Fault Models for Embedded Systems*
(Extended Abstract)

Jens Chr. Godskesen**

The IT-University in Copenhagen
DK–2400 NV, Denmark
jcg@itu.dk

Abstract. In this paper we present the notion of an *input fault model* for embedded systems and outline how to obtain *minimal complete* test suites for a fault model. The system software is expected to be *embedded* and specified as a finite state machine.

1 Introduction

Testing a system correct with respect to a specification is often referred to as *conformance testing*. The goal is to derive, hopefully automatically a set of tests that may help to ensure the conformance of the system against the specification. The tests should at least enjoy the property that if the system does not pass all the tests then it is not in conformance with its specification.

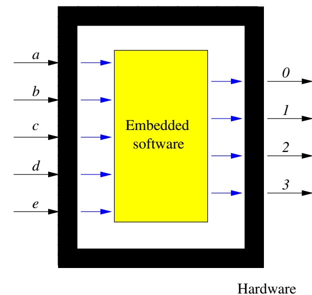

Fig. 1. A conceptualization of embedded systems.

For finite state machines (FSM's) it has been shown ([1]) that conformance between a specification S and its implementation I can be tested by a test suite that is polynomial in the size of the number of states n and input symbols in S provided I is a FSM with n states. Unfortunately, if I has more than n states the size of the test suite will be exponential. For industrial sized examples it may often be that S contains millions of states and hundreds of input symbols so even if S and I are of equal size the conformance testing problem is intractable.

In this paper we put forward an alternative theory for testing *embedded systems*. Embedded systems are dedicated systems like mobile phones, hi-fi equipment,

* The full version of this paper is obtainable from http://www.itu.dk/~jcg

** Supported by a grant from the Danish Technical Research Council.

L. Pierre and T. Kropf (Eds.): CHARME'99, LNCS 1703, pp. 354–359, 1999.
© Springer-Verlag Berlin Heidelberg 1999

remote controls etc., where the software is embedded. Conceptually an embedded system may be regarded as the *composition* of the two system components: the embedded software and the hardware as depicted in figure 1. The inputs from the system environment to the software have to pass through the hardware towards the software via *connections* (the unlabelled arrow arrows) and likewise the outputs generated by the software have to pass via connections through the hardware. Connections are considered to be purely abstract notions, they may have no physical counterpart. Each input and output has its own connection.

Exploiting the ability to automatically generate code from specifications and assuming compilers to be correct it would be reasonable to expect the generated software to be correct with respect to its specification. The hardware we shall regard as a black box interfacing the embedded software through the connections. As a consequence we shall not be able to refer directly to hardware errors. Given these two assumptions a fault may indirectly be modeled as erroneous connections. Therefore, what has to be tested in order to establish the system correctness is *the non-existence of faults manifested as errors in the connections*. Hence the focus is on testing the composition of the two system components instead of testing the behaviour of the whole system.

In figure 1 a fault could be that the connection for input b is missing. For a mobile phone this may correspond to some button being disconnected such that the software will never receive the input, making the pressing of the button cause no effect. In order to be able to handle faults we introduce the notion of a *fault model* and we outline a framework for testing embedded systems by means of such models. Fault models in a testing framework has appeared elsewhere in the literature, see e.g. [2]. However, although similar their notion is in the setting of conformance testing.

2 Finite State Machines

A deterministic *finite state machine* is a five tuple $M = (\Sigma, \varepsilon, \Omega, \tau, s_M)$ where Σ is a finite set of states, ε a finite set of inputs, Ω a finite set of outputs, and τ a transition function, $\tau : \Sigma \times \varepsilon \to \Sigma \times \mathcal{P}(\Omega)$. $s_M \in \Sigma$ is the *initial state*. We let s range over Σ, α over ε, ω over Ω, ι over $\mathcal{P}(\varepsilon)$, o over $\mathcal{P}(\Omega)$, and σ over ε^*. [1] If σ' is a prefix of σ we write $\sigma' \preceq \sigma$. $|\sigma|$ denotes the length of σ. ϵ is the empty sequence. $\sigma \downarrow \iota$ denotes an abstraction of σ where all occurrences of $\alpha \in \iota$ are removed, that is $\sigma \downarrow \iota$ may be inductively defined by $\sigma \downarrow \iota = \epsilon$ if $\sigma = \epsilon$, $\sigma \downarrow \iota = \sigma' \downarrow \iota$ if $\sigma = \sigma'\alpha$ and $\alpha \in \iota$, and $\sigma \downarrow \iota = (\sigma' \downarrow \iota)\alpha$ if $\sigma = \sigma'\alpha$ and $\alpha \notin \iota$.

We write $s \xrightarrow{\alpha/o} s'$ and $s \xrightarrow{\alpha} s'$ if $\tau(s, \alpha) = (s', o)$. If $s_i \xrightarrow{\alpha_i} s_{i+1}$ for $i = 1, \ldots, n$ we write $s_1 \xrightarrow{\sigma} s_{n+1}$ where $\sigma = \alpha_1 \ldots \alpha_n$. We shall regard s as a function

[1] $\mathcal{P}$ is the power set constructor.

$s : \varepsilon^* \to \mathcal{P}(\Omega)$ defined by $s(\epsilon) = \emptyset$ and $s(\sigma\alpha) = o$ if $s \xrightarrow{\sigma} s'$ and $s' \xrightarrow{\alpha/o} s''$. We shall write $M(\sigma)$ for $s_M(\sigma)$. If for all σ, $s(\sigma) = s'(\sigma)$ we write $s = s'$. If $s_M = s_{M'}$ we write $M = M'$. If not $s = s'$ ($M = M'$) we write $s \neq s'$ ($M \neq M'$).

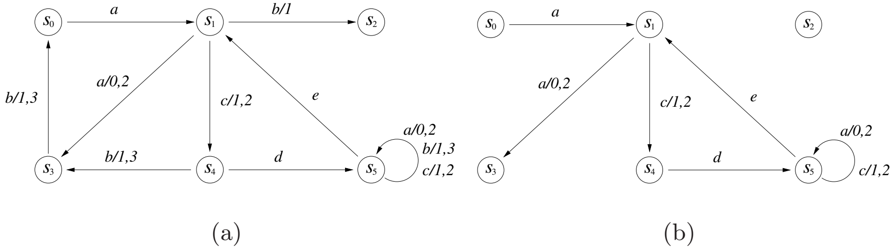

Fig. 2. (a) A finite state machine. (b) The FSM in (a) with input b removed. [2]

The machine $M[\iota]$ denotes M where any transition $s \xrightarrow{\alpha/o} s'$ in M is replaced by $s \xrightarrow{\alpha/\emptyset} s$ whenever $\alpha \in \iota$. Letting M denote the machine in figure 2(a), $M[\{b\}]$ is the machine in figure 2(b). For any set of inputs $\{\alpha_1, \ldots, \alpha_n\}$ we shall adopt the convention of writing $M[\alpha_1, \ldots, \alpha_n]$ instead of $M[\{\alpha_1, \ldots, \alpha_n\}]$.

M is α-*active* if $M(\sigma\alpha) \neq \emptyset$ for some σ. M is ι-active if it is α-active for some $\alpha \in \iota$. The machine in figure 2(b) is not b-active whereas the one in figure 2(a) is. M is α-*distinguishable* if $M(\sigma_1\alpha\sigma_2) \neq M(\sigma_1\sigma_2)$ for some σ_1 and σ_2. M is ι-distinguishable if it is α-distinguishable for some $\alpha \in \iota$. M is ι-*sensitive* if it is ι-active or ι-distinguishable. The FSM in figure 2 are e-distinguishable because $M(acdeaa) \neq M(acdaa)$, hence they are e-sensitive although not e-active.

Lemma 1. *$M \neq M[\iota]$ if and only if M is ι-sensitive.*

3 Fault Models

The purpose of a fault model is to capture the faults that may occur among the connections in an embedded system. For instance, we would like that a fault corresponding to the disconnection of the b-input in figure 1 can be handled by a fault model. In general we let any M' where $M \neq M'$ be a fault relative to M, and we define a *fault model* for M as a set of machines $\mathcal{M}^M$ such that for all $M' \in \mathcal{M}^M$, $M \neq M'$.

[2] Empty outputs and transitions to the same state with empty output are left out.

The fault that a set of inputs ι is not connected can be modeled by $M[\iota]$. Let $\mathcal{I} \subseteq \mathcal{P}(\mathcal{P}(\varepsilon)/\emptyset)$. Then, if M is ι-sensitive for any $\iota \in \mathcal{I}$, the fault model for unconnected input for M is defined by $\mathcal{M}^M_{\mathcal{I}} = \{M[\iota]\iota \in \mathcal{I}\}$. Intuitively, $\mathcal{M}^M_{\mathcal{I}}$ represents any one implementation that for some $\iota \in \mathcal{I}$ contains the disrupted input connections in ι, but otherwise behaves as expected by M. Letting M denote the FSM in figure 2(a) then the fault model below is an input fault model for M. It contains beyond the multi-fault $M[b, d]$ the single-faults $M[c]$, $M[d]$, and $M[e]$.

$$\mathcal{M}^M_{\mathcal{I}} = \mathcal{M}^M_{\{\{c\},\{d\},\{e\},\{b,d\}\}} \tag{1}$$

4 Tests

A *test* is a finite sequence of input events $\sigma \in \varepsilon^*$. A *test suite* $T \subseteq \varepsilon^*$ is a finite set of tests. The application of a test and a test suite respectively to M' with respect to some reference M is defined by

$$apply_M(M', \sigma) = \begin{cases} pass \text{ if } \forall \sigma' \preceq \sigma.\ M(\sigma') = M'(\sigma') \\ fail \text{ otherwise} \end{cases}$$

$$apply_M(M', T) = \begin{cases} pass \text{ if } \forall \sigma \in T.\ apply_M(M', \sigma) = pass \\ fail \text{ otherwise} \end{cases}$$

We say that a test suite T is *sound* for a fault model $\mathcal{M}^M$ if for any $\sigma \in T$ there exists some $M' \in \mathcal{M}^M$ such that $apply_M(M', \sigma) = fail$. A test suite T is *exhaustive* for $\mathcal{M}^M$ if for all $M' \in \mathcal{M}^M$, $apply_M(M', T) = fail$. Finally, we say that a test suite T is *complete* for $\mathcal{M}^M$ if it is sound and exhaustive for $\mathcal{M}^M$.

Given a set of tests T we let $max(T) = \{\sigma \in T \forall \sigma' \in T.\ |\sigma| \geq |\sigma'|\}$ and $min(T) = \{\sigma \in T \forall \sigma' \in T.\ |\sigma| \leq |\sigma'|\}$. For a family of set of tests $\langle T_i \rangle_{i \in I}$ for some index set I we write $T \in \langle T_i \rangle_{i \in I}$ whenever $T = \{\sigma_i i \in I\}$ where $\sigma_i \in T_i$ for all $i \in I$. When it is clear from the context we let $\langle T_i \rangle_{i \in I}$ denote the set of set of tests $\{TT \in \langle T_i \rangle_{i \in I}\}$.

In order to obtain complete test suites for a fault model we want to obtain a set of tests for each fault in the model that characterizes the fault. A fault $M[\iota]$ can be characterized by $T^M_\iota = \bigcup_{\alpha \in \iota} \{\sigma\alpha M(\sigma\alpha) \neq \emptyset\} \cup \bigcup_{\alpha \notin \iota} \{\sigma\alpha M(\sigma\alpha) \neq M(\sigma\alpha \downarrow \iota)\}$. Letting M be the FSM in figure 2(a) it is obvious that $ac \in T^M_{\{c\}}$, $acda \in T^M_{\{d\}}$, $acdeaa \in T^M_{\{e\}}$, and $ab \in T^M_{\{b,d\}}$.

Lemma 2. *$apply_M(M[\iota], \sigma) = fail$ if and only if $\exists \sigma' \in T^M_\iota.\ \sigma' \preceq \sigma$.*

Given a family of characterizing sets of tests, one for each fault in a fault model $\mathcal{M}^M_{\mathcal{I}}$, a complete test suite can be defined.

Theorem 1. *Let $\mathcal{M}_{\mathcal{I}}^{M}$ be a fault model. Then any $T \in \langle T_\iota^M \rangle_{\iota \in \mathcal{I}}$ is complete for $\mathcal{M}_{\mathcal{I}}^{M}$.*

No complete test suite for $\mathcal{M}_{\mathcal{I}}^{M}$ needs to contain more tests than the number of member in the family $\langle T_\iota^M \rangle_{\iota \in \mathcal{I}}$. Letting $\mathcal{M}_{\mathcal{I}}^{M}$ be the input fault model defined by equation 1, the test suite $\{ac, acda, acdeaa, ab\}$ is complete for $\mathcal{M}_{\mathcal{I}}^{M}$ because $ac \in T_{\{c\}}^M$, $acda \in T_{\{d\}}^M$, $acdeaa \in T_{\{e\}}^M$, and $ab \in T_{\{b,d\}}^M$. However, since $acda \in T_{\{b,d\}}^M \cap T_{\{d\}}^M$ it turns out that also $\{ac, acda, acdeaa\}$ is complete for $\mathcal{M}_{\mathcal{I}}^{M}$.

5 Minimal Complete Test Suites

A test suite T is *minimal complete* for $\mathcal{M}_{\mathcal{I}}^{M}$ if $T \in \langle min(T_\iota) \rangle_{\iota \in \mathcal{I}'}$ for some $\mathcal{I}' \subseteq \mathcal{I}$, if T is complete for $\mathcal{M}_{\mathcal{I}}^{M}$, and if no $T' \subset T$ is complete for $\mathcal{M}_{\mathcal{I}}^{M}$.

For any family of set of tests $\langle T_i \rangle_{i \in I}$ we write $T \in \mu \langle T_i \rangle_{i \in I}$ if $T \in \langle T_i \rangle_{i \in I}$ and

$$\forall \sigma \in T.\, \exists i \in I.\, T \cap min(T_i) = \{\sigma\} \tag{2}$$

When it is clear from the context we let $\mu \langle T_i \rangle_{i \in I}$ denote the set of tests $\{T T \in \mu \langle T_i \rangle_{i \in I}\}$. For a family of set of tests $\langle T_i \rangle_{i \in I}$ and a set of tests $T \in \langle min(T_i) \rangle_{i \in I}$ we let $\mu_{\langle T_i \rangle_{i \in I}}(T)$ denote a subset of T such that $\mu_{\langle T_i \rangle_{i \in I}}(T) \in \mu \langle T_i \rangle_{i \in I}$.[3] Letting M be the FSM in figure 2(a) it can be shown that $\{ac, acda, acdeaa, ab\}$ belongs to $\mu \langle T_\iota^M \rangle_{\iota \in \mathcal{I}}$ where $\mathcal{I} = \{\{c\}, \{d\}, \{e\}, \{b, d\}\}$.

Define for any $\langle T_i \rangle_{i \in I}$ the function $\Phi_{\langle T_i \rangle_{i \in I}} : \langle min(T_i) \rangle_{i \in I} \to \mathcal{P}(\varepsilon^*)$ by

$$\Phi_{\langle T_i \rangle_{i \in I}}(T) = \begin{cases} \emptyset & \text{if } T = \emptyset \\ T' \cup \Phi_{\langle T_i \rangle_{i \in I''}}(T'') & \text{otherwise} \end{cases}$$

where $T' = \mu_{\langle T_i \rangle_{i \in I'}}(max(T))$ for $I' = \{i \in I \exists \sigma \in max(T).\, \sigma \in T_i\}$, and where $T'' = \{\sigma_i \in Ti \in I''\}$ for $I'' = \{i \in I \forall \sigma \in T'.\, \forall \sigma' \preceq \sigma.\, \sigma' \notin T_i\}$, whenever $T = \{\sigma_i i \in I\}$.

The functions may as results give minimal complete test suites.

Theorem 2. *Whenever $\mathcal{M}_{\mathcal{I}}^{M}$ is a fault model and $T \in \langle min(T_\iota^M) \rangle_{\iota \in \mathcal{I}}$. Then $\Phi_{\langle T_\iota^M \rangle_{\iota \in \mathcal{I}}}(T)$ is minimal complete for $\mathcal{M}_{\mathcal{I}}^{M}$.*

Intuitively, a minimal complete test suite for a fault model is constructed by first selecting a set of tests T containing a smallest test for each fault in the fault

[3] $\mu_{\langle T_i \rangle_{i \in I}}$ can be considered a function $\mu_{\langle T_i \rangle_{i \in I}} : \langle min(T_i) \rangle_{i \in I} \to \mu \langle T_i \rangle_{i \in I}$ because the resulting set of tests can be defined by keeping all tests in the argument that satisfy equation 2 and thereafter successively removing tests from the argument as needed due to some lexicographical ordering of the elements in ε^* until the definition of $\mu \langle T_i \rangle_{i \in I}$ is satisfied. This also shows the existence of tests in $\mu \langle T_i \rangle_{i \in I}$ if $I \neq \emptyset$.

model. The test for a fault is selected from the set of tests that characterizes the fault. Secondly, all the longest tests in T are collected and among those some tests may be eliminated such that equation 2 is satisfied. This gives as a result T' which will be contained in the resulting test suite. All faults in the model that may be detected by at least one of the remaining largest tests are disregarded and not considered further in the computation. The process is repeated for the faults still to be considered until no fault in the model needs consideration. As an example, letting M be as defined in figure 2, it can be shown that $\Phi_{\langle T_\iota^M \rangle_{\iota \in \mathcal{I}}}(\{ac, acda, acdeaa, ab\}) = \{acdeaa\}$ where $\mathcal{I} = \{\{c\}, \{d\}, \{e\}, \{b, d\}\}$.

In this paper we have put forward an approach for test generation of embedded systems with software that is modeled as a deterministic FSM. The approach differs from the traditional conformance testing approaches in that the focus is on testing the composition of the embedded software and the hardware. That is, it is the interface (as defined in section 1) between the hardware and the software that is to be tested.

Currently we are involved in ongoing work where algorithms for generating minimal complete test suites based on fault models are developed.

References

1. T.S Chow. Testing Software desing modeis by finite-state machines. *IEEE Transactions on Software Engeneering,* 4(3):178-187, 1978. 354
2. A. Petrenko, N. Yevtushenko, and G. v. Buchmann. Fault modeis für testing in context. In *Proceedings of the First Joined International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing and Verification (FORTE/PSTV'96),* pages 163-178, University of Kaiserslautern, Department of Informatics, October 1996. Chapman & Hall. 355