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

Gromov M.L.

***Abstract*—The paper addresses the problem of the relationship between tests with the guaranteed fault coverage which are derived from two models of finite transition systems: LTS with inputs and outputs and input/output FSM. On one hand, there are well developed methods for deriving such finite tests against FSM while the model of input/output LTSs lacks such methods. On the other hand, almost all FSM-based methods return preset test suites and there are no good FSM-based methods for deriving adaptive tests. For deriving adaptive tests, it is worth to use LTS model where all the methods return adaptive test suites. We propose the transformation between two models and show how a complete test suite with respect to one model can be transformed into a complete test suite with respect to another model.**

I. Introduction

Testing software systems, it is often necessary to take into account potential nondeterministic behaviour, when for a given input sequence several output sequences can be produced. For example, the nondeterminism in the specification can be used for the description of implementation options and/or allowable interleaving of outputs. In this, paper we allow an implementation to be less deterministic as its specification. Testing a reactive system, two finite transition models are widely used for such test derivation. Those are Labelled Transition Systems (LTS) with inputs and outputs [1] and input/output Finite State Machines (FSM) [2]. Corresponding conformance relations are the so-called ioco relation for LTSs and the reduction relation for FSMs. However, the model of input/output LTSs lacks methods for deriving finite test suites with the guaranteed fault coverage while there are well developed FSM-based methods for deriving such finite tests. On the other hand, the need for adaptive tests for nondeterministic machines has widely been discussed in various work but almost all FSM-based methods return preset test suites and thus, for deriving adaptive tests, it is worth to use LTS model where all the methods return adaptive test suites. Here by testing we understand an experiment with an implementation, and preset test means, that all input sequences to apply and output sequences to observe are given beforehand, while adaptive test imply that the next input action depends on the previous input action reaction (so, preset test is an algorithm, which is convenient to give in the form of the same formal model in which specification is given) [3]. In this paper, we propose the transformation between two models and show how a complete finite test suite with respect to one model can be transformed into a complete finite test suite with respect to another model, that is, in fact, we propose a method how finite LTS-based test suites with the guaranteed fault coverage can be derived. This paper is organised as follows. Section II presents the notions used in the paper. Section III defines the transformation of an LTS to an FSM. In Section IV, we define a fault model for LTSs and FSMs and complete test suites with respect to such models. Subsection IV-C shows how an LTS-based complete test suite can be derived using FSM-based derivation methods. Section V concludes the paper with the discussion on the future work.

II. Definitions

In this section, we briefly remind basic definitions for a *labelled transition system* (LTS), taken from [1] and a *finite state machine* (FSM), taken from [2].

A. Labelled transition system

Definition 1: A *(finite, deterministic) Labelled Transition System* (LTS) with inputs I and outputs O is a quintuple $\langle S, I, O, \lambda, s_0\rangle$, where S is a non-empty, finite set of states with the initial state $s_0 \in S$; I and O are disjoint finite sets representing the set of input actions and output actions, respectively. We denote their union by A. The transition relation $\lambda \subseteq S \times A \times S$ defines a function, mapping subset of $S \times A$ to S.

Let $L = \langle S, I, O, \lambda, s_0\rangle$ be an LTS, and $s, s', \ldots$ range over S. Throughout this paper, we use the following conventions: for all actions a, we write $s \xrightarrow{a} s'$ iff $\langle s, a, s'\rangle \in \lambda$, and $s \not\xrightarrow{a}$ iff for all $s' \in S$ $s \xrightarrow{a} s'$ does not hold.

In **ioco**-based testing theory [1] the notion of *quiescence* is added to an LTS as follows:

Definition 2: A state $s \in S$ of an LTS $L = \langle S, I, O, \lambda, s_0\rangle$ is called *quiescent* – notation $\delta(s)$ – if for all $a \in O$, $s \not\xrightarrow{a}$.

Informally, a quiescent state is a state where LTS refuses to provide outputs. Let $\delta \notin A$ be a fresh label representing the possibility to observe quiescence; A_δ abbreviates $A \cup \{\delta\}$. Let $\sigma, \sigma', \ldots$ range over A_δ^*, actions a range over A_δ, and $S', S'', \ldots \subseteq S$. We generalise the transition relation λ to $\Lambda \subseteq S \times A_\delta^* \times S$, and write $s \stackrel{\sigma}{\Longrightarrow} s'$ iff $\langle s, \sigma, s'\rangle \in \Lambda$. We define Λ as the smallest relation satisfying the following four rules:

$$\frac{}{s \stackrel{\epsilon}{\Longrightarrow} s} \qquad \frac{s \stackrel{\sigma}{\Longrightarrow} s' \ \wedge \ s' \xrightarrow{a} s''}{s \stackrel{\sigma\cdot a}{\Longrightarrow} s''} \qquad \frac{s \stackrel{\sigma}{\Longrightarrow} s' \quad \delta(s')}{s \stackrel{\sigma\cdot\delta}{\Longrightarrow} s'}$$

Like for $\rightarrow$, we write $s \stackrel{\sigma}{\Longrightarrow}$ if $s \stackrel{\sigma}{\Longrightarrow} s'$ for some s'. For the simplicity of presentation, we introduce the following notations.

1) $[s]_\sigma \stackrel{\text{def}}{=} \{s' \in S \mid s \stackrel{\sigma}{\Longrightarrow} s'\}$; generalised: $[S']_\sigma \stackrel{\text{def}}{=} \bigcup_{s \in S'} [s]_\sigma$;
2) $\mathbf{out}(s) \stackrel{\text{def}}{=} \{a \in O \mid s \xrightarrow{a}\} \cup \{\delta \mid \delta(s)\}$; generalised: $\mathbf{out}(S') \stackrel{\text{def}}{=} \bigcup_{s \in S'} \mathbf{out}(s)$;

3) $\mathbf{in}(s) \stackrel{\text{def}}{=} \{a \in I \mid s \xrightarrow{a}\}$; generalised: $\mathbf{in}(S') \stackrel{\text{def}}{=} \bigcup_{s \in S'} \mathbf{in}(s)$;
4) *s-traces*$(s) \stackrel{\text{def}}{=} \{\sigma \in A_\delta^* \mid s \stackrel{\sigma}{\Longrightarrow}\}$;
5) *traces*$(s) \stackrel{\text{def}}{=}$ *s-traces*$(s) \cap A^*$;
6) $\mathbf{der}(s) \stackrel{\text{def}}{=} \bigcup_{\sigma \in A^*} [s]_\sigma$; generalised: $\mathbf{der}(S') \stackrel{\text{def}}{=} \bigcup_{s \in S'} \mathbf{der}(s)$;
7) $\mathbf{init}(s) \stackrel{\text{def}}{=} \{a \in A_\delta \mid s \xrightarrow{a}\}$.

The *conformance* relation **iocop** is defined as follows:

Definition 3: Let $L_1 = \langle S_1, I, O, \lambda_1, s_{01}\rangle$ and $L_2 = \langle S_2, I, O, \lambda_2, s_{02}\rangle$ be two LTSs. LTS L_1 is in **iocop** relation with L_2 – notation L_1 **iocop** L_2 – when the following holds

$$\begin{array}{l}\forall \sigma \in \textit{s-traces}(s_{02}) : \\ \quad (\text{if } s_{01} \stackrel{\sigma}{\Longrightarrow} \text{then } \mathbf{in}([s_{01}]_\sigma) \supseteq \mathbf{in}([s_{02}]_\sigma)) \wedge \\ \wedge (\mathbf{out}([s_{01}]_\sigma) \subseteq \mathbf{out}([s_{02}]_\sigma))\end{array}$$

Note, that our definition of conformance relation is different from the classical **ioco** definition [1] that requires the LTS L_1 to be input-enabled. However original **ioco** relation is not reflexive, while **iocop** is reflexive. Moreover, for an input-enabled L_1 it holds that L_1 **ioco** $L_2 \iff L_1$ **iocop** L_2.

B. Finite State Machine

Definition 4: An *(observable) Finite State Machine* (FSM) with the set of inputs I and the set of outputs O is a quintuple $\langle T, I, O, \mu, t_0\rangle$, where T is a non-empty, finite set of states with initial state $t_0 \in T$; I and O are disjoint finite sets representing the set of input actions and output actions, respectively. We denote their union by A. The transition relation $\mu \subseteq T \times I \times O \times T$ defines a function, mapping a subset of $T \times I \times O$ to T.

Let $F = \langle T, I, O, \mu, t_0\rangle$ be an FSM, and $t, t', \ldots$ range over T. Throughout this paper, we use the following conventions: for all input-output pair $\langle i, o\rangle \in I \times O$, we write $t \xrightarrow{i/o} t'$ iff $\langle t, i, o, t'\rangle \in \mu$, and $t \not\xrightarrow{i/o}$ iff for all $t' \in T$ $t \xrightarrow{i/o} t'$ does not hold.

Let $\eta, \eta', \ldots$ range over $(I \times O)^*$, input actions i range over I, output actions o range over O, and $T', T'', \ldots \subseteq T$. We generalise the transition relation μ to $\mathcal{M} \subseteq T \times (I \times O)^* \times T$, and write $t \stackrel{\eta}{\Longrightarrow} t'$ iff $\langle t, \sigma, t'\rangle \in \mathcal{M}$. We define $\mathcal{M}$ as the smallest relation satisfying the following four rules:

$$\frac{}{t \stackrel{\epsilon}{\Longrightarrow} t} \qquad \frac{t \stackrel{\eta}{\Longrightarrow} t' \ \wedge \ t' \xrightarrow{i/o} t''}{t \xrightarrow{\eta \cdot i/o} t''}$$

Similar to $\rightarrow$, we write $t \stackrel{\eta}{\Longrightarrow}$ if $t \stackrel{\eta}{\Longrightarrow} t'$ for some t'. For ease of use, we introduce the following notations.

1) $[t]_\eta \stackrel{\text{def}}{=} \{t' \in T \mid t \stackrel{\eta}{\Longrightarrow} t'\}$; generalised: $[T']_\eta \stackrel{\text{def}}{=} \bigcup_{t \in T'} [t]_\eta$;
2) $\mathbf{out}(t, i) \stackrel{\text{def}}{=} \{o \in O \mid t \xrightarrow{i/o}\}$; generalised: $\mathbf{out}(T', i) \stackrel{\text{def}}{=} \bigcup_{t \in T'} \mathbf{out}(t)$;
3) $\mathbf{in}(t) \stackrel{\text{def}}{=} \{i \in I \mid t \xrightarrow{t}\}$; generalised: $\mathbf{in}(T') \stackrel{\text{def}}{=} \bigcup_{t \in T'} \mathbf{in}(t)$;
4) *Tr*$(t) \stackrel{\text{def}}{=} \{\eta \in (I \times O)^* \mid t \stackrel{\eta}{\Longrightarrow}\}$;
5) $\mathbf{der}(t) \stackrel{\text{def}}{=} \bigcup_{\eta \in (I \times O)^*} [t]_\eta$; generalised: $\mathbf{der}(T') \stackrel{\text{def}}{=} \bigcup_{t \in T'} \mathbf{der}(t)$;
6) $\mathbf{init}(t) \stackrel{\text{def}}{=} \{\langle i, o\rangle \in I \times O \mid t \xrightarrow{i/o}\}$.

In this paper we extend the well-known quasi-reduction between FSMs, since the old notion [2] is incorrect for partial non-deterministic FSMs with non-harmonised traces.

Definition 5: Let $F_1 = \langle T_1, I, O, \mu_1, t_{01}\rangle$ and $F_2 = \langle T_2, I, O, \mu_2, t_{02}\rangle$ be two FSMs. FSM F_1 is quasi-reduction of F_2 – notation $F_1 \precsim F_2$ – when the following holds

$$\begin{array}{l}\forall \eta \in \textit{Tr}(t_{02}) : \\ \quad (\text{if } t_{01} \stackrel{\eta}{\Longrightarrow} \text{then } \mathbf{in}([t_{01}]_\eta) \supseteq \mathbf{in}([t_{02}]_\eta)) \wedge \\ \wedge (\mathbf{init}([t_{01}]_\sigma) \cap (\mathbf{in}(t_2) \times O) \subseteq \mathbf{init}([t_{02}]_\sigma))\end{array}$$

III. LTS to FSM transformation

Given LTS $L = \langle S, I, O, \lambda, s_0\rangle$, we build an FSM $F_\varepsilon^L = \langle S, I_\varepsilon, O_{\varepsilon\delta}, \mu, s_0\rangle$, where $\varepsilon_i, \varepsilon_o \notin A$ are fresh input and output actions (here and later in the paper we assume, that $I_\varepsilon \equiv I \cup \{\varepsilon_i\}$, $O_{\varepsilon\delta} \equiv O \cup \{\varepsilon_o, \delta\}$), and μ is defined by following rules:

$$\frac{s \xrightarrow{a} s' \wedge a \in I}{\langle s, a, \varepsilon_o, s'\rangle \in \mu} \qquad \frac{s \xrightarrow{a} s' \wedge a \in O}{\langle s, \varepsilon_i, a, s'\rangle \in \mu} \qquad \frac{\delta(s)}{\langle s, \varepsilon_i, \delta, s\rangle \in \mu}.$$

Note, that according to our transformation in FSM F_ε^L there are no transitions with labels $\varepsilon_i/\varepsilon_o$, i_1/i_2, o_1/o_2, i_1/o_1, o_1/i_1, where $i_1, i_2 \in I$, $o_1, o_2 \in O_\delta$.

The following theorem holds:

Theorem 1: Let $L_1 = \langle S_1, I, O, \lambda_1, s_{01}\rangle$ and $L_2 = \langle S_2, I, O, \lambda_2, s_{02}\rangle$ be two LTSs. Then L_1 **iocop** L_2 iff $F_\varepsilon^{L_1} \precsim F_\varepsilon^{L_2}$.

IV. Fault models and complete tests

A. A fault model and a complete test for an LTS

Definition 6: Given deterministic LTS $L = \langle S, I, O, \lambda, s_0\rangle$, L_δ is an LTS $\langle S, I, O_\delta, \lambda_\delta, s_0\rangle$, for which $\lambda_\delta = \lambda \cup \{\langle s, \delta, s\rangle | \delta(s)\}$.
LTS L_δ differs from L only by added loops with labels δ for quiescent states. It is known [1], that *s-traces*$(L) =$ *traces*(L_δ).

Definition 7: An LTS *test case* is a finite LTS $C = \langle V, I, O_\delta, \lambda_C, v_0\rangle$, which satisfies the following conditions:

1) $V \supseteq \{\texttt{pass}, \texttt{fail}\}$.
2) Transition graph of the LTS C is acyclic.
3) For each $v \in V \setminus \{\texttt{pass}, \texttt{fail}\}$ either $\mathbf{init}(v) = O_\delta$, or $\mathbf{init}(v) = \{i\}$, $i \in I$ holds.
4) $\mathbf{init}(\texttt{pass}) = \mathbf{init}(\texttt{fail}) = \varnothing$.

A finite set of test cases we call *a test suite*.
Note, that in practice we are interested only in finite test suites and thus consider only this kind of tests.

Definition 8: An LTS $L = \langle S, I, O, \lambda, s_0\rangle$ *passes a test case* $C = \langle V, I, O_\delta, \lambda_C, v_0\rangle$ iff none of sequences $\sigma \in$ *s-traces*(L) takes LTS C from the initial state to the state `fail`, that is $(S \times \{\texttt{fail}\}) \cap \mathbf{der}(L_\delta \cap C) = \varnothing$.

Definition 9: An LTS *fault model* is a triple $\langle L_M,$ **iocop** $, E_L\rangle$, where L_M is the LTS-specification (or simply specification), E_L is a set of LTSs with input alphabet I and output alphabet O, which called the fault domain, such that for any LTS $L' \in E_L$ holds: for each $\sigma \in$ *s-traces*(L_M) if $L' \stackrel{\sigma}{\Longrightarrow}$, then $\mathbf{in}([L']_\sigma) \supseteq \mathbf{in}([L_M]_\sigma)$.

Definition 10: A test suite $\mathcal{T} = \{C_1, \ldots, C_n\}$ is *complete* with respect to the fault model $\langle L_M,\ \mathbf{iocop}\ , E_L\rangle$ iff for each LTS $L' \in E_L$ that is not in the $\mathbf{iocop}$ relation with L_M there exists such a test case $C_k \in \mathcal{T}$ that L' does not pass (fails) C_k.

B. Fault models and complete test for an FSM

Definition 11: An FSM *test case* is FSM $C = \langle V, I, O, \mu_C, v_0\rangle$, which satisfies following conditions:

1) $V \supseteq \{\mathtt{pass}, \mathtt{fail}\}$.
2) Transition graph of the FSM C is acyclic.
3) For each $v \in V \setminus \{\mathtt{pass}, \mathtt{fail}\}$ holds $|\mathbf{in}(v)| = 1$ and $\mathbf{out}(v, i) = O$, where $\mathbf{in}(v) = \{i\}$.
4) $\mathbf{init}(\mathtt{pass}) = \mathbf{init}(\mathtt{fail}) = \varnothing$.

A finite set of test cases we call *a test suite*.

Definition 12: An FSM $F = \langle T, I, O, \mu, t_0\rangle$ *passes a test case* $C = \langle V, I, O_\delta, \mu_C, v_0\rangle$ iff none of input-output sequences $\eta \in Tr(F)$ takes FSM C from the initial state to the state $\mathtt{fail}$, that is $(T \times \{\mathtt{fail}\}) \cap \mathbf{der}(F \cap C) = \varnothing$.

Definition 13: An FSM *fault model* is a triple $\langle F_M, \precsim, E_F\rangle$, where F_M is the FSM-specification (or simply specification), E_F is a set of FSMs with input alphabet I and output alphabet O, called the fault domain, such that for any FSM $F' \in E_F$ holds: for each $\eta \in Tr(F_M)$ if $F' \stackrel{\eta}{\Longrightarrow}$, then $\mathbf{in}([F']_\eta) \supseteq \mathbf{in}([F_M]_\eta)$.

Definition 14: A test suite $\mathcal{T} = \{C_1, \ldots, C_n\}$ is *a complete* with respect to the fault model $\langle F_M, \precsim, E_F\rangle$ iff for each FSM $F' \in E_F$ that is not in the relation $\precsim$ with F_M there exists such a test case $C_k \in \mathcal{T}$ that F' fails C_k.

C. The complete test derivation for an LTS, based on a complete test for an FSM

To the best of our knowledge, there are no methods for the complete finite test derivation for an LTS fault model $\langle L, \mathbf{ioco}, E_L\rangle$, while for the FSM fault model $\langle F, \precsim, E_F\rangle$ such methods are well-known (for example [2]).

We now show, how results of Section III can be used to derive complete finite test for an LTS fault model $\langle L, \mathbf{ioco}, E_L\rangle$.

Given LTS fault model $\mathcal{M}_L = \langle L, \mathbf{ioco}, E_L\rangle$, where $E_L = \{L_1, L_2, \ldots\}$, apply the transformation, described in Section III, obtain an FSM fault model $\mathcal{M}_F = \langle F_\varepsilon^L, \precsim, E_F\rangle$, where $E_F = \{F_\varepsilon^{L_1}, F_\varepsilon^{L_2}, \ldots\}$ and derive a complete finite test suite $\mathcal{T}_F$ for $\mathcal{M}_F$. For each test case $C_{F_\varepsilon} \equiv \langle V, I_\varepsilon, O_{\varepsilon\delta}, \mu, v_0\rangle$ from $\mathcal{T}_F$ we build an LTS $\nabla(C_{F_\varepsilon}) = \langle V, I, O_\delta, \lambda, v_0\rangle$, using the following rules for λ:

$$\frac{\mathbf{in}(v) \equiv \{i\} \subseteq I \ \wedge\ v \xrightarrow{i/\varepsilon_o} v'}{\langle v, i, v'\rangle \in \lambda}$$

$$\frac{\mathbf{in}(v) = \{\varepsilon_i\} \ \wedge\ v \xrightarrow{\varepsilon_i/o} v' \ \wedge\ o \neq \varepsilon_o}{\langle v, o, v'\rangle \in \lambda}.$$

It is possible to show, that $\nabla(C_{F_\varepsilon})$ is an LTS test case, and moreover the following statement holds:

Theorem 2: Given an LTS fault model $\mathcal{M}_L = \langle L, \mathbf{ioco}, E_L\rangle$, where $E_L = \{L_1, L_2, \ldots\}$, let $\mathcal{T}_F = \{C_1, \ldots, C_n\}$ be a complete test suite for the FSM fault model $\mathcal{M}_F = \langle F_\varepsilon^L, \precsim, E_F\rangle$, where $E_F = \{F_\varepsilon^{L_1}, F_\varepsilon^{L_2}, \ldots\}$. Then $\nabla(\mathcal{T}_F) = \{\nabla(C_1), \ldots, \nabla(C_n)\}$ is a complete test for the LTS fault model $\mathcal{M}_L$.

V. Conclusion and future work

In this paper, we have shown how, using an FSM fault model and complete finite test for, complete finite test for an LTS fault model. The converse seems also to be possible: there can be define such a transformation of an FSM to an LTS and an LTS test case to an FSM test case, that using those and some test suite for an LTS fault model we can get a test suite for an original FSM fault model. This is one of the directions of our future work.

Another part of our future work is application of obtained results to the general case of non-deterministic LTSs and unobservable FSMs.

References

[1] J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software—Concepts and Tools*, 17(3):103–120, 1996.

[2] A. Petrenko and N. Yevtushenko. Conformance tests as checking experiments for partial nondeterministic fsm. In *Proceedings of the 5th International Workshop on Formal Approaches to Testing of Software (Fates 2005)*, volume 3997 of *LNCS*, pages 118–133, 2005.

[3] R. Alur, C. Courcoubetis, and M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines. In *27th ACM Symp.on Theory of Comp.*, pages 363–372, 1995.