Verification and Diagnosis of Testing Equivalence and Reduction Relation*

Z.P.Tao, G.v. Bochmann and R. Dssouli

Université de Montréal, Département d'Informatique et de
Recherche Operationnelle, C.P. 6128, Succ. "A"
Montréal, P.Q., Canada H3C-3J7
Email: {tao, bochmann, dssouli}@iro.umontreal.ca

Abstract

In protocol engineering, a common approach for system design and implementation is to verify if an implementation specification (or any lower level specification) satisfies its service specification. If an implementation specification does not satisfy its service specification, it is necessary to find out the faults and correct them. In this paper, we present an efficient algorithm for verifying whether an implementation satisfies its service specification related by the testing equivalence and the reduction relation [1], and generating diagnostic information if an implementation does not satisfy its service specification, based on the transformation of the service specification into a special deterministic machine, called refusal graph, and the coupled product of the refusal graph and the implementation.

1. Introduction

In protocol engineering, a common approach for system design and implementation is to verify if an implementation specification (or any lower level specification) satisfies its service specification. Here the implementation specification represents a protocol specification or any lower level specification of a protocol, specifying *how* the functions should be implemented; while a service specification describes abstractly *what* functions a protocol should have. If an implementation specification does not satisfy its service specification, it is necessary to find out the faults and correct them. To verify that an implementation of a protocol satisfies its service specification, one needs to prove the conformance relation between the service specification and the implementation specification. Many conformance relations have been proposed in the literature for this purpose, among which are the testing equivalence and the reduction relation [1].

A number of methods have been proposed in recent years to solve this problem. An algorithm is reported in [2] for verifying if two specifications are testing equivalent: first the two specifications are transformed into *acceptance graphs*, which are deterministic and trace equivalent to their original specifications, with every state assigned an acceptance set; then an algorithm for strong bisimulation is used for the verification of testing equivalence. This algorithm is capable of verifying if two specifications satisfy the testing equivalence. However, when they are not related by the testing equivalence, the algorithm is unable to give diagnosis information indicating where the problem is. Recently, an algorithm has been developed for giving diagnosis information for testing preorder [3] (in fact, it is a kind of reduction relation). The algorithm also first transforms the two specifications into acceptance graphs, and then uses the algorithm proposed in [7] to verify if the two acceptance graphs satisfy bisimulation preorder. If they do not satisfy the bisimulation preorder, a postprocessing step is used to generate diagnostic information based on the results saved into a stack during the verification step.

This paper is motivated by the following observations:

1) The algorithms presented in [2] and [3] are unnecessarily complex both in time and space.
2) The algorithm proposed in [3] can only produce information for a single fault, and the algorithm proposed in [2] is not capable of producing diagnostic information.
3) Protocol design and implementation often involve repeated debugs and revisions, which result in many repeated applications of the algorithm for the verification of two specifications. Therefore it is important to improve the algorithm so that it is able to generate complete fault information , and is efficient both in time and space.

* This work is supported by the IDACOM-NSERC-CWARC Industrial Research Chair on Communication Protocols.

0-8186-7216-1/95 $04.00 © 1995 IEEE

4) The definition of the testing preorder and testing equivalence in [2] and [3], although similar to the testing equivalence and the reduction relation defined in [1], discriminates two specifications that have different *divergent* properties. This may be unnecessary, since in protocol engineering, a correct specification of reliable message transmission through an unreliable channel always has some divergence, while its service specification may not have it. Therefore, the definition, and the corresponding algorithm, need to be modified in protocol design. In addition, to detect which state is divergent needs significant computation.

The contribution of this paper is as follows:

1) A new method and a corresponding algorithm for the verification of testing equivalence and reduction relation are presented. The algorithm reduces computation significantly, both in space and time, compared with the method proposed in [2] and [3]. The simplification is mainly due to the fact that we only transform the service specification, not the implementation, into a refusal graph for computing the reduction relation.
2) The algorithm produces a diagnostic information graph which contains complete fault information.

This paper is organized as follows. In the next section, we will define the model and the relations used in this paper. In Section 3, algorithms will be proposed for verifying the reduction relation and the testing equivalence, and producing diagnostic information. An example will be given to show the application in Section 4. In Section 5, we will discuss and compare the results of this paper with those in [2] and [3].

2. Preliminaries

We will use the model of Finite Labeled Transition Systems (FLTS) [6]. An FLTS is defined as follows.

Definition 1 (FLTS) [6]: A non-deterministic FLTS M is a four-tuple M = (Q, Σ, δ, q0), where

- Q is a finite set of states.
- Σ is a set of observable events.
- δ is a transition function, $\delta: Q\times(\Sigma\cup\{\tau\})\times\rightarrow 2^Q$ with τ denoting an internal event, which defines a set $S \subseteq 2^Q$ of next states when an event $e \in \Sigma\cup\{\tau\}$ occurs in the current state $q \in Q$. When the FLTS is in state q, we say that the transition to q', written $q - e \rightarrow q'$ or $q' \in \delta(q, e)$, is enabled, where $\forall q, q' \in Q$, $e \in \Sigma\cup\{\tau\}$. The transition $q - e \rightarrow q'$ is said an incoming transition of q' and an outgoing transition of q; q is said a parent state of q'.
- q0 is the initial state.

When the transition function is defined by $\delta: Q\times\Sigma\rightarrow Q$, then such an FLTS is said *deterministic*.

The following notations are used in the rest of this paper.

$q-e\rightarrow$	$\exists q'$, such that $q - e \rightarrow q'$.
$q - e \not\rightarrow$	$\neg(q - e \rightarrow)$,
$q - \tau^k\rightarrow q'$	An FLTS may engage in a sequence of k internal events, and after doing so, enters state q'.
$q = e \Rightarrow q'$	$\exists k0, k1 \in N$, such that $q - \tau^{k0}e\tau^{k1} \rightarrow q'$ for $e \neq \tau$.
$q = \varepsilon \Rightarrow q'$	$\exists k0 \in N$, such that $q-\tau^{k0}\rightarrow q'$.
$q = e \Rightarrow$	$\exists q'$, such that $q = e \Rightarrow q'$.
$q = e \not\Rightarrow$	$\neg(q = e \Rightarrow)$, i.e., there is no state q' such that $q = e \Rightarrow q'$.
$q = t \Rightarrow q'$	For $t = e1 ...en$ where $e1, ..., en \in \Sigma$, $\exists k0, ..., kn \in N$ such that $q - \tau^{k0}e1\tau^{k1} e2 ... en \tau^{k0} \rightarrow q'$. t is called a trace.
$q = t \Rightarrow$	$\exists q'$, such that $q = t \Rightarrow q'$.
Tr(M)	Is the trace set of an FLTS M, i.e., $Tr(M) = \{t \mid q0 = t \Rightarrow\}$.
S after t	S after $t = \{q \mid q0 = t \Rightarrow q\}$, where q0 is the initial state.
Ref(q)	$Ref(q) = \{e \mid q=e \not\Rightarrow \text{ and } e\in\Sigma\}$.
Acc(p)	$Acc(p) = \{e \mid p = e \Rightarrow \text{ and } e\in\Sigma\}$.
$\mathcal{P}(S)$ of	For a given set S, $\mathcal{P}(S)$ denotes a power set S, i.e., set of subsets of S.

Definition 2 (*Coupled product*): A *coupled product* M1 # M2 of two FLTSs M1 = (Q1, Σ1, δ1, p0) and M2 = (Q2, Σ2, δ2, q0) is an FLTS M = (Q, Σ, δp, (p0, q0)) where:

- Q is a subset of Q1× Q2;
- $\Sigma = \Sigma1 \cup \Sigma2$ is the set of events;
- (p0, q0) is the initial state;
- δp is the transition relation defined on Q such that for $p, p' \in Q1$ and $q, q' \in Q2$:
  1) $(p, q) - e_1 \rightarrow (p', q)$ if $p- e_1 \rightarrow p'$ and $e_1 \in \Sigma1\cup\{\tau\} - \Sigma2$;
  2) $(p, q) - e_2 \rightarrow (p, q')$ if $q- e_2 \rightarrow q'$ and $e_2 \in \Sigma2\cup\{\tau\} - \Sigma1$;
  3) $(p, q) - \mu \rightarrow (p', q')$ if $p- \mu \rightarrow p'$ and $q- \mu\rightarrow q'$ with $\mu \in \Sigma1\cap\Sigma2$.
  4) for other cases, no transition is defined.

Definition 3 (*Submachine*): An FLTS M' = (Q', Σ', δ', q0') is a submachine of another FLTS M = (Q, Σ, δ, q0) if (a) q0' = q0; (b) $Q' \subseteq Q$; (c) $\Sigma' \subseteq \Sigma$; and (d) $\delta' \subseteq \delta$.

Definition 4*(reduction) [1]:* Given two FLTSs M1 = (Q1, Σ1, δ1, p0) and M2 = (Q2, Σ2, δ2, q0), M1 is a reduction of M2, written M1 $\angle$ M2, if the following conditions are satisfied:

1) Trace (M1) $\subseteq$ Trace(M2).
2) For any t $\subseteq$ Tr(M1) $\cap$ Tr(M2) and any q $\in$ S after t in M1, there is a state p $\in$ S after t in M2 such that Ref(q) $\subseteq$ Ref(p).

Intuitively, M1 $\angle$ M2 iff M1 has fewer traces than M2, and placed in any environment whose traces are limited to those of M1, M1 can not deadlock when M2 can not deadlock, i.e., M1 deadlocks less often than M2.

Definition 5 *(testing equivalence) [1]:* Given two FLTSs M1 and M2, M1 is testing equivalent to M2, written M1 ~ M2, if M1 $\angle$ M2 and M2 $\angle$ M1.

3. The Proposed Algorithms

3.1. The Definition of Refusal Graph

In this section, we define a special deterministic machine, called Refusal Graph, for a (nondeterministic) FLTS, similar to the definition of an acceptance graph defined in [2] and [3].

Definition 6 (*After set*): For M = (Q, Σ, δ, q0), we define the *after set* of state p as A(p) = {p' | p = ε ⇒p' }. For any S $\in$ $\mathcal{P}$(Q), we denote $S^{\tau} = \bigcup_{p \in S} A(p)$.

The *After* set A(p) intuitively describes all the reachable states from p by executing zero, one or more internal events. Below we will define the concept of refusal graph, similar to the acceptance graph defined in [8].

Definition 7 (*Refusal graph*): A *Refusal Graph* (RG) is a 5-tuple G = (Q, Σ, δ, R, q0), where

- Q is a finite set of states;
- Σ is a set of observable events;
- δ: Q×Σ→Q is a transition function;
- R: Q→ $\mathcal{P}$($\mathcal{P}$(Σ)) is a mapping from Q to a set of subsets of Σ. R(p) is called a set of refusal sets of state p.
- q0 is the initial state.

Note that RG is deterministic.

The mapping of R and δ should satisfy the following constraints:

- ∀p$\in$ Q, there is a subset Ref$\in$ R(p) such that Σ - Ref $\neq \phi$;
- ∀p$\in$ Q, if there is an event e $\in$ Σ such that p – e $\not\rightarrow$, then there is a subset Ref$\in$ R(p) and e $\in$ Ref;
- ∀p$\in$ Q, if ∃q$\in$ Q such that p–e→q, then ∃Ref $\in$ R(p) such that e $\in$ Σ -Ref.

Definition 8 (*Correspondence between an FLTS and a RG*): Given an FLTS M = (Q, Σ, δ, q0) and a RG G = (Q', Σ, δ', R, q0'). We say that G is a corresponding RG of M iff:

1) Tr(M) = Tr(G);
2) ∀t$\in$ Tr(M), if q0 = t ⇒q, then for any state q' such that q0' = t ⇒q', there is a set Ref $\in$ R(q') and Ref = Ref(q); similarly, if q0' = t ⇒q', for every Ref $\in$ R(q'), there is a state q such that q0 = t ⇒q, and Ref = Ref(q), where q $\in$ Q and q' $\in$ Q'.

Lemma 1 (Finding a corresponding RG for a given FLTS): Given an FLTS M = (Q, Σ, δ, q0), the following G = (Q', Σ, δ', R, q0') is the corresponding RG of M:

1) Q' = {p|p$\in$ $\mathcal{P}$(Q), p = p^{τ}};
2) q0' = A(q0);
3) ∀p$\in$ Q', R(p) = {Ref(q)|q $\in$ p};
4) ∀p, q$\in$ Q', we have p–e→q iff q = {q'|∃p'$\in$ p such that p'–e→q'}$^{\tau}$.

Proof: 1) The proof of Tr(M) = Tr(G) (condition 1 of Definition 8) can be constructed by using a similar method provided in [4].

2)To prove that the RG from Lemma 1 satisfies the condition 2 of Definition 8, we consider the fact that G is deterministic (condition 2 and 4 of Lemma 1). For ∀t$\in$ Tr(M) and q $\in$ Q, if q0 = t ⇒q, then there is one and only one state q' $\in$ Q' such that q0' = t ⇒q'. Obviously, q$\in$ q'. From condition 3 of Lemma 1, there is a set Ref $\in$ R(q') and Ref = Ref(q). Similarly, we can prove that for every Ref $\in$ R(q'), there is a state q $\in$ Q such that q0 = t ⇒q, and Ref = Ref(q).

In this lemma, a set of states in M ($0 \le i \le n$) is considered as one state in G. This is similar to the method given in [4] for transforming a nondeterministic finite state machine to a trace equivalent deterministic finite state machine, except for the refusal set.

The following algorithm is developed to construct a refusal graph for a given FLTS M according to Lemma 1. The algorithm works as follows. In Step 1, a sub algorithm Algorithm-Ref(M) is used to obtain Ref(p) for each state p of M. It first computes the acceptance set Acc(p) of a state p, by simply adding every observable event that can be enabled from a state p' reachable from p through executing a number of internal events. And then, compute Ref(p) = Σ – Acceptance-set(p), where Σ is the set of observable event for a given FLTS M. This sub-algorithm can be implemented more efficiently. However, for the sake of presentation, we do not optimize it in this paper.

In Step 2, the initial state p0' is constructed by computing X0 = A(p0), and a set of refusal sets is copied to p0' from the states contained in X0. p0' is marked TP,

representing that this state will be further processed. In Step 3, every state pi' marked TP is expanded according to the definition of coupled product. Each new state pj' contains a set of refusal sets from the states contained in Xj. The new state pj' is marked TP, and the processed state pi' is marked PD, representing that the processing has been done. The procedure continues until no state is marked TP.

Algorithm-RGraph(M)
Input: An FLTS M = (Q, Σ, δ, p0);
Output: The refusal graph G = (Q', Σ, δ', R, q0') of M;

Var: pi', pj' /*state of a refusal graph*/
Var: p, p' /*state of M*/
Var: Xi(e), Xj /*a set of states in M*/

Begin
1) Call Algorithm-Ref(M).
2) Compute X0 = A(p0), create state p0' and mark it TP; /*TP = To be Processed */
   Let R(p0') = {Ref(p)|p ∈ X0};
3) Do the following while there is a state pi' marked TP:
   a) For every e ∈ Σ do the following:
      i) Compute $Xi(e) = \bigcup_{p \in Xi} \{A(p') | p - e \rightarrow p' \in \delta\}$;
      ii) If Xi(e) is not empty and there is no previously created Xj containing *exactly* all the states in Xi(e)), do the following:
      - Create such an Xj containing *exactly* all the states in Xi(e);
      - Create a state pj' and mark it TP;
        Let R(pj') = {Ref(p)|p ∈ Xj};
      - Create a transition labelled e from pi' to pj' .
   b) Mark pi' PD. /* PD = ProcesseD */

End

Algorithm-Ref(M)
Input: M = (Q, Σ, δ, p0);
Output: Ref(p) is assigned to Each state p of M.

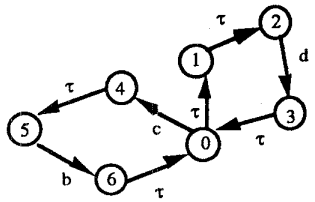
Begin
For every state p in Q
Compute Ref(p) = Σ – Acceptance-set(p);
End

Acceptance-set(p)
Var: mark[] /*an array of state marks*/

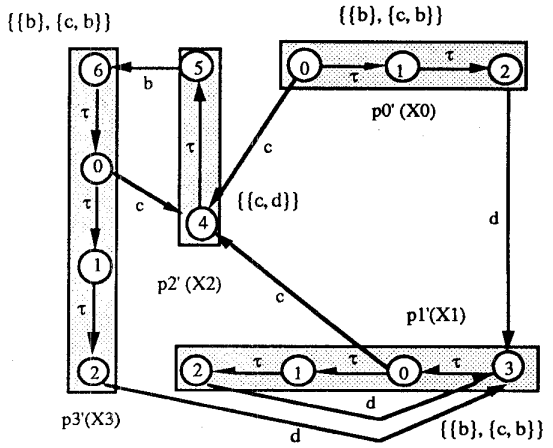
Begin
1) Let mark[p] = p;
2) For every e ≠ τ if there is a state p' such that p–e→ p' then add e to Acc(p);
3) For every p' such that p–τ→ p' If mark[p'] ≠ p then Acc(p) = Acc(p) ∪ Acceptance-set(p');
Return Acc(p)

End

Example 1: For the given FLTS M specified by Fig.1(a), where Σ = {c, d, b}, the obtained refusal graph is shown in Fig.1(b). In Fig.1(b), we have the shadowed boxes: X0 = A(0), X1 =A(3), X2 = A(4), X3 = A(6). The refusal sets are shown beside each state of the RG.



(a)



(b)

Fig.1 (a) The specification M, (b) The RG of M.

3.2. Verification and Diagnosis of the Reduction Relation

The purpose of the algorithm proposed in this section is: given an implementation specification P = (Q_P, Σ_P, δ_P, p0) and a service specification S = (Q_S, Σ_S, δ_S, q0) , to check 1) whether P ∠ S is true; 2) if not, generating the diagnostic information.

We have defined the concept of refusal graph. An important property of a refusal graph is that: for any trace t of G = (Q', Σ, δ', R, q0'), there is only one state q such that q0' = t ⇒ q. Construct the product P#G, for any trace t ∈ Tr(P)∩Tr(S) and every state p ∈ S after t in P, there is only one (p, q) in P#G such that (q0, q0') = t ⇒ (p, q). If P ∠ S, then according to the definition of the reduction relation, there is at least one refusal set Ref ∈ R(q) such that Ref(p) ⊆ Ref since q is the only state such that q0' = t ⇒ q in G. In

addition, to satisfy $Tr(P) \subseteq Tr(S)$, for any event $e \in \Sigma_p$, $p - e \rightarrow p'$ in P implies that there is a state (p', q') such that $(p, q) - e \rightarrow (p', q')$ in P#G. Based on this discussion, we have the following theorem.

Theorem 1: Given $P = (Q_p, \Sigma_p, \delta_p, p0)$ and $S = (Q_S, \Sigma_S, \delta_S, q0)$, let $Gs = (Q, \Sigma_S, \delta, R, q0')$ be the corresponding RG of S. $P \angle S$ iff there is a submachine M of P#Gs such that for every state (p, q) of M,

1) There is at least one refusal set $Ref \in R(q)$ such that $Ref(p) \subseteq Ref$.
2) For any event $e \in \Sigma_p$, $p - e \rightarrow p'$ in P implies that there is a state (p', q') such that $(p, q) - e \rightarrow (p', q')$ in M.

Proof: ($\Rightarrow$) If $P \angle S$, then $Tr(P) \subseteq Tr(S)$, that is to say, the condition 2 of Theorem 1 is true. For any trace $t \in Tr(P) \cap Tr(S)$ and every state $p \in S$ after t in P, there is only one (p, q) in P#G such that $(q0, q0') = t \Rightarrow (p, q)$. According to Definition 4, the condition 2 of Theorem 1 is true.

($\Leftarrow$) From condition 2, $Tr(P) \subseteq Tr(S)$. For any trace $t \in Tr(P) \cap Tr(S)$ such that $p0 = t \Rightarrow p$ and $q0 = t \Rightarrow q$, there must be a state (p, q) in M from the definition of # product and the fact $Tr(P) \subseteq Tr(S)$. From condition 1, there is at least one refusal set $Ref \in R(q)$ such that $Ref(p) \subseteq Ref$. Therefore, $P \angle S$ is proved.

This theorem gives us the idea how to construct an algorithm for verifying the reduction relation and generating diagnostic information: we can first construct the product P#Gs, and then check if there is any state violating the two conditions in P#Gs.

The algorithm works as follows. In Step 1, it computes Ref(p) for each state p of P by using Algorithm-Ref, and compute Gs of S by Algorithm-RGraph.

In the second step, M = P#Gs is computed, at the same time, the following two conditions are verified:

1) $Tr(P) \subseteq Tr(S)$ is verified by checking if there is a state (p, q) in M such that $p-e\rightarrow$ in P but $q - e \not\rightarrow$ in Gs.
2) For every state (p, q) of M, checking if there is at least one refusal set $Ref \in R(q)$ such that $Ref(p) \subseteq Ref$.

Since a RG is also an FLTS except for the refusal set assigned to each state, we simply ignore the refusal set when computing P#Gs.

If the first condition is violated at a state (p, q), then (p, q) is marked BD0; if the second condition is violated, then (p, q) is marked BD1. We call a state *fault state* if it is marked either BD0 or BD1. Each state (p, q) contains a set of pointers pointing to its parent states (this will be used in Step 3). A variable FS is used to hold a set of pointers that point to the fault states, which will also be used in Step 3. If no state is marked either BD0 or BD1, then $P \angle S$. Otherwise, the markings will be used in Step 3 to generate a diagnostic information *graph* by Dia-Info(M, FS), which removes all the states and related transitions in M that can not reach any fault state without visiting the initial state. Hence, the final result contains all the traces from (p0, q0') to the fault states. The diagnosis information graph has the power of Intuitionistic Hennessy-Milner Logic used in [3] and [7].

Algorithm-DiaRed(P, S)

Input: Protocol specification $P = (Q_p, \Sigma_p, \delta_p, p0)$ and service specification $S = (Q_S, \Sigma_S, \delta_S, q0)$.

Output: report $P \angle S$, or a diagnosis information graph.

Var: FS a set of pointers to states;
/*each of the pointer points to a state*/
p, q states in P, S or Gs;
(p, q) state of P#Gs;
e an event;
Ref a set of events;

Begin

1) Computing Ref(p) for every state p in P by Algorithm-Ref, and compute the RG $Gs = (Qs', \Sigma_S, \delta s', R, q0')$ of S by Algorithm-RGraph;
2) let $FS = \phi$, computing $M = P\#Gs = (Q, \Sigma, \delta, (p0, q0'))$: the following two conditions are checked for each state (p, q) of P#Gs during the computation:
   a) if $p-e\rightarrow$ in P but $q - e \not\rightarrow$ in Gs for $e \neq \tau$, then mark (p, q) BD0 and create a pointer in FS which points to (p, q); otherwise
   b) if there is at least one refusal set $Ref \in R(q)$ such that $Ref(p) \subseteq Ref$, then mark state (p, q) BD1 and create a pointer in FS which points to (p, q);
3) If $FS = \phi$, then report $S \angle P$; otherwise compute Dia-Info(M, FS).

End

Dia-Info(M, FS)

Begin

1) Mark (p0, q0') PF;
   /*a PF marks a state that can reach a fault state*/
2) While $FS \neq \phi$ do the following
   a) Take a pointer from FS that points to a fault state (p, q) in M;
   b) Checkparent(p, q);
3) Remove from M all the states that are not marked PF;

End

Checkparent(p, q)

Begin

Mark (p, q) PF;
For each parent state (p', q') of (p, q):
if (p', q') is not marked PF then Checkparent(p', q')
Return

End

Theorem 2: $P \angle S$ iff $FS = \phi$ at the end of Step 2 of Algorithm-DiaRed.

Proof: 1) If $P \angle S$, then $Tr(P) \subseteq Tr(S)$. Hence, no state will be marked BD0 by Step 2a of the algorithm. For every state (p, q) in M, there is a trace $t \in Tr(P) \cap Tr(S)$ such that $p0 = t \Rightarrow p$ and $q0 = t \Rightarrow q$ from the definition of # product. From the definition of reduction relation, there is at least one refusal set $Ref \in R(q)$ such that $Ref(p) \subseteq Ref$. Therefore, no state will be marked BD1. That is, $FS = \phi$.

2) If $FS = \phi$ at the end of step 2 of Algorithm-DiaRed, then $Tr(P) \subseteq Tr(S)$ from step 2a. Since $Tr(P) \subseteq Tr(S)$, for every $t \in Tr(P) \cap Tr(S)$ there must be a state (p, q) such that $p0 = t \Rightarrow p$ and $q0 = t \Rightarrow q$. If no state will be marked BD1, then for every state (p, q) in M, there is at least one refusal set $Ref \in R(q)$ such that $Ref(p) \subseteq Ref$. Henceforth $P \angle S$.

Proposition 1: The time and space complexity of Algorithm-DiaRed(P, S) is $O(|Qp| \times |Qs'| \times |Qs|)$ in the worst case.

This can be proved by the fact that the main computation of the algorithm is in step 2. The number of states of P#Gs is at most $|Qp| \times |Qs'|$. For any state (p, q) of P#Gs, q contains at most |Qs| refusal sets. Therefore, the computation needed in step 2 is $O(|Qp| \times |Qs'| \times |Qs|)$.

In the worst case $|Qs'|$ has $2^{|Qs|}$ states. However, if Qs is small (this is true since a service specification is much smaller than its implementation specification) or the service specification is "less nondeterministic", the number of states of Gs will not be large.

3.3. Verification and Diagnosis of Testing Equivalence

From the definition of the testing equivalence, it is very easy to construct an algorithm to verify the testing equivalence and generate diagnosis information from the algorithm proposed in the last section: it simply combines Algorithm-DiaTe(P, S) with Algorithm-DiaTe(S, P) to verify $P \angle S$ and $S \angle P$, respectively.

Algorithm-DiaTe(P, S)

Input: Protocol specification $P = (Q_p, \Sigma_p, \delta_p, p0)$ and service specification $S = (Q_S, \Sigma_S, \delta_S, q0)$.

Output: report $P \sim S$, or a diagnosis information graph.

Var:
- FS1, FS2 a set of pointers to states; /*each of the pointer points to a state*/
- p, q states in P, S or Gs;
- (p, q) state of P#Gs or S#Gp;
- e an event;
- Ref a set of events;

Begin

1) Computing Ref(p) for every state p in P and S by Algorithm-Ref;
2) computing the refusal graphs, $Gp = (Q_p', \Sigma_p', \delta_p', Rp, p0')$ and $Gs = (Q_S', \Sigma_S', \delta_S', Rs, q0')$, of P and S by Algorithm-RGraph, respectively;
3) let $FS1 = \phi$, compute $M = P\#Gs = (Q1, \Sigma, \delta1, (p0, q0'))$: the following two conditions are checked during the computation for any state (p, q) of P#Gs:
   a) if $p - e \rightarrow$ in P but $q - e \not\rightarrow$ in Gs for $e \neq \tau$, then mark (p, q) BD0 and add a pointer to FS which points to (p, q);
   b) if there is at least one refusal set $Ref \in R(q)$ such that $Ref(p) \subseteq Ref$, then mark state (p, q) BD1 and create a pointer in FS which points to (p, q);
4) let $FS2 = \phi$, compute $M2 = Gp\#S = (Q2, \Sigma, \delta2, (p0', q0))$: the following two conditions are checked during the computation for any state (p, q) of M2:
   a) if $q - e \rightarrow$ in S but $p - e \not\rightarrow$ in Gp for $e \neq \tau$, then mark (p, q) BD0 and add a pointer to FS" which points to (p, q);
   b) if there is at least one refusal set $Ref \in R(q)$ such that $Ref(p) \subseteq Ref$, then mark state (p, q) BD1 and create a pointer in FS2 which points to (p, q);
5) If $FS1 = \phi$ and $FS2 = \phi$, then Report $S \sim P$; otherwise, if $FS1 \neq \phi$ then compute Dia-Info(M1, FS1), and if $FS2 \neq \phi$ then compute Dia-Info(M2, FS2).

End

From proposition 1, we have the following result.

Proposition 2: The time and space complexity of Algorithm-DiaTe is $O(|Qp| \times |Qp'| \times |Qs|)$ in the worst case.

The proof of this proposition is obvious from proposition 1 by considering that $O(|Qp| \times |Qs'| \times |Qs|) << O(|Qp| \times |Qp'| \times |Qs|)$, where << means "much smaller".

4. Example

Given an implementation P and a service specification S as shown in Fig.2(a) and (b), respectively, we need 1) to verify whether $P \angle S$; 2) if $P \angle S$ is not true, to generate diagnosis information.

We obtain the RG Gs of S as shown in Fig.3(a), in which the characters in { } beside a state q represent the refusal set R(q) of q. Fig.3(b) shows the results of Step 2, in which each state is named by two digits to represent a state (p, q) of P#Gs. Algorithm-DiaRed finds two fault states: 95 and 75. For state 95, there is not a refusal set Ref in R(5) such that $Ref(9) \subseteq Ref$. Therefore, state 95 is

marked BD1. For state 75, there is an event e can be enable at state 7 in P, but no such an event can be enabled at state 5 of Gs. Therefore, state 75 is marked BD0. Fig.(3b) is also the result of Step 3 in this example.

We modify the specification P as shown in Fig.4(a). Applying the algorithm to Fig.4(a) and Fig.2(b), we have the result depicted in Fig.4(b). It is clear that the modified implementation specification is a reduction of the service specification S.

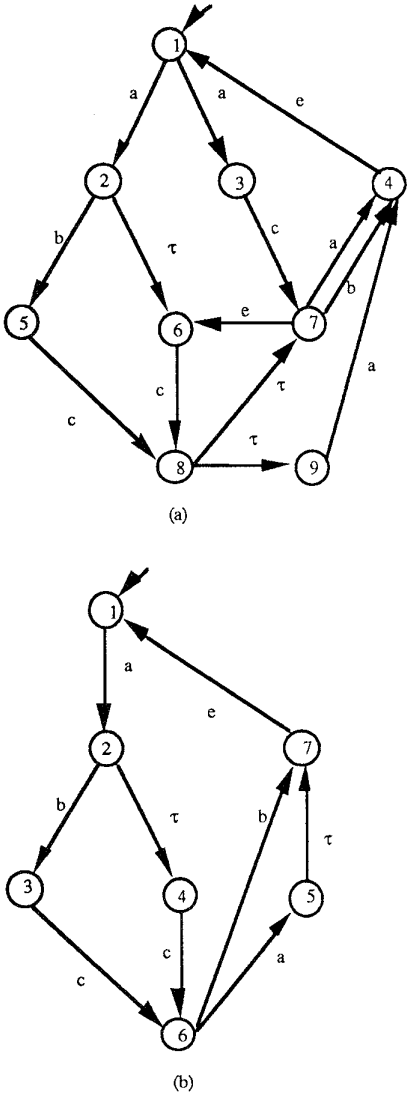


Fig.2 (a) The implementation P; (b) The service S.

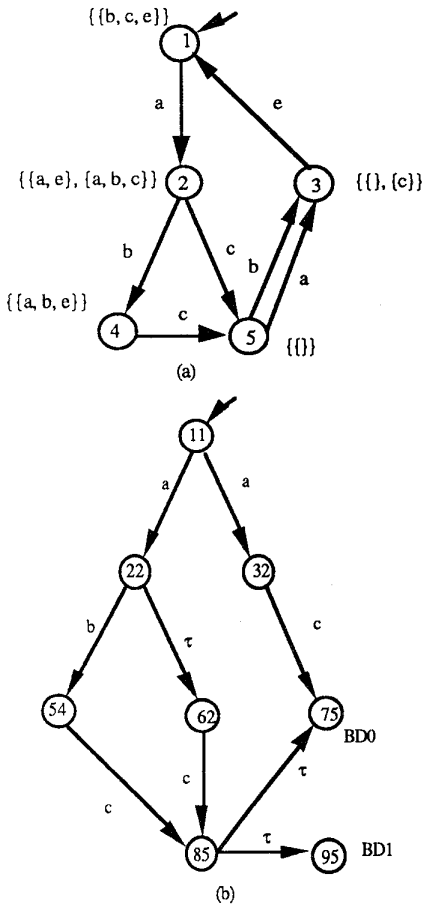
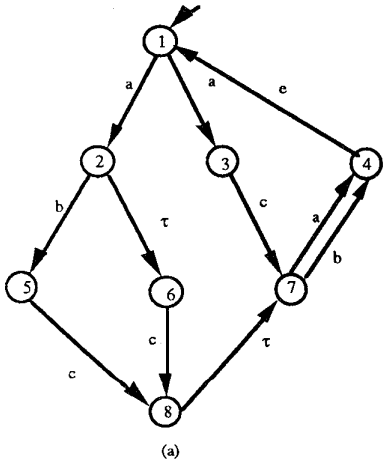


Fig.3 (a) The RG of P; (b) The output of the algorithm.



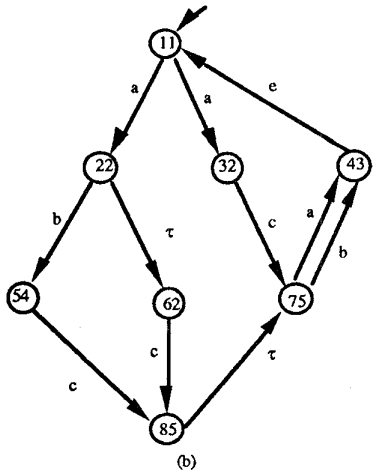


Fig.4 (a) the modification of P; (b) The result.

5. Discussions and Conclusions

In this paper, we have presented two efficient algorithms: the first one, Algorithm-DiaRed, is used for generating diagnostic information if an implementation is not a reduction of its service specification (i.e., the two specifications do not satisfy the reduction relation [1]). The basic method of verifying the reduction relation is to first transform the service specification into a *refusal graph*, then to check and record any violation of the two conditions given in Theorem 1 for each state of the coupled product of the refusal graph and the implementation specification P. The second algorithm, Algorithm-DiaTe for testing equivalence, is based on the one for the reduction relation. The advantages of our method are:

1) The time and space complexity of Algorithm-DiaRed is $O(|Qp|\times|Qs'|\times|Qs|)$. For the related work proposed in [3], the time and space complexity is $O(|Qp|\times|Qp'|\times|Qs'|\times|Qs|)$ (note that in [3] it claimed that the computation needed is $|Qp'|\times|Qs'|$, this is not correct since during the computation of bisimulation of two acceptance graphs, each state of the acceptance graph contains a set of acceptance sets in dimension $|Qp|$ and $|Qs|$ in the worst case. In addition, the computation of the postprocessing step is not counted there). Therefore, compared with the time and space complexity of the method proposed in [3], Algorithm-DiaRed may save a factor of $O(|Qp'|)$ in time and space in the worst case. Since $|Qp'| = 2^{|Qp|}$ in the worst case, the improvement of our algorithm is significant. In addition, our algorithm does not need a complex postprocessing algorithm.
2) The time and space complexity of Algorithm-DiaTe is $O(|Qp|\times|Qp'|\times|Qs|)$ in the worst case. For the related work proposed in [2], the time and space complexity is $O(|Qp|\times|Qp'|\times|Qs'|\times|Qs|)$ (with the same reason discussed above). Our algorithm, Algorithm-DiaTe, may save a factor of $O(|Qs'|)$ in time and space complexity.
3) Both Algorithm-DiaRed and Algorithm-DiaTe are able to generate all fault information. However, the algorithm proposed in [2] is not capable of producing diagnosis information. The algorithm proposed in [3] can only produce the information for a single fault.

Reference

[1] E. Brinksma, "A Theory for the Derivation of Tests", in Proceedings of IFIP Workshop PSTV, 1988.

[2] R.Cleaveland and M. Hennessy "Testing Equivalence as a Bisimulation Equivalence" in Formal Aspects of Computing, Vol.5, No.1, 1993.

[3] U. Celikkan and R. R.Cleaveland "Computing Diagnostic Tests for Incorrect Processes" in Proceedings of PSTV, XII, 1992.

[4] Harry R. Lewis, *Elements of the Theory of Computation*, Printice-Hall, 1981, pp59-pp62.

[5] Kanellakis, P.C. and Smolka, S.A. "CCS Expressions, Finite State Processes, and Three Problems of Equivalence." in Information and Computation, Vol.86, No.1, may 1990, pp. 43-68.

[6] Rocco De Nicola, "Extensional Equivalencies for Transition Systems", Acta Informatica 24, 1987.

[7] U. Celikkan "Generating Diagnostic Information for Behavioral Preorders" in proceedings of Computer-Aided Verification, 1991.

[8] F. Khendek, G.v. Bochmann "Merging Behavior Specifications" Publication #856, Dept. IRO, Universite de Montreal.