Conformance testing of protocols specified as labeled transition systems*

P. V. Koppol and K. C. Tai

Department of Computer Science

North Carolina State University

Raleigh, North Carolina 27607-8206, USA

Tel: (919) 515 7146, Fax: (919) 515 7896

e-mail: gr-kvp@aristotle.csc.ncsu.edu, kct@csc.ncsu.edu

Abstract

In this paper, conformance testing of protocols specified as sets of labeled transition systems (LTSs) is considered. LTSs serve as the semantic model for a number of languages such as CCS, CSP and LOTOS. A straightforward approach to conformance testing of such a protocol is to generate the LTS corresponding to the global behaviour of the set of component LTSs and to apply known test generation procedures to the global LTS. In general, construction of the global behaviour is difficult due to the state explosion problem. Also, testing is complicated due to internal (nonobservable) events and nondeterminism. In this paper, we propose a novel approach to alleviate state explosion in the construction of the global LTS. We also propose how the reduced LTS is used in conformance testing.

Keywords

Protocol conformance testing, adaptive testing, labeled transition systems, reachability analysis

*This work was supported in part by US National Science Foundation under grant CCR-9320992

1 INTRODUCTION

Conformance testing of communication protocols has been an active area of research for almost two decades. Given a protocol specification S and an implementation I (referred to as the **implementation under test** or **IUT**), conformance testing is to show by testing, while treating I as a black box, that I conforms to S . Test generation is therefore based on the specification S . Traditional methods for test generation assume that the specification S is in the form of a strongly connected, minimal and deterministic finite state machine (FSM). Some of these techniques also require the FSM to be completely specified. The most widely studied formal methods based on these assumptions are the T-method, D-method, W-method and UIO method (Sidhu and Leung, 1989). More recently, test generation for protocols specified as labeled transition systems (LTSs) has received much attention (Arkko, 1994) (Cavalli, Favreau and Phalippou, 1994) (Cavalli and Kim, 1992) (Cavalli et al., 1993) (Cavalli, Kim and Maigron, 1994). LTSs serve as a semantic model for the standardized FDT LOTOS and other languages such as CCS (Milner, 1989) and CSP (Hoare, 1985). The semantics of the other standardized FDTs, Estelle and SDL, can also be partly expressed in LTSs (Tretmans, 1994). Details on standardized FDTs may be found in (Turner, 1993) and references therein.

Test generation methods for LTSs may be broadly classified into two categories (Cavalli, Favreau and Phalippou, 1994). The first category corresponds to the testing theory presented in (Brinksma et al., 1990). These methods are based on the coverage of all traces (Arkko, 1994). The second category consists of those methods that are built upon the traditional test generation methods such as UIO method. Methods in the first category have certain practical limitations as has been pointed out in (Cavalli, Favreau and Phalippou, 1994) (Arkko, 1994) and hence are not considered any further in this paper. Methods in the second category assume that the specification is in the form of a single (monolithic) LTS or that such an LTS can be obtained from the protocol specification (Arkko, 1994) (Cavalli and Kim, 1992) (Cavalli et al., 1993) (Cavalli, Kim and Maigron, 1994) (Fujiwara and Bochmann, 1992). In this paper, we adopt the terminology of (Cavalli, Favreau and Phalippou, 1994) and refer to the methods in the second category as **checking experiments based methods**.

In this paper, protocols specified as sets of (communicating) LTSs are considered. Examples of such specifications include CCS and CSP specifications involving parallel composition and constraint oriented LOTOS specifications. In general, the following three-step process may be used for test generation based on such specifications. First, derive the global LTS corresponding to the set of component LTSs (e.g. using expansion law in CCS). Second, transform the global LTS into a form such that a test generation technique may be applied, and third, apply a known test generation technique. Note that once the first and second steps of this general procedure are completed, this procedure coincides with the checking experiments based approaches. However, the following problems need to be addressed: 1) state space explosion during global LTS construction, 2) preserving the expected behaviour of the implementation while performing LTS transformation, and 3) assuring the executability of the generated test cases. Executability becomes a problem due to the lack of control over the nondeterminism in the IUT. Another important concern is that the length of the generated test cases could be exponential in the number of component LTSs (Lee et al, 1993).

To cope with the above problems, an adaptive testing approach was presented in (Lee

et al, 1993) wherein the need for global LTS generation is obviated and testing is based on an adaptive guided random walk. However, due to the nonexistence of the global LTS, the fault detection ability of the approach is not clear. In this paper, we propose an algorithm for the construction of a reduced global LTS, while alleviating state space explosion, such that the reduction preserves the expected behaviour of the implementation and all necessary information that will enhance the fault detection capability of the adaptive testing approach. We show how to construct testers and also identify testability issues that need to be considered in the context of adaptive testing.

The organization of this paper is as follows. Section 2 gives basic definitions. Section 3 provides the background related to checking experiments based conformance testing and the adaptive testing strategy presented in (Lee et al, 1993). It also discusses the motivation for the results of this paper. Section 4 presents an algorithm that, for a given set L of LTSs, incrementally composes and reduces subsets of L and finally produces a reduced LTS whose behavior is equivalent to the global behaviour of L . Section 5 addresses the problem of testing based on the reduced global LTS and addresses issues related to tester construction and testability. Section 6 discusses related work and section 7 concludes this paper with a discussion on future work.

2 PRELIMINARIES

In this paper, we consider communication protocols specified as sets of communicating processes with synchronous communication and direct naming. Each process is modeled as a rooted labeled transition system (LTS) which is defined as a quadruple $\langle Q, E, \rightarrow, q_0 \rangle$, where Q is a finite set of states, E is a set of events (including a special event τ), $\rightarrow \subseteq Q \times E \times Q$ is the transition relation and q_0 is a designated start state. For readability, a transition $(s, a, s') \in \rightarrow$ is denoted as $s \xrightarrow{a} s'$.

Each event in the set of events for each component process denotes a send (receive) statement where the receiver (sender) is explicitly identified. For example, let L_1 and L_2 be two LTSs corresponding to processes P_1 and P_2 . An event denoting sending of a message m from P_1 to P_2 is specified as $P_2!m$ and the corresponding receive event is specified as $P_1?m$. This kind of specification is similar to that of CSP and has also been used in (Lee et al, 1993). A matching operation between a send and receive is referred to as a synchronization event and is denoted by τ . We assign a unique identifier 'u' to matching send and receive events. As a convention, we denote the send event as u and the receive event as $\bar{u}$. Thus, only u or $\bar{u}$, but not both, may belong to the event set of a component LTS. A receive (send) event in L is said to be an **input** (**output**) of L if the corresponding sender (receiver) is not in L . Events that are neither inputs nor outputs are referred to as **internal events**. Also, transitions labeled by internal (input, output) events are referred to as **internal** (input, output) transitions. Classification of events in an LTS into input and output events has also been used in (Phalippou, 1994).

Given a set L of LTSs, the LTS representing the global behaviour of the set is referred to as the **composite LTS** of L . Each LTS $L_i \in L$ is referred to as a **component LTS** of L . The composite LTS of L may be obtained as the parallel composition of the component LTSs of L while hiding the internal events in L . For $L_1 = \langle Q_1, E_1, \rightarrow_1, q_1 \rangle$ and $L_2 = \langle Q_2, E_2, \rightarrow_2, q_2 \rangle$, the composite LTS L of L_1 and L_2 is given by the quadruple $\langle Q, E, \rightarrow, q_0 \rangle$, where,

- $Q \subseteq Q_1 \times Q_2$;
- $E = \{\tau\} \cup (E_1 \cup E_2) \setminus (E_1 \cap E_2)$;
- $\rightarrow \subseteq Q \times E \times Q$. A transition $(x, y) \xrightarrow{a} (x', y')$ iff
 - $(a \in E) \wedge (y = y' \wedge x \xrightarrow{a_1} x')$ or
 - $(a \in E) \wedge (x = x' \wedge y \xrightarrow{a_2} y')$ or
 - $a = \tau$ and $\exists e. ((x \xrightarrow{e_1} x' \wedge y \xrightarrow{e_2} y') \vee (x \xrightarrow{e_1} x' \wedge y \xrightarrow{e_2} y'))$.
- $q_0 = (q_1, q_2)$.

Figure 1 shows two LTSs L_1 and L_2 (events i1 through i7 are internal events). The composite LTS L of L_1 and L_2 is shown in Figure 2(a). The above definition of a composite LTS for a set of two LTSs may be extended to a set of n LTSs in a straightforward manner. The composition operation is both associative and commutative. Each state of the composite LTS is a collection of states of the component LTSs. For example, the start state of the composite LTS of L_1 and L_2 , whose start states are q_1 and q_2 respectively, is (q_1, q_2) . The states of the composite LTS are referred to as **global states**. For a global state s , $outdegree(s)$ is the number of transitions leaving s and $S_\tau(s) = \{s' \mid s \xrightarrow{\tau} s'\}$. The global states are classified into three categories. A state s for which $outdegree(s) = |S_\tau(s)|$ and $|S_\tau(s) \setminus \{s\}| = 1$ is referred to as a **hidden state**. The states which have only input transitions leaving them are referred to as **stable states**. All other states are referred to as **transient states**. It is assumed that states that have outdegree equal to zero do not exist (if they do, then it will be considered a design error and will not be relevant in the context of conformance testing).

Given an LTS $L = \langle Q, E, \rightarrow, q_0 \rangle$ and $s_1, s_2 \in Q$, s_1 and s_2 are said to be **strongly equivalent**, denoted by $s_1 \sim s_2$, if the following condition and its symmetric condition hold: $\forall a \in E, s_1 \xrightarrow{a} s'_1 \Rightarrow \exists s'_2. s_2 \xrightarrow{a} s'_2 \wedge s'_1 \sim s'_2$. Informally, $s_1 \sim s_2$ if whenever s_1 has an a transition to s'_1 , then s_2 also has an a transition to some state s'_2 such that $s'_1 \sim s'_2$ and the symmetric condition holds. States s_1 and s_2 are said to be **observationally equivalent**, denoted by $s_1 \approx s_2$, if the following conditions and their symmetric conditions hold:

- (1) $\forall a \in E \setminus \{\tau\}, s_1 \xrightarrow{\tau^* a \tau^*} s'_1 \Rightarrow \exists s'_2. s_2 \xrightarrow{\tau^* a \tau^*} s'_2 \wedge s'_1 \approx s'_2$, and
- (2) $s_1 \xrightarrow{\tau^*} s'_1 \Rightarrow \exists s'_2. s_2 \xrightarrow{\tau^*} s'_2 \wedge s'_1 \approx s'_2$.

Informally, $s_1 \approx s_2$ if whenever s_1 has a **weak a transition** (i.e., an a transition which subsumes zero or more of its preceding and succeeding τ transitions) to s'_1 , then s_2 also has a weak a transition to some state s'_2 such that $s'_1 \approx s'_2$ and the symmetric condition holds. Details concerning these equivalences may be found in (Milner, 1989).

3 BACKGROUND AND MOTIVATION

3.1 Checking experiments based approaches to conformance testing

Conformance testing based upon checking experiments involves testing the behaviour of each transition in a global LTS. A transition, (s, a, s') , is tested in the following manner:

Figure 1 Set of LTSs, $L = \{L_1, L_2\}$

- Step 1: From the current state of the IUT, bring the IUT to state s . This is usually done in two stages: 1) the implementation is brought into a known state, e.g. the initial state, using a sequence of events referred to as the **synchronizing sequence** and 2) a **transferring sequence** is then applied to bring the IUT to state s .
- Step 2: Check if the IUT can synchronize with a matching event for a ; if the IUT fails, a fault is detected, else proceed to step 3.
- Step 3: Check if the implementation has reached state s' . This is usually done using a sequence of events referred to as the **state identification sequence**.

The above three step procedure, however, is difficult to achieve due to limited controllability over the IUT. A number of test generation techniques assume the existence of a synchronizing sequence for some or all states of the LTS. For example, the **reset** capability was assumed in (Cavalli, Kim and Maigron, 1994). In the presence of nondeterminism in an LTS, it is difficult or even impossible to find transferring sequences for certain states. Generation of state identification sequences has been studied in several papers. In (Fujiwara and Bochmann, 1992), it is assumed that the LTS does not contain any *tau* transitions. In (Cavalli et al., 1993) (Cavalli, Kim and Maigron, 1994), the LTS is assumed to be deterministic. If it is not, then it is transformed into a deterministic LTS with no τ transitions. As pointed out in (Arkko, 1994), this transformation does not take the inherent nondeterminism in the IUT into consideration. (Arkko, 1994) considers LTSs containing nondeterminism and also τ transitions. Algorithms are provided to generate **state identification machines** that can be used to perform adaptive testing to check if the IUT was in an expected state. However, the complexity of these algorithms is not known. Also, to our knowledge, no such algorithms exist for generation of transferring sequences.

When a set of LTSs is considered, a test generation technique cannot make assumptions of determinism and nonexistence of τ transitions. The reason for this is that, whenever

a protocol is specified as a set of (communicating) LTSs, there are bound to be internal (nonobservable) synchronizations that result in τ transitions in the composite LTS.

3.2 The adaptive approach to conformance testing

In this section, we give an overview of the adaptive guided random walk approach (Lee et al, 1993) to conformance testing of protocols specified as a set of communicating processes where each process may be modeled as an LTS.

The basis of this approach is the claim that a test set that checks the conformance of all component processes is also a conformance test for the combined behaviour (the proofs may be found in (Lee et al, 1993) and/or references therein). An implication of this is that it suffices to test every transition in each component machine at least once rather than to test all transitions in the global behaviour. In this context, **testing** a transition means observing the starting and ending states of that transition. A transition is said to be **tested** if both its starting and ending states have been observed, **weakly tested** if its starting state is observed but the ending state is not observed, and **untested**, otherwise.

The following assumptions are made: 1) inputs can be applied to the IUT only when the IUT is in a stable state and 2) at each stable state, the state of each component process can be observed (this assumption may be realized, for example, using a status message for each component process). Assumption about being able to observe every stable state has also been made in (Arakawa and Soneoka, 1992) where it is assumed that for each stable state, the system appearance can be generated using some hardware specifications. The complexity results presented in (Lee and Yannakakis, 1994) make this assumption more interesting. With the goal of testing every transition in each component process, conformance testing is performed as follows. At each stable state reached during testing, the states of all component processes are observed and the input transitions are grouped into three categories: tested, weakly tested and untested. If untested transitions exist, one of them is selected at random. If there are no untested transitions, then one of the weakly tested transitions is selected at random. If no such transitions exist either, then one of the tested transitions is selected. The input associated with the selected transition is then applied and the next stable state reached is observed. Assume that there are n component processes. When a stable state $s' = (s'_0, s'_1, \dots, s'_{n-1})$ is reached from stable state $s = (s_0, s_1, \dots, s_{n-1})$, coverage of transitions of component processes and faulty behaviour are ascertained as follows:

- If no outputs are observed during the transition from s to s' , then a check is performed if, $\forall i$, s'_i is reachable from s_i only through internal transitions. If the check fails, then a fault has been detected. Otherwise, a positive probabilistic measure is added to the coverage probability of each internal transition in each component process that could potentially have been on the path from s to s' .
- Let the set of outputs observed be OO . Then from each component process a set OE of potentially observable outputs during the transition from s to s' is generated and the two sets are compared. If $OO \subseteq OE$ then it is assumed that there was no fault. Otherwise, a fault has been detected.

The testing procedure stops when a reasonable terminating condition is reached. An example of such a condition would be that the probabilities of coverage of all internal

transitions have reached a certain minimum threshold and all input/output transitions have been tested (Lee et al, 1993). Efficient techniques for keeping track of coverage of external transitions and the probability of coverage of internal transitions can be found in (Lee et al, 1993).

3.3 Motivation

Checking experiments based testing approaches, when applied in the realm of protocols specified as sets of LTSs, have the following limitations:

- (i) State explosion during the construction of the composite LTS.
- (ii) It is not necessary, based on results obtained in (Lee et al, 1993), to test all transitions of the composite LTS.
- (iii) Generation of state identification sequences (machines) is a difficult problem. Also, such sequences (machines) are difficult to apply to the IUT because of the lack of controllability over the IUT.
- (iv) Driving the IUT from the initial state to a given reachable state s is difficult due to the limited controllability over the IUT.

On the other hand, the adaptive testing approach has the following limitations:

- (i) When a stable state s is reached during adaptive testing, it is not known if s is reachable according to the specification or was reached due to a fault in the implementation.
- (ii) The expected output is in the form of a set of outputs rather than sequences of outputs; this may limit the fault detecting capability of the approach considerably.

These limitations can be alleviated by generating the global LTS. This, however, is a difficult task due to the state space explosion problem. These observations constitute the motivation for the results of this paper, which are:

- (i) Generation of a reduced composite LTS such that state explosion is alleviated and all stable states in the composite LTS are preserved.
- (ii) Applying the adaptive testing strategy to conformance testing with a greater fault detection capability due to the existence of the reduced composite LTS.
- (iii) Identifying testability related issues that need to be addressed during the specification if the adaptive approach to conformance testing is to be applied.

4 GENERATION OF REDUCED GLOBAL LTS

A straightforward approach to generating a reduced global LTS is to first generate the composite LTS L_c for a given set of LTSs L and then apply some reduction technique to L_c . This, however, does not alleviate the state explosion problem. To alleviate state space explosion, we use the incremental technique given in (Tai and Koppol, 1993). This technique works as follows:

- Step 1: Select a subset of L .

Figure 2 Global LTS for $(L_1 \parallel L_2)$ and its minimal LTS

- Step 2: Generate the composite LTS for the selected subset.
- Step 3: Minimize the composite LTS with respect to observational equivalence.
- Step 4: Update set L as $L = L \setminus (\text{subset of step 1}) \cup (\text{minimum LTS from step 3})$.
- Step 5: If cardinality of L is greater than 1, goto step 1, else terminate.

One of the main concerns of incremental/compositional techniques for reduced LTS generation is the selection of subsets in step 1 above. The reason is that, if a 'wrong' subset is selected, the LTS generated in step 2 may have more states than the global LTS itself! The hierarchy-based approach presented in (Tai and Koppol, 1993) provides heuristic techniques to alleviate this problem. Step 2 is based on the operational semantics of CCS. For step 3, the set of states is partitioned into equivalence classes such that each class contains all states that are observationally equivalent. The minimum observationally equivalent LTS is generated by collapsing the states in each equivalence class into a single state and updating the transitions accordingly. Due to this reason, the correspondence among global states and the states of component processes is lost. The partitioning algorithm used is the one implemented in the Concurrency Workbench (Cleaveland et al., 1993). Consider the following example. Figure 1 shows two LTSs L_1 and L_2 . The composite LTS L that represents the combined behaviour of L_1 and L_2 is shown in Figure 2(a). The LTS L_m shown in Figure 2(b) represents the minimal LTS that is observationally equivalent to L . L_m is constructed by collapsing equivalent states in L into a single state and rearranging the transitions accordingly. Note that there is no correspondence between the states of L and L_m . Also, the stable states of L are not preserved in L_m .

To preserve state information of component processes, we propose a new reduction technique for step 3 which is based on the following observations:

- **Observation 1:** For adaptive testing purposes, the correspondence between component process states and global states is needed only in the case of stable states.
- **Observation 2:** Observational equivalence may equate a stable state and a transient state. A hidden state s is observationally equivalent to the state s' ($s \neq s'$) reached by one of its τ transitions.
- **Observation 3:** Strong equivalence never equates two states that belong to different categories, i.e., if partitioning of states is done with respect to strong equivalence, then all states in a certain equivalence class are either all stable, transient or hidden.

The basic idea of the reduction algorithm is to keep stable states intact while gaining as much reduction as possible in terms of the nonstable states. We exploit observations 2 and 3 to reduce the number of hidden and transient states while leaving stable states intact.

Reducing hidden states

Let $L = \langle Q, E, \rightarrow, q_0 \rangle$ be the given LTS. Let H be the set of all hidden states in L . For $s \in Q$, let $Pred(s) = \{s' \mid s' \xrightarrow{a} s\}$ and let $Succ(s) = \{s' \mid s \xrightarrow{a} s'\}$. The goal is to eliminate the hidden states in L . To achieve this goal, for each state $s \in H$, L is updated according to the following procedure:

- Step 1: $\rightarrow = \rightarrow \setminus \{s \xrightarrow{\tau} s\}$. Informally, if there is a self looping τ -transition at state s , delete it.
- Step 2: Perform the following updates to L in the given order:
 - (i) $\rightarrow = \rightarrow \cup \{x \xrightarrow{a} y \mid x \xrightarrow{a} s \wedge s \xrightarrow{\tau} y\}$. Informally, for each $p \in Pred(s)$, if p has an a transition to s then, $\forall q \in Succ(s)$, add a new transition $p \xrightarrow{a} q$.
 - (ii) $\rightarrow = \rightarrow \setminus \{x \xrightarrow{a} y \mid x = s \vee y = s\}$. i.e., delete all transitions leading into s and all those leaving s .
 - (iii) $Q = Q \setminus \{s\}$.

The above procedure will henceforth be referred to as procedure **h_reduce(L)**. Since $H \subseteq Q$ and due to the fact that Q is finite, it follows that **h_reduce(L)** terminates and has polynomial complexity. Let L^{r1} be the LTS obtained by applying the above procedure to L . L^{r1} does not contain any hidden states. For example, consider the set of LTSs shown in Figure 1. The reduced LTS, L_{p1} , is shown in Figure 3. L_{p1} does not contain any hidden states, but contains all the stable states contained in L .

Property 1 $L^{r1} \approx L$.

Proof. Follows from observation 2. $\square$

Reducing transient states

Procedure **h_reduce(L)** eliminates all hidden states from the LTS L . Further reduction can be obtained by reducing the number of transient states. For example, L_{p1} shown in

Figure 3 L_{p1} , the reduced LTS without hidden states

Figure 3, can be reduced further by collapsing the equivalent transient states (2,1) and (3,1) into a single state. The goal is to achieve this reduction. We exploit observation 3 to gain this reduction. Let $L = \langle Q, E, \rightarrow, q_0 \rangle$ be the LTS that is to be reduced. It is assumed that L has already been processed by **h_reduce** and hence does not contain any hidden states. The reduction procedure for LTS L , henceforth referred to as **t_reduce**(L), is as follows:

- **Step 1:** Obtain the partition P of the set of states Q with respect to strong equivalence. This may be done by using well known algorithms given in (Kanellakis and Smolka, 1983) (Paige and Tarjan, 1987). According to observation 3, each block $B \in P$ either contains all stable states or all transient states. A block is said to be stable (transient) if it contains only stable (transient) states. Let $P_s = \{B \in P \mid B \text{ is stable}\}$.
- **Step 2:** Refine the partition P as follows. For each $B \in P_s$, do the following:
 - Let $k = |B|$. Create k sets $B_i, 0 \leq i \leq k-1$, where $B_i = \{s \mid s \text{ is the } i\text{th element of } B\}$.
 - Update P as follows: $P = P \setminus \{B\} \cup \cup_{i=0}^{k-1} \{B_i\}$.
- **Step 3:** Construct the reduced LTS by collapsing states in the same block in P into a single state and arranging the transitions appropriately.

The complexity of the above algorithm is the same as the complexity of the partitioning algorithm. The partitioning algorithms have polynomial complexity. Note that due to step

Figure 4 L_{p2} , the reduced LTS without equivalent nonstable states

2 of the algorithm above, each stable block contains exactly one stable state. Hence, when states in a block are collapsed into a single state in step 3, the component process state information may be lost in the case of transient blocks. In the case of stable blocks, due to the fact that no collapsing of states is needed, the component process state information is preserved. Figure 4 shows the LTS L_{p2} which is the result of applying the above reduction procedure to the LTS L_{p1} . L_{p2} contains all the stable states of L and does not contain any equivalent transient states.

Property 2 Let L^{r2} be the LTS obtained by applying the above procedure to L . Then, $L^{r2} \approx L$.

Proof. Since we use partitioning with respect to strong equivalence, $L^{r2} \sim L$. Due to the fact that strong equivalence implies observational equivalence, $L^{r2} \approx L$. $\square$

Procedure **t_reduce(L)** may, however, introduce hidden states. These hidden states can be eliminated using **h_reduce(L)**. Thus, the overall reduction procedure, referred to as **reduce(L)**, is as shown in Figure 5. In this algorithm, **h_flag** and **t_flag** are boolean variables. It is assumed that the procedures **h_reduce(L)** and **t_reduce(L)** return **true** if reduction is gained and **false** otherwise. Procedure **reduce(L)** has polynomial complexity. Also, it can be made more efficient by making procedures **h_reduce(L)** and **t_reduce(L)**

```

procedure reduce(L)
{
  h_flag = h_reduce(L);
  t_flag = t_reduce(L);
  while (t_flag)
  {
    h_flag = h_reduce(L);
    if (h_flag)
      t_flag = t_reduce(L);
    else
      t_flag = false;
  }
}

```

Figure 5 Algorithm for reduction

to only consider a relevant subset of the states in L . For the LTSs shown in Figure 1, the reduced LTS generated by the above procedure is given in Figure 4.

5 CONFORMANCE TESTING BASED ON REDUCED GLOBAL LTS

In this section, we describe how the reduced global LTS generated in the previous section may be used to perform conformance testing based on an adaptive guided random walk. The following issues need to be addressed: 1) generation of test cases 2) construction of the tester, and 3) issues related to testability.

Since the approach to testing being considered is an adaptive guided random walk, no test cases have to be generated *a priori*. At each stable state reached during testing, the next input to be provided to the IUT is done as described in section 3.2. The construction of testers is more involved. Note that when testing LTSs, the tester needs to synchronize with the IUT.

5.1 Tester generation

At each stable state, a tester can be constructed during adaptive testing as follows:

- Identify the stable state and select the next input to provide to the IUT as discussed in section 3.2.
- In the reduced LTS generated in the previous section, start a depth first traversal at s along the selected input transition(s).
- Construct the subgraph T corresponding to this traversal with the condition that traversal is made only along internal and output transitions.

- If T is nondeterministic, transform it into a deterministic LTS.
- Invert inputs and outputs in T .

5.2 Testability issues

In this paper, we restrict our discussion on testability to the executability of test cases. Issues which are related to coverage need further study and will not be discussed here. When performing conformance testing based on an adaptive guided random walk, the main testability criterion is stable state identification. Stable state identification may become problematic due to **divergence** (a loop consisting of only internal transitions). We assume fairness on the part of the scheduler and deem divergence as not being catastrophic as long as it can be exited. If there is a situation when the IUT stays in an internal loop once it enters it, it is considered a design error and is not relevant during conformance testing - it is assumed that such situations do not exist. Fair divergence can still pose problems in regard to testability. We identify three types of divergence: 1) **hidden divergence** which corresponds to a τ loop that can be exited through only τ transitions, 2) **transient divergence** which corresponds to a τ loop that can be exited using only τ and/or output transitions, and 3) **stable divergence** which corresponds to a τ loop that can be exited only through input transitions.

Hidden and transient divergence do not cause difficulty during state identification, however, when a stable divergence is reached, the IUT cannot proceed any further without an input. In an adaptive guided random walk, an input is provided after observing the current stable state of the IUT. With stable divergence, the IUT would be looping internally among a set of transient states and hence identifying the states of component processes is not possible. Therefore, the adaptive guided random walk fails when stable divergence exists. The necessary testability criterion, hence is not to have stable divergence. The reduced LTS generated in section 4 can be used to detect stable divergence, if exists. The procedure would be as follows:

- Step 1: Identify strongly connected components of τ transitions (i.e., a set of global states where for any pair of states belonging to this set, there is a path consisting of only τ transitions).
- Step 2: Check if each component is such that there exists at least one state that has an output transition or a τ transition leading out of the strong component.
- Step 3: If the check in step 2 fails, stable divergence has been detected.

6 RELATED WORK

Reduction of the LTS while preserving certain information has been studied in different contexts by a number of researchers. In the context of conformance testing, however, the only two approaches that we are aware of are presented in (Cavalli, Kim and Maigrón, 1994) and (Arakawa and Soneoka, 1992). In (Cavalli, Kim and Maigrón, 1994), specifications given in the form of sets of LTSs are considered. The reduction technique is applied to the composite LTS. This does not alleviate the state explosion problem because the composite LTS has to be constructed in the first place. Also, the reduction is with re-

spect to trace equivalence. This reduction does not take into consideration the inherent nondeterminism in the IUT.

The approach presented in (Arakawa and Soneoka, 1992) considers specifications which are in the form of sets of communicating processes. Each process is represented by its control flow graph. Interprocess communication is assumed to be through FIFO channels. For a specification S , the global behaviour of the set of component processes may be obtained in terms of a reachability graph. A global state is defined as a collection of states of component control flow graphs and the contents of the communication channels. To alleviate state space explosion inherent in the construction of a reachability graph, (Arakawa and Soneoka, 1992) presents an on the fly reduction technique wherein a reduced reachability graph is generated. Each transition in the reduced graph may be associated with more than one event. However, transitions leaving stable states are associated with exactly one input event. This reduced graph contains all stable states that are reachable from the initial global state while adhering to the above requirements on the transitions. The reduction algorithm, however, is based on the following assumptions: 1) the control flow graph for each component process is deterministic, and 2) in the control flow graph for each process, if a node has outdegree greater than 1, then the set of all outgoing transitions from that node must only contain a mixture of either input transitions or internal receive transitions. Due to these assumptions and due to the different semantic model used for the specifications, this reduction technique is not directly applicable in the realm of LTSs. Also, the reduced graph may contain equivalent nonstable states. Hence, procedure $\text{reduce}(L)$ presented in this paper is more general and is directly applicable in the realm of LTSs.

A reduction technique, referred to as the ICR method, was proposed in (Lapone et al., 1989) in the context of protocol verification. The reduction is with respect to observational equivalence. However, the notion of observational equivalence used in this approach is quite different from the one given in (Milner, 1989). For instance, it equates $a + \tau.b$ and $a + b$. Nevertheless, the reduction does preserve all stable states and hence is applicable in the context of conformance testing using an adaptive guided random walk. It has been demonstrated through a large real life protocol example that this technique produces at least an order of magnitude reduction in the number of states. The rules for reduction in this approach do not collapse equivalent nonstable states into a single state. For this reason, we feel that our approach produces a more significant reduction.

Testing of communicating finite state machines was also considered in (Petrenko et al., 1994). Each component process is represented as a nondeterministic finite state machine (NFSM). Only serial composition of component processes is considered. Test generation is based on a fault model wherein it is assumed that exactly one component process may be faulty. It is our belief that such a fault model may only be suitable when the IUT is embedded within a system under test but may not be suitable when the IUT is directly accessible. Due to these reasons, we refrain from any comparison of the work presented in this paper with that of (Petrenko et al., 1994).

7 CONCLUSIONS AND FUTURE WORK

We have presented a reduction technique which, for a set L of LTSs, generates a reduced LTS observationally equivalent to the composite LTS for L while preserving all the stable

states of the composite behaviour of L . The reduction is performed such that the problem of state explosion is alleviated. For a special class of LTSs, we have shown how our reduction technique can be used effectively in conjunction with the adaptive guided random walk approach to conformance testing. This class of LTSs includes LTSs which correspond to CSP style specifications and also to the IOFSM formalism (for the case where only direct naming is considered) of (Phalippou, 1994), for which it has been proved that testing each transition of each component process is sufficient for conformance testing of a set of communicating processes. Also, we have shown how to construct testers and identified criteria for testability in the context of adaptive guided random walk.

While empirical evidence for our reduction technique is not provided in this paper, we have shown with valid reasoning that our reduction approach may perform better than other approaches that have been shown to produce significant reduction. We plan to perform empirical studies with large protocols and study the impact on fault coverage in relation to the adaptive guided random walk. Also, in light of the results presented in (Okazaki et al., 1994), we plan to investigate how our reduction technique can aid test sequence generation for interoperability testing.

Acknowledgement We would like to thank Dr. Rance Cleaveland and V. Natarajan for helpful discussion on equivalences in process algebra.

REFERENCES

- Arkko, J. (1994) On the Existence and Production of State Identification Machines for Labeled Transition Systems, *Sixth Int. Conference on Formal Description Techniques - FORTE'93*, pp 351-366, North Holland.
- Arakawa, N. and Soneoka, T. (1992) A Test Case Generation Method for Concurrent Programs, *Fourth Int. Workshop on Protocol Test Systems*, pp 95-106, North Holland. Also appears as A Test Case Generation Method for Black Box Testing of Concurrent Programs, *IECE Trans. Commun.*, Vol. E75-B, No. 10.
- Brinksma, E., Alderden, R. and Langerak, R. (1990) A Formal Approach to Conformance Testing, *2nd Int. Workshop on Protocol Test Systems*, pp 349-363, North Holland.
- Cavalli, R. R., Favreau, J-P. and Phalippou, M. (1994) Formal Methods in Conformance Testing, *Sixth Int. Workshop on Protocol Test Systems*, pp 3-17, North Holland.
- Cavalli, A. and Kim, S. U. (1992) Protocol Conformance Test Generation Using a Graph Rewriting System, *Fourth Int. Workshop on Protocol Test Systems*, pp 285-288, North Holland.
- Cavalli, A., Kim, S. U. and Maigron, P. (1993) Automated Protocol Conformance Test Generation Based on Formal Methods for LOTOS Specifications, *Fifth Int. Workshop on Protocol Test Systems*, pp 237-248, North Holland.
- Cavalli, A., Kim, S. U. and Maigron, P. (1994) Improving Conformance Testing for LOTOS, *Sixth Int. Conference on Formal Description Techniques - FORTE'93*, pp 367-381, North Holland.
- Cleaveland, R., Parrow, J. and Steffen, B. (1993) The Concurrency Workbench: A Semantics Tool for the Verification of Concurrent Systems, *ACM Tran. Programming Languages and Systems*, Vol 15, No. 1, pp 36-72.
- Fujiwara, S. and Bochmann, G.v. (1992) Testing nondeterministic state machines with

- fault coverage, *Fourth Int. Workshop on Protocol Test Systems*, pp 267-280, North Holland.
- Hoare, C.A.R. (1992) *Communicating Sequential Processes*, Prentice-Hall.
- Kanellakis, P.C. and Smolka, S.A. (1983) CCS Expressions, Finite State Processes, and Three Problems of Equivalence, *Proc. of ACM Symposium on Distributed Computing*, pp 228-240.
- Lapone, A. M., Sabnani, K. K. and Uyar, M. U. (1989) An Algorithmic Procedure for Checking Safety Properties of Communication protocols, *IEEE Transactions on Communications*, pp 940-948, September.
- Lee, D., Sabnani, K., Kristol, D. M., Paul, S. and Uyar, M. U. (1993) Conformance Testing of Protocols Specified as Communicating FSMs, *IEEE INFOCOM'93*, pp 115-127.
- Lee, D. and Yannakakis, M. (1994) Testing Finite-State Machines: State Identification and Verification, *IEEE trans. on Computers*, Vol. 43, No. 3, pp 306-320.
- Milner, R. (1989) *Communication and Concurrency*, Prentice-Hall.
- Okazaki, N., Park, M., Ohta, M. and Takahashi, K. (1994) A New Test Sequence Generation Method for Interoperability Testing, *Proc. 7th Intl. Workshop on Protocol Test Systems*, pp 229-245.
- Paige, R. and Tarjan, R. E. (1987) Three Partition Refinement Algorithms, *SIAM Journal of Computing*, Vol. 16, No. 6, pp 973-989.
- Phalippou, M. (1994) Executable testers, *Sixth Int. Workshop on Protocol Test Systems*, pp 35-50, North Holland.
- Petrenko, A., Yevtushenko, N. and Dssouli, R. (1994) Testing Strategies for Communicating FSMs, *7th Int. Workshop on Protocol Test Systems*, pp 181-196.
- Sidhu, D. and Leung, T. (1989) Formal Methods for Conformance Testing: A Detailed Study, *IEEE transaction on software engineering*, vol. 15, no. 4, pp 413-426.
- Tai, K. C. and Koppol, P. V. (1993) Hierarchy-Based Incremental Analysis of Communication Protocols, *proc. of Int. Conf. on Network Protocols*, pp 318-325.
- Tretmans, J. (1994) A Formal Approach to Conformance Testing, *Sixth Int. Workshop on Protocol Test Systems*, pp 257-276, North Holland.
- Turner, K. J. (1993) *Using Formal Description Techniques: An Introduction to Estelle, LOTOS and SDL*, Wiley.

Biography

Pramod V. Koppol is a Ph.D. student in the Computer Science department at North Carolina State University. He received an M.S. degree in Computer Science from Southern Illinois University at Carbondale, IL. His research interests are in Software Engineering, Distributed Systems and Communication Protocols. He is currently working on analysis, testing and debugging of concurrent software and communication protocols.

Kuo-Chung Tai is a professor in the Computer Science department at North Carolina State University. He has published papers in the areas of software engineering, distributed systems, programming languages, and compiler construction. His current research interests include analysis, testing, and debugging of sequential and concurrent software. He received his Ph.D. degree in Computer Science from Cornell University in 1977.