{W}_2 = \{baaa, aa, a\}$, $\mathscr{W}_3 = \{aa, a\}$
$\mathscr{W}_4 = \{aaa, a\}$, $\mathscr{W}_5 = \{aaa, a\}$

Is it phase 2 needed?

Let's consider the following FSM:



Now introduce an operation error or a transfer error on a "*c*" transition

The $\mathscr{W}p$ procedure (assuming $m > n$)

Modify the derivation of the two sets as follows:

- $\mathscr{T}_1 = \mathscr{S} \cdot \mathscr{Z}$ where $\mathscr{Z} = \mathscr{X}[m-n] \cdot \mathscr{W}$
- $\mathscr{T}_2 = (\mathcal{R} \cdot \mathscr{X}[m-n]) \otimes \mathcal{W}$
    - Let $\mathcal{S} = \mathcal{R} \cdot \mathscr{X}[m-n] = \{s | s = r \cdot u \text{ s.t. } r \in \mathcal{R} \wedge u \in \mathscr{X}[m-n]\}$ then $\mathscr{T}_2 = \mathcal{S} \otimes \mathscr{W} = \cup_{s \in \mathscr{S}} (s \cdot \mathscr{W}_l)$ where $\delta(q_0, s) = \delta(\delta(q_0, r), u) = q_l$

Possible alternatives to W-method

- W-method high effectiveness in bugs identification
- High number of generated tests

To solve this issue alternative solutions have been proposed possibly reducing effectiveness:

- UIO-sequence method
- Distinguishing signatures