Traces and specifications

- Trace = set of events + end time stamp
  - Event = message + time stamp
  - Prefix, extension, snapshot
- Specification \cong set of valid traces
  - Prefix-closed
  - Serial
Test step disjointness

- For any \( i, s, T, \) and \( T_1 \) and \( T_2 \) it must hold that if \( T_1 \neq T_2 \) and
- \( \xi(i, s, T)[T_1] > 0 \) and
- \( \xi(i, s, T)[T_2] > 0 \), then
- \( T_1 \prec T_2 \), and
- \( T_2 \prec T_1 \)
- A technical convenience
Progressivity

- There does not exist an infinite sequence $T_1, T_2, T_3, \ldots$ and a constant $K \in \mathbb{R}$ such that
  $\xi(i, s, T_i)[T_{i+1}] > 0$ for all $i$, but such that for all $T_i = <E_i, t_i>$ it holds that $t_i < K$.

Choice of $\xi$

- We have defined properties of $\xi$, not the function itself
- The particular choice for $\xi$ depends on
  - the set of implementations $I$,
  - the set of testing strategies $T$, and
  - the desired structure of test steps.
Trace probabilities

- Let $P[i, s, T]$ denote the probability of observing $T$ as a prefix of a long enough trace when strategy $s$ is executed against implementation $i$
- Idea is to compute the product of the preceding test step probabilities

Trace probabilities

- Random experiment:
- Implementation $i$ and a testing strategy $s$ fixed
- A trace prefix $T^*$ has fixed, $T^* = <E,K>$
- $s$ is executed against $i$ many times, yielding traces $T_1 = <E_1,t_1>$, $T_2 = <E_2,t_2>$, ..., such that for all $n$, $t_n > K$
- What is the probability that for a uniformly chosen $n$, $T_n[K] = T^*$?
  - $X[t]$ is that prefix of $X$ whose end time stamp is $t$
Solution

» Traces that end at test step boundaries are easy: compute product probability
» Traces that end at non-boundaries require an extra construct

Step 1: Traces at test step boundaries

» Denote by $P^*[i,s,T]$: 

$$\max_{T_1,\ldots,T_n: \prod_{i \in [1, n-1]} \xi(i,s,T_i)[T_{i+1}]}$$

where $T_1 = \epsilon$ and $T_n = T$

» $P^*[i,s,T]$ is the compound probability for trace $T$, if $T$ "happens" at test step boundary
Sketch

This trace does not end a test step boundary
Step 2: traces at non-boundaries

Denote by $P[i,s,T]$

$$P^*[i,s,T^*] \times \left( \sum_{T':T \preceq T'} \xi(i,s,T^*)[T'] \right)$$

where $T^*$ is the longest prefix of $T$ such that $P^*[i,s,T^*] > 0$
Sanity checks

- If $P^*[i,s,T] > 0$,
  - then $P[i,s,T] = P^*[i,s,T]$.
  - Ok.
- If $P[i,s,T] = 0$ (trace $T$ cannot be produced),
  - there still exists the greatest prefix $T^*$ of $T$ such that $P^*[i,s,T^*] > 0$.
  - Every test step succeeding $T^*$ must result in a trace differing from $T$ — ok.

Sanity check 1 memo

- Assume $P^*[i,s,T] > 0$
- Note $T^* = T$
- $P[i,s,T] = P^*[i,s,T] \times (\sum_{T': T \prec T'} \xi(i,s,T[T']))$
- The sum yields one
Execution summary

- $\xi$ defines execution semantics
- Properties for $\xi$
  - Gives probability distribution over traces
  - Test step = trace extension
  - Test step disjointness
  - Progressivity
- However, no concrete structure
- $P[i,s,T]$ is the probability of producing trace $T$ when $s$ is run against $i$
  - Hides test steps

Why test steps?

- 1 test step =
  - unit of testing cost
  - unit of benefit
- Testing can be stopped between test steps, but not during them
  $\rightarrow$ stopping criteria
- Technical construct for describing arbitrarily long executions without the concept of “an infinite trace” (there is no such concept here)
Cost or benefit

Measuring the “size” of a trace

- Temporal length = end time stamp
- Size of event set
- Number of test steps used to produce
Verdicts

- Specification: Guides
- Definition: Defines correctness of

Tester ← Interaction → SUT

Announces

Verdict = test result
Verdicts

- Pass
- Fail
- Error
- Confused

Verdicts

Correct execution

Incorrect execution
Verdicts explained

<table>
<thead>
<tr>
<th>Verdict</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass</td>
<td>System under test has behaved correctly</td>
</tr>
<tr>
<td>Fail</td>
<td>System under test has behaved incorrectly</td>
</tr>
<tr>
<td>Error</td>
<td>Tester has behaved incorrectly</td>
</tr>
<tr>
<td>Confused</td>
<td>Fail–and–Error, result produces by an ambiguous specification (a special corner case)</td>
</tr>
</tbody>
</table>

Verdicts as traffic lights

- Pass
- Fail
- Error ("lights broken")
- Confused
Calculating verdict

- Verdict is calculated from a trace $T$ and a specification $S$

$$\text{verdict}(T,S) \in \{ \text{pass, fail, error, conf} \}$$

Pass verdict

$$\text{verdict}(T,S) = \text{pass}$$

if and only if

$$T \in \text{Tr}(S)$$
Other verdicts

- Hence, $T \notin \text{Tr}(S)$ implies

  \[ \text{verdict}(T,S) \in \{ \text{fail, error, conf} \} \]

- There is one verdict for $T \in \text{Tr}(S)$, and three for the other case

Other verdicts

- The problem: how to classify the cases $T \notin \text{Tr}(S)$ into
  - errors of the SUT ($\rightarrow$ fail),
  - errors of the tester ($\rightarrow$ error),
  - and those cases where the erring party cannot be defined ($\rightarrow$ confused)?
Solution sketch (1)

Tester

Valid prefixes

SUT

\[ t = 0.2 \]

\[ t = 0.9 \]

\[ t = 1.2 \]

\[ t = 2 \]

Valid prefixes

Solution sketch (2)

Tester

Valid prefixes

SUT

\[ t = 0.2 \]

\[ t = 0.9 \]

Valid continuations?
Solution sketch (3)

- All valid continuations differ from T first at input events?
  - error [tester error]
- All valid continuations differ from T first at output events?
  - fail [SUT failure]
- Otherwise
  - confused [unclear]

Technicallity

- The set of end time stamps for the valid prefixes of T can be either open or closed at the upper boundary
- Open set requires basically a limit construct (as usual)
**Solution sketch (4)**

- **Tester**
- **SUT**

**Valid prefixes**

- **A**
- **t=0.2**
- **t=0.9-ε**
- **t=0.9**

**Valid continuations?**

---

**Details**

- Assume $T \notin Tr(S)$
- Let $V = Tr(S) \cap Pfx(T)$
  - Note: $ε \in V$
  - This is the set of valid **proper prefixes** of $T$
- Let $K = \{ t \mid \exists E: <E, t> \in V \}$
- $K$ is either
  - closed: $[0, t]$, or
  - open: $[0, t)$.
  - It is the set of **end time stamps** in $V$. 
Example (closed set)

- \( \text{Tr}(S) = \)
- \( \bigcup \{ \text{Pfx}(\langle \langle A, t' \rangle, t \rangle) \mid t \in [2, \infty), t' \leq 1 \} \)
- \( T = \langle \emptyset, 10 \rangle \)
- Note that \( T \notin \text{Tr}(S) \)
- \( V = \{ \langle \emptyset, t \rangle \mid t \leq 1 \} \)
- \( K = [0, 1] \)
- Especially \( \langle \emptyset, 1 \rangle \) is in \( V \), because \( \langle \langle A, 1 \rangle, 1.1 \rangle \) is valid

Example (open set)

- \( \text{Tr}(S) = \)
- \( \bigcup \{ \text{Pfx}(\langle \langle A, t' \rangle, t \rangle) \mid t \in [2, \infty), t' < 1 \} \)
- \( T = \langle \emptyset, 10 \rangle \)
- Note that \( T \notin \text{Tr}(S) \)
- \( V = \{ \langle \emptyset, t \rangle \mid t < 1 \} \)
- \( K = [0, 1) \)
- Especially \( \langle \emptyset, 1 \rangle \) is not in \( V \), because for any \( t' < 1 \), event \( \langle A, t' \rangle \) should belong to the event set at time 1
Details continued

- Choose $\delta \in K$ (note: $0 \in K$ always, so $K$ is not empty)
- Let $X_\delta$ denote the set of all valid extensions of $T[\delta]$ beyond the end time stamp of $T$
- $T[\delta]$ is the prefix of $T$ with end time stamp $\delta$

Sketch

- Observed trace
- Valid traces

Copyright © Antti Huima 2004–06. All Rights Reserved.
Details continued

- For every $T'$ in $X_\delta$, $T'$ differs from $T$ and $\Delta(T,T')$ is defined
- For every $T'$, denote by $\alpha\ {T'}|_{\Delta(T,T')}$ if not $\tau$
  - Otherwise denote by $\alpha\ {T'}|_{\Delta(T,T')}$
  - Note: $\alpha$ can not be $\tau$
- Let $D_\delta$ be the union of all $\alpha$
- $D_\delta$ lists those events on which valid extensions of $T[\delta]$ differ from $T$ first

Details continued

- Assume there exists $\delta \in K$ such that $D_\delta \subseteq \Sigma_{\text{in}}$
  - Tester failure $\rightarrow$ error
- Assume there exists $\delta \in K$ such that $D_\delta \subseteq \Sigma_{\text{out}}$
  - SUT failure $\rightarrow$ fail
- Otherwise
  - undefined $\rightarrow$ confused
Details continued

- If K is closed, we can always choose \( \delta = \max K \)
- If K is open, we must choose a \( \delta \) ”close enough” the upper bound of K
  - \((\sup K) - \epsilon \) for \( \epsilon > 0 \)

Disjointness

- \( D_\delta \subseteq \Sigma_{\text{in}} \) and \( D_\delta \subseteq \Sigma_{\text{out}} \) are disjoint conditions, because
  - \( \delta \leq \epsilon \) implies \( D_\epsilon \subseteq D_\delta \)
  - \( D_\delta \) is always non-empty
  - \( \Sigma_{\text{in}} \) and \( \Sigma_{\text{out}} \) are disjoint
Summary

► Is $T \in \text{Tr}(S)$?
  • Verdict is "pass"
► Else
  • Does there exists $\delta \in K$ such that $D_\delta \subseteq \Sigma_{\text{out}}$?
    • Verdict is "fail"
  • Otherwise, does there exists $\delta \in K$ such that $D_\delta \subseteq \Sigma_{\text{in}}$?
    • Verdict is "error"
  • Otherwise verdict is "confused"

Examples

► Spec: “System must send out X before time 2”
► Observed trace $<\emptyset, 2>$
► Verdict?
Examples

▶ Spec: “System must send out X at latest at time 2”
▶ Observed trace $<\emptyset, 2>$
▶ Verdict?

Examples

▶ Spec: “System must send out X before time 2 if it receives Y before 1”
▶ Observed trace $<\emptyset, 2>$
▶ Verdict?
Examples

▶ Spec: “System must send out X before time 2 if it receives Y before 1”
▶ Observed trace <{<Y,0.5>}, 2>
▶ Verdict?

Examples

▶ Spec: “System must send out X before time 2 if it receives Y before 1”
▶ Observed trace <{<Y,1.5>}, 2>
▶ Verdict?
Examples

- Spec: “System must receive clock signal every 1 second starting from \( t=1 \)”
- Observed trace \(<\{<\text{clock},1>\}, 1.5>\)
- Verdict?

Examples

- Spec: “System must receive clock signal every 1 second starting from \( t=1 \)”
- Observed trace \(<\{<\text{clock},1>\}, 2>\)
- Verdict?
Examples

▶ Spec: “System must receive clock signal every 1 second starting from t=1”
▶ Observed trace <{<clock,1>}, 2.5>
▶ Verdict?

Examples

▶ Spec: “Starting from t=1, on every second system must either send or receive X”
▶ Observed trace <∅, 10>
▶ Verdict?
Discussion

- These definitions are abstract and intensional
- They do not involve an algorithm
- They are given directly for trace sets making them universal
Conformance?

What does it mean that a system conforms to a specification?
- System functions as specified
- System passes all tests
- Which “all” tests?
- System passes every “test” that is “correct”
- What is “a test”? What is “a correct test”?

What is “a test”?

A test = ?
- a specific testing strategy
- a specific test execution trace
- a specific tester
- a specific tester execution

A correct test = ?
- A test execution trace with verdict ≠ ERROR
- A testing strategy or tester that “never works illegally”
  - What does this mean?
Correct testing strategies

A testing strategy $s$ is correct with respect to a specification $S$ if for any implementation $i$:

$$P[i,s,T] > 0 \Rightarrow \text{verdict}(T, S) \neq \text{ERROR}$$

Denote by $CT(S)$ the set of all correct testing strategies with respect to $S$

Synthetic correct strategies

A correct testing strategy can be (informally) constructed by the following loop:

- Guess the next action (send/wait) so that a valid trace extension will result
- Execute the chosen action, observing the actions of the SUT
- Restart loop

More on this later!

Shows that correct testing strategies exist
- Possible because of the seriality of valid set of traces
- In real life computationally intensive
Correct strategies ctd

▶ If we assume these synthetic strategies belong to the set of available testing strategies...
▶ ... then all correct and failing behaviours (at least prefixes of) of a system can be observed.

Classifying traces

▶ Let i be an implementation and S a specification
▶ For every trace T, one of the following is true:
  • There exists $s \in \text{CT}(S)$ such that $P[i, s, T] > 0$
  • There exists s, but none in CT(S), such that $P[i, s, T] > 0$
  • There does not exist any s such that $P[i, s, T] > 0$
Trace taxonomy ctd

- Furthermore, for every trace \( T \) it holds that \( \text{verdict}(T, S) \) is one of \( \text{PASS}, \text{FAIL}, \text{ERROR} \)
- We ignore ambiguous specifications \( \rightarrow \) \( \text{verdict CONFUSED} \) for now

Matrix for a trace \( T \)

<table>
<thead>
<tr>
<th>Condition</th>
<th>( \exists s \in \text{CT}(S): P[i, s, T] &gt; 0 )</th>
<th>Possible</th>
<th>FAIL</th>
<th>Not possible (def. correct strategy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \exists s: P[i, s, T] &gt; 0 ) ( \forall s': P[i, s', T] &gt; 0 ) ( \Rightarrow s' \not\in \text{CT}(S) )</td>
<td>Not possible (synthetic testers)</td>
<td>Possible</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \exists s: P[i, s, T] &gt; 0 )</td>
<td>Possible (impl. restriction)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Trace map of a general system**

CORRECT AND INCORRECT TESTERS

I: correct, producible behaviour
II: incorrect, producible
INCORRECT TESTERS ONLY
III: producible behaviour against malfunctioning environment only
NO TESTERS AT ALL
IV: correct behaviour not implemented
V: incorrect behaviour not implemented
VI: behaviour involving malfunctioning environment that has not been implemented

**Trace map of a correct system**

CORRECT AND INCORRECT TESTERS

I: correct, producible behaviour
INCORRECT TESTERS ONLY
III: producible behaviour against malfunctioning environment
NO TESTERS AT ALL
IV: correct behaviour not implemented
V: incorrect behaviour not implemented
VI: behaviour involving malfunctioning environment that has not been implemented
Correct system ctd.

- If we restrict ourselves to correct testers, then
- all behaviour that can be generated is included within the set of valid traces $\text{Tr}(S)$. [Note: assumed that system is “correct”.]

Execution against correct strategies

- Define set of execution traces
  \[ \text{ETr}(i) = \{ T \mid \exists s : P[i,s,T] > 0 \} \]
- Define now
  \[ \text{ETr}(i, S) = \{ T \mid \exists s \in \text{CT}(S) : P[i,s,T] > 0 \} \]
- Here $\text{CT}(S)$ is the set of testing strategies correct with respect to $S$
- Note $\text{ETr}(i, S) \subseteq \text{ETr}(i)$ for all $S$
Continued…

► We have now eliminated the **ERROR** verdict
► Suppose for all $s \in CT(S)$,
  \[ P[i, S, T] > 0 \text{ implies } \text{verdict}(T, S) \neq \text{FAIL} \]
► Then (assuming unambiguous specifications),
  \[ P[i, S, T] > 0 \text{ implies } \text{verdict}(T, S) = \text{PASS} \]
► Hence, $ETr(i, S) \subseteq Tr(S)$

Conclusion

► We have thus reduced the conformance of an implementation $i$ to a specification $S$ to the equation
  \[ ETr(i, S) \subseteq Tr(S) \]
► This is the underlying notion of conformance in the known theory of formal conformance testing
Conclusion ctd.

▶ Conformance = trace inclusion
  • Traces generated by implementation are included in those generated by specification
  • Incorrectly generated/out-of-specification traces must be excluded
  • Note: no explicit mention of single testing strategies above!

Implications

▶ Quantifying over all testers leads to simple trace set inclusion
▶ This trace set inclusion can be also checked for directly under suitable conditions → model checking
▶ Thus formal conformance testing = “partial model checking”