Generating Conformance Tests for Nondeterministic Protocol Machines

Luo Gang (罗钢)  
Department d'IRO, University de Montreal, C.P. 6128, Succ. A, Montreal, P.Q., H3C 3J7, Canada
Received November 24, 1992.

Abstract

We present a method of generating test cases from the software specifications which are modeled by nondeterministic finite state machines. It is applicable to both nondeterministic and deterministic finite state machines. When applied to deterministic machines, this method yields usually smaller test suites with full fault coverage than the existing methods that also assure full fault coverage. In particular, the proposed method can be used to test the control portion of software specified in the formal specification languages SDL or ESTELLE.

Keywords: Automata theory, ESTELLE, finite state machines, nondeterministic finite state machines, protocol conformance testing, protocol engineering, SDL and software testing.

1 Introduction

The testing phase represents a large effort within the common development cycle. In the area of communication software, systematic approaches have been developed for protocol conformance testing [1, 2], and the selection of appropriate test suites [3-7]. These approaches can produce significant economic benefits [8, 9]. A considerable amount of work has been done in the area of test generation for the software modeled by deterministic finite state machines (FSMs) [3, 5, 7, 10-14]. However, little work has been done to generate test cases for nondeterministic finite state machines (NFSMs). Nondeterminism is an important and useful feature of the formal specification languages for communication software. For example, both LOTOS [15] and ESTELLE [16] have a nondeterministic nature, and the CCITT SDL [17] will have it too [18]. Practical need for testing communication protocols specified by nondeterministic models is reported in [19].

Some work on test generation for nondeterministic models has been done in the context of basic LOTOS [4, 20] and finite labeled transition systems [21, 22]. However, these methods are not applicable to testing NFSMs where every transition is associated with an input/output pair. Several methods of generating test cases for NFSMs have been reported [23-25], which are all based on the generalization of unique I/O sequence [13]. However, even when these methods are applied to FSMs, a specific class of NFSMs, they still cannot guarantee full fault coverage, while full fault coverage for FSMs can be assured by many other methods. The reason is the same as pointed out in [12]. Therefore, they have limited fault detection power.

We present in this paper a test generation method for NFSMs. The motivation of our work is to generate tests for the control portion of the software written in SDL or ESTELLE.

1 e-mail: luo@iro.umontreal.ca Fax: (514)343-5834
By neglecting parameters, an SDL process or an ESTELLE model can be abstracted to an NFSM. Test cases are then developed from the NFSM.

In Section 2, we first define NFSMs and related notations. Then, a conformance relation between implementations and specifications, called trace-equivalence, is given for NFSMs. It is presented in the context of the black box testing strategy under which implementations are viewed as black boxes.

Guided by the given conformance relation, we present, in Section 3, a method for generating test cases from NFSMs. Our method is based on extending the state identification approach used for deterministic FSMs\cite{3,7,10,13} to NFSMs. We first transform an NFSM to an equivalent one that has a lower degree of nondeterminism, called observable NFSM (ONFSM). An ONFSM has a property that a state and an input/output pair uniquely determine the next state, while a state and an input alone do not necessarily determine a unique next state and an output. Test cases are then generated from the resulting ONFSMs by an approach generalized from the method given in \cite{10} which is applicable only to finite state machines.

We conclude by discussing the application of the method to generate test cases from SDL specifications.

2 Notations and Abstract Testing Framework

2.1 Nondeterministic Finite State Machines

We first present formally the definition of NFSMs, which is similar to the one given in \cite{26}. For the convenience of presentation, we define additional notations for NFSMs using an approach similar to that for labelled transition systems\cite{20-22}, and we also define a specific class of NFSMs, Observable NFSMs, which is a useful concept for test generation.

Definition 1 (Nondeterministic Finite State Machine). A nondeterministic Finite State Machine (NFSM) is a 5-tuple \((S_t, L_i, L_o, h, S_o)\), where:

1. \(S_t\) is a finite set of states, \(S_t = \{S_0, S_1, \ldots, S_{n-1}\}\).
2. \(L_i\) is a finite set of inputs.
3. \(L_o\) is a finite set of outputs.
4. \(h\) is a behavior function.

\[ h : S_t \times L_i \rightarrow \text{powerset}(S_t \times L_o) / \{\emptyset\} \]

where \(\emptyset\) denotes the empty set. Let \(P, Q \in S_t\), \(a \in L_i\) and \(b \in L_o\). We write \(P - a/b \rightarrow Q\) to denote \((Q, B) \in h(P, a)\); \(P - a/b \rightarrow Q\) is also called a transition from \(P\) to \(Q\) with label \(a/b\).

5. \(S_0\) is the initial state which is in \(S_t\).

We also assume that a “reliable” reset input \(r\) is available in any implementation of an NFSM such that upon receiving \(r\) in any state it returns to the initial state.

According to the above definition, the NFSM is completely defined, thus for every input and every state, the NFSM has at least one transition with this input. We assume that \(L_o\) may include a special symbol \(\lambda\) which represents an empty output. In testing, whether an input causes a transition with an empty output may be determined by waiting a certain period of time; if there is no output during that period, the transition invoked by the input is considered to produce an empty output \(\lambda\). We do not include spontaneous transitions (or called internal actions) in our model, since the NFSMs that allow spontaneous transitions can be modeled by equivalent (“trace-equivalent” in Subsection 2.2) NFSMs without spontaneous transitions, using an approach similar to that in \cite{27}.