A novel method for derivation of a test with guaranteed coverage for LTS

Gromov Maxim
gromov@sibmail.com
Tomsk State University
What is testing?

• Testing — is experimenting
• There is a formal specification
• Implementation is a «black box» with known interfaces
• Test — is prescription what to apply and what to observe at every moment
• There is a formal definition, what to consider conformant
FSM and LTS

Finite State Machine
- Specification is FSM
- Conformance relation is quasi-reduction
- Implementation is supposed to behavior as an FSM

Labelled Transition System
- Specification is LTS
- Conformance relation is ioco
- Implementation is supposed to behavior as an LTS
Finite State Machine

\(<S, I, O, s_0, \lambda>\)

- \(S\) – set of states
- \(I\) and \(O\) – disjoint sets of inputs and outputs
- \(\lambda \subseteq S \times I \times O \times S\) – transition relation;
  
  \((s, i, o, s') \in \lambda\)
Conformance with FSM

\[ lm \leq sp \]

\[ \forall \sigma \in Tr(sp) \] (for any valid word of specification)

**IF** \( \sigma \in Tr(lm) \) (it's valid for implementation) **THEN**

\[ \text{in}(lm \text{ after } \sigma) \supseteq \text{in}(sp \text{ after } \sigma) \]  
(input behaviour of implementation is larger then one of the specification)

\[ \forall i \in \text{in}(sp \text{ after } \sigma) \]

\[ \text{out}(lm \text{ after } \sigma, i) \subseteq \text{out}(sp \text{ after } \sigma, i) \]  
(output behaviour of implementation is not larger then one of the specification)
Labeled Transition System

\[ <S, I, O, s_0, \lambda> \]

- \( S \) – set of states
- \( I \) and \( O \) – disjoint sets of inputs and outputs
- \( \lambda \subseteq S \times (I \cup O) \times S \) – transition relation;

\[(s, a, s') \in \lambda\]
Quiescence

\[
\begin{align*}
&s \\
&s' \\
&s''
\end{align*}
\]

\[
\begin{align*}
&s \\
&s' \quad ?b \\
&s'' \\
&s'' \quad \delta
\end{align*}
\]
Conformance with LTS

\[ \text{Im \ iocop \ Sp} \]
\[ \forall \sigma \in s\text{-traces}(Sp) \ (\text{for any valid word of specification}) \]

**IF** \( \sigma \in s\text{-traces}(\text{Im}) \) (it's valid for implementation)

**THEN**

\[ \text{in(Im after } \sigma) \supseteq \text{in(Sp after } \sigma) \] (input behaviour of implementation is **larger** then one of the specification)

\[ \text{out(Im after } \sigma) \subseteq \text{out(Sp after } \sigma) \] (output behaviour of implementation is **not larger** then one of the specification)
Fault model

**FSM**

\(<F, \leq, E>\), where

- \(F\) – specification (a state machine)
- \(\leq\) – quasi-reduction
- \(E\) – set of FSMs, describing faulty behaviour

**LTS**

\(<L, \text{iocop}, E>\), where

- \(L\) – specification (a state machine)
- \(\text{iocop}\) – iocop relation
- \(E\) – set of LTSes, describing faulty behaviour
From LTS to FSM

\[ L_1 \rightarrow F_1^\varepsilon \]
\[ L_2 \rightarrow F_2^\varepsilon \]

\[ L_1 \text{ iocop } L_2 \leftrightarrow F_1^\varepsilon \leq F_2^\varepsilon \]
If $F^\varepsilon$ passes $T^\varepsilon$ then $L$ passes $T$
If $F^\varepsilon$ fails $T^\varepsilon$ then $L$ fails $T$
Test derivation for an LTS

- Given an LTS fault model;
- Transform all LTSes to FSMs to get an FSM fault model;
- Build complete test for the FSM fault model;
- Transform FSM test to LTS test;
- Obtained test will be complete in respect to the given LTS fault model
Questions?

Thank you for your attention!!