National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

A Unified Approach for Equivalence Relations of Indeterministic Distributed Systems

(with application to network protocols)

by

Guoqiang Wang

A M.Sc. Thesis

Submitted to the school of Graduate studies and Research
in partial fulfilment of the requirements for the
Master of Computer Science Degree*

University of Ottawa
Ottawa, Ontario
Canada

*The Master of Computer Science Program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute for Computer Science

ISBN 0-315-80003-8

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ABSTRACT

Equivalence relations are criteria for comparing the behavior of systems. They have important applications in the verification and testing of communications protocols. This thesis presents our investigation on equivalence relations in two parts. Part One contains mainly the theoretical results. It first gives a survey on many important equivalence relations. It then proposes new types of reachability for indeterministic distributed systems and a unified definition which reveals the essential and common features of many equivalence relations. This uniform approach points out that the various equivalence relations differ from one another mainly in the domain of the action sequences, the types of reachability and the observable properties of the reached sets of states. Exhibited behavior equivalence relation is redefined and a new polynomial time algorithm for its verification is presented. Three reduction methods among the equivalence relations are discussed and a number of results on property-preserved transformations are given. A proof that EB equivalence is stronger than testing equivalence in non-strongly convergent labeled transition systems is presented. A disproof of the conclusion existing in literature that EB equivalence is stronger than weak equivalence is provided. Part Two reports our implementation of the various verification algorithms in a system called LTS-EVS in the Sun Workstation.

ACKNOWLEDGEMENT

I am very grateful to my supervisor, Dr. To-yat Cheung, for his valuable time, patience and for his guidance and advice throughout my graduate studies. His instruction for revising the drafts of the thesis has greatly improved its contents and presentation. The many discussions we have had have been of great influence and assistance.

I acknowledge with gratitude the financial support provided by the Telecommunications Research Institute of Ontario and the Natural Sciences and Engineering Research Council of Canada.

I would like to thank the Protocol Research Group for providing an excellent research environment for the entire period of my study leading to this thesis. In particular, discussions with members of the Group, especially Xinming Ye and Youwen Wu, have turned out to be extremely helpful.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION AND FUNDAMENTALS

1.1 VERIFICATION OF INDETERMINISTIC DISTRIBUTED SYSTEMS

As communications systems and their protocols are becoming more and more complicated, informal methods are no longer adequate for reducing the large number of errors bound to arise in their design and implementation. Many formal description techniques have been developed for their specification, verification and testing.

Verification is the process of proving that distributed systems are "error-free" in their design. Several approaches have been developed for such purposes. For example, reachability analysis is used for checking logical correctness, such as the absence of deadlocks, unspecified receptions and non-executable transitions; invariance analysis is used for finding place invariants, etc. But, these approaches are not suitable for dealing with another objective of verification, namely, proving the similarity and dissimilarity of behavior of two systems. To do this requires a drastically different technique, especially if the systems exhibit indeterministic behavior.

Indeterminism is introduced into a specification partly for allowing the specification to have more independency from the implementational details. But, it makes verification very complicated and difficult since some of the behaviors of the system are not observable and controllable. One way for comparing such systems is to provide a model for describing their indeterministic behavior and to define some criteria for their comparison. For example, we can specify the observational behavior of a protocol and its service by two labeled transition systems and use an equivalence relation as a criterion for comparing their behavior. This is the same as proving the equivalence between their specifications. Equivalence relations have also been applied to conformance testing of protocols [BRI89]. In fact, Conformance testing can be considered as the process of proving the equivalence of an implementation under test and a specification based on which test sequences are generated.

Many models, such as CCS (Calculus of Communicating Systems) [MIL80], LOTOS (Language fOr Temporal Ordering Specification) [ISO8807] and IPN (Indeterministic Petri Net) [CHE90b], have been developed for describing the indeterministic behavior of distributed systems and facilitating the verification of their design. Based on these models, many equivalence relations have been given [MIL80, DEN84, POM86, BOL87b, BOL89] for comparing the behavior of the systems. Different kinds of equivalent relations capture different properties of the specifications. But, the definitions of these equivalence relations lack a sense of uniformity. This thesis attempts

to give a uniform approach for redefining them and provide a deeper understanding of their characteristics.

1.2 INDETERMINISM, OBSERVATION AND EQUIVALENCE RELATIONS

In this section, we give an example in CCS [MIL80] to show how the observable and indeterministic internal behaviors of a system are described and observed and how the criteria for comparison are represented.

Three systems S, T and R are shown in Figure 1.1 as black boxes for testing. An observer wants to investigate their behaviors by asking them to accept actions one at a time. All of them can execute the observable actions a, b, c and d. Each box has four buttons, one for each of the four actions. There are four atomic experiments an observer can do, one for each action. For example, doing an a-experiment on a box involves pressing the a-button (at their starting states s_0, t_0 and r_0, respectively), with two possible outcomes :

1) Failure - if the button is locked.

2) Success - if the button is unlocked and goes down (and a state transition occurs internally).

These two outcomes describe the possible deadlock situations at any state of the system. Two states p and q are said to have the same deadlock situation if and only if, for any observable action x, the outcome of the x-experiemnt is either failure or success at both of them.

In this example, we cannot distinguish between S and T at their initial states s_0 and t_0 by any single atomic experiment. At both states, the a-experiment succeeds and the other three experiments fail. After a successful a-experiment, S moves to state s_1 and T moves to either state t_1 or state t_2. Suppose T moves to t_2. We may try another atomic experiment to see if the systems are equivalent or not. Clearly, a b-experiment now succeeds for S but fails for T, though the other three experiments fail to distinguish them. However, if T moves to t_1, we still cannot distinguish between S and T.

Next, let us conduct several b-experiments on S and T at these states. We find that S's b-button is always unlocked, but that T's b-button is sometimes locked and sometimes unlocked. This means that, after an a-experiment, S is not deadlockable at button b but T is deadlockable at button b. Hence, we can conclude that S and T are not equivalent at these states as far as their deadlock behavior is concerned.

Let us now compare S and R. After an a-experiment, S moves to state s_1 where both the b-experiment and c-experiment will be successful. However, R may move to state r_2 or state r_1 depending on whether it has executed an internal action τ or not. If R has not executed the internal action τ and stays at state r_1, a c-experiment may find the c-button sometimes locked but eventually unlocked (eventually, because although R may take a little time to decide on its internal action τ, it

will do so since no b-expertiment is attempted). In this way, we cannot distinguish between S and R. On the other hand, suppose R has executed the internal action τ and moved to state r_2, a b-experiment on R may find the b-botton locked. Hence, we can conclude that after an a-experiment, S is not deadlockable at button b but R is deadlockable at button b. Hence, S and R are not equivalent.

Through this example, we see that the concepts of reachability and deadlock can be used to describe an equivalence relation. That is, two systems are equivalent under observation if the following statement is true of both or neither of them: for a given sequence of actions $a_1 a_2 ... a_n a_{n+1}$, it is possible to do an $a_1 a_2 ... a_n$-experiment and reach a state where an a_{n+1}-experiment is either a success or a failure. Two systems with different deadlock situations are not considered to be equivalent.

From Figure 1.1, we see that there are two kinds of indeterminism existing in these systems. One is due to a multiple choice of external actions, just as T at state t_0. Another is due to internal actions, just as R, where there is an internal action τ from state r_1 to state r_2. The existence of different forms of indeterminism makes the system states possess different deadlock situations and thus makes the comparison of systems more complicated.

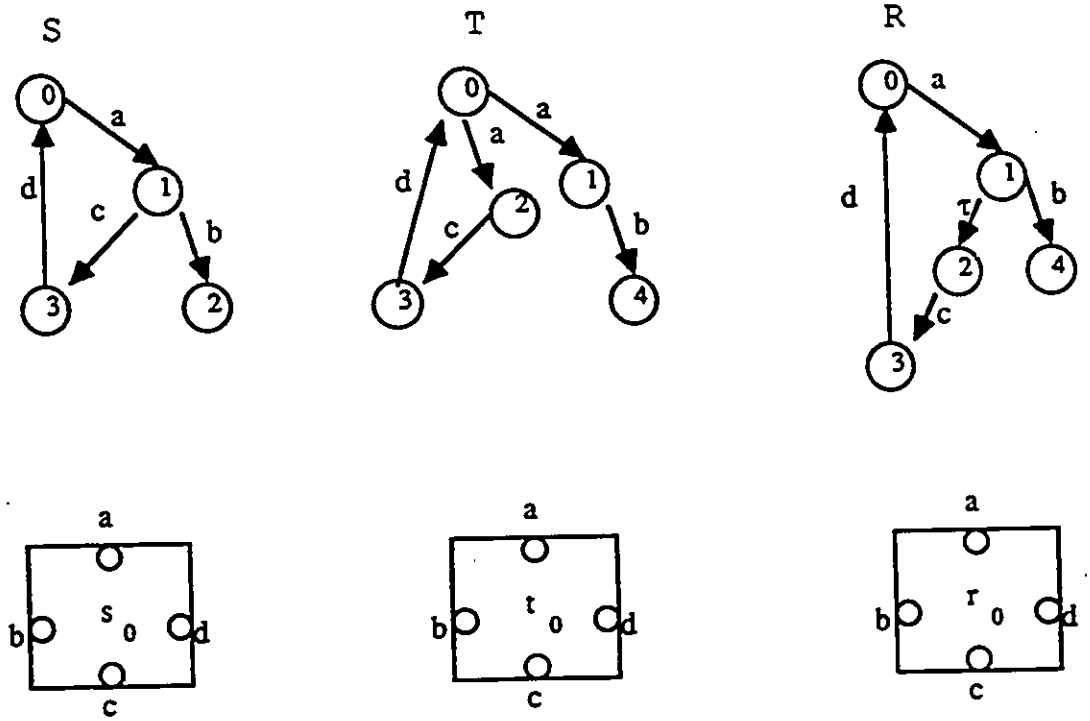


Figure 1.1 Three systems which are not observationally equivalent.

1.3 MOTIVATION AND CONTRIBUTIONS OF THE THESIS

As the design of distributed systems becomes more and more complicated, more sophisticated methods for their verification are required. However, such advanced methods require deep understanding of the behavior of the systems. Equivalence relations are rigorous means for studying their behavior. R.Milner [MIL80] proposed the notion of equivalence based on an observer and on observable/unobservable actions. Since then, other proposals have appeared [DEN84, CIN85]. But, in the literature, the concepts and definitions of equivalence relations are not very well presented and are often uncoordinated, i.e., without a unified view. Some results are even erroneous. Also, a recent trend in protocol research is to develop tools for experimental investigation. Very few tools have been reported for systematically applying these equivalence relations for verifying LOTOS specifications and CCS. We know of only two application systems for verifying the equivalence relations of LOTOS specifications and one system for verifying CCS processes. One of them is called SQUIGGLES [BOL89]. Some of whose features are listed below:

1) It can verify strong, weak and testing equivalence.
2) The verification algorithms are implemented according to the partitioning algorithm of Paige and Tarjan [PAI87].
3) The behavior graph is obtained from the text of basic LOTOS.

Another system [SHI89] defines a quasi-strong bisimulation equivalence and implements a verification method based on the partitioning algorithm of [AHO74] without any implementational details. The third system [MIC87] is implemented in PROLOG and is capable of deriving a bisimulation containing a given pair of CCS-processes in case they are equivalent.

In this thesis, based on the concepts of reachability and deadlock, we propose a uniform approach for redefining the various equivalence relations and show that they differ from one another only in the types of reachability and in the observable properties of the reached sets of states. Based on this common view, we show how they are used to compare the behavior of systems. Verification algorithms and transformation rules are then explored. Experimental works have also been done. Original contribution of this thesis can be divided into two parts:

Part I (theoretical results):

1. We define new types of reachability for indeterministic labeled transition systems. Based on these types of reachability, we give a new view of the various equivalence relations and redefine exhibited behavior equivalence (Chapter 2).
2. We propose a unified definition of equivalence relation which captures the essential and common features of the various equivalence relations. This unified approach shows that

the various equivalence relations differ from one another in their types of reachability and in the properties of the reached set of states for observation (Chapter 4).

3. We provide a proof that EB equivalence is stronger than testing equivalence in non-strongly convergent labeled transition systems and provide a disproof of the conclusion existing in literature that EB equivalence is stronger than weak equivalence [POM86] (Chapter 4).
4. We propose a rule for transforming one labeled transition system to another so that the verification of EB equivalence can be reduced to the verification of weak equivalence. We derive many properties which are satisfied and preserved under different transformations (Chapter 3).
5. We propose a new polynomial algorithm for verifying EB equivalence (Chapter 5).

Part II (experimental results):

6. We have implemented an equivalence verification system (LTS-EVS) which has the following features (Chapter 6):
   a. It can verify strong, weak, exhibited behavior and testing equivalence relations. The verification algorithms are based on the partitioning algorithms of [KAN83].
   b. LTS-EVS can be either used as an independent tool or integrated into UO-GLOTOS [CHE89A], a graphical system for research in the specification, verification and testing of LOTOS. LTS-EVS serves the following two purposes : i) To provide a prototype for commercial applications. ii) To provide a tool for further research.

The rest of the thesis is organized as follows. Chapter 2 includes a review of many equivalence relations. New types of reachability are introduced and exhibited behavior equivalence is redefined. This chapter provides the basis for the whole thesis. Chapter 3 discusses the reduction between equivalence relations. Also, the transformations and the properties preserved under these transformations are discussed in this chapter. Based on the different types of reachability defined in Chapter 2, Chapter 4 provides a unified definition for the various equivalence relations. This uniform approach shows that these relations differ from one another only in the types of reachability and in the properties of the reached sets of states for observation. Chapter 5 presents the algorithms for verifying these equivalence relations. Chapter 6 describes part of the implementation details of these algorithms in a system called LTS-EVS. In Chapter 7, besides summarizing and evaluating the work of our study, some future work is pointed out.

Chapter 2

REVIEW ON EQUIVALENCE OF DISTRIBUTED SYSTEMS

2.1 INTRODUCTION

Among the many problems of verifying distributed systems, one is to show whether two systems have the same behavior under a certain criterion for observation. For example, to verify that a protocol does provide the required service, one can prove the equivalence of the specifications of the protocol and service. For conformance testing, one can prove the equivalence of a specification and the system under test. Different equivalence relations, such as strong equivalence, weak equivalence, exhibited behavior equivalence and testing equivalence, capture different aspects of the behavior of the specified systems.

In this chapter, we review the terminology and concepts of the most important equivalence relations [MIL80, DEN84, KAN83, POM86, BOL87b, BOL89]. In addition, this chapter includes our following original contributions:

1) We define different types of reachability for indeterministic labeled transition systems.
2) Based on these definitions of reachability, we give a new view of the various equivalence relations.
3) We redefine exhibited behavior equivalence in terms of a labeled transition system.

2.2 LABELED TRANSITION SYSTEMS

The model used for formulating a behavioral equivalence relation plays an important role in the design of strategies and tools for the verification and analysis of the specified systems, especially in the case where the semantics of the specification language is based on such notions. For example, deterministic finite state machines are mostly used for describing deterministic systems without differentiating between external and internal actions. A labeled transition system is a more flexible model. For example, it is used as a semantic model for describing the behavior of a system specified in LOTOS (Language fOr Temporal Ordering Specification), a formal description technique standardized by [ISO8807]. A property between two systems specified in LOTOS can be verified by proving that some relation holds between the two LTSs representing the two specifications.

In the following, we first give a formal definition of a labeled transition system.

Definition 2.1 (labeled transition system)

A *labeled transition system* (LTS) is a quadruple $< K, \Sigma, \Delta, p_0 >$, where

K is a countable set of states;

Σ is a countable set of actions, including the special action τ called *internal action*;

Δ is a transition relation in $K \times \Sigma \times K$; and

$p_0 \in K$ is the initial state.

A *finite labeled transition system* is an LTS in which K and Σ are both finite.

In this thesis, we are concerned with finite LTSs only. From an observer's point of view, the symbol τ represents an unobservable action. It corresponds to an internal step of the system and is analogous to the empty move denoted by the label ε in classical automata theory.

For the rest of this chapter, all terminology and notation are based on the labeled transition system $LTS = < K, \Sigma, \Delta, p_0 >$.

Definition 2.2 (some notation)

$\Sigma_{obs} = \Sigma - \{ \tau \}$ denotes the set of observable actions.

Σ^+ denotes the set of finite nonempty sequences from Σ.

$\Sigma^* = \Sigma^+ \cup \{ \varepsilon \}$, where ε denotes the empty sequence.

$O_S(p) = \{ a \mid a \in \Sigma_{obs}, \exists p' \in K$ such that $(p, a, p') \in \Delta \}$ denotes the set of *strongly observable actions* at state $p \in K$.

Definition 2.3 (unobservable state)

A state p is said to be *unobservable* iff $p \neq p_0$, $O_S(p) = \varnothing$ and there exists at least one $p' \in K$, such that $(p, \tau, p') \in \Delta$.

At an unobservable state, all actions departing from it are internal. The initial state p_0 and all terminal states (i.e., states which do not have any departing transitions) are not considered as unobservable. This is based on the assumption that it is possible to observe the beginning and ending of a normal execution of the system. Such an assumption is consistent with similar assumptions used in the literature [MIL80].

Definition 2.4 (set of observable states)

$K_O = K - \{ p \mid p$ is an unobservable state, $p \in K \}$ is called the *observable subset* of K.

An equivalence relation is used as a criterion for comparing certain properties of the states of an LTS. In the literature, the various equivalence relations are defined separately, i.e., without a common theme or approach. In this chapter, we make some observations on these definitions and come up with three common features based on which these equivalence relations can be defined.

The process of verifying the equivalence of two states p and q is as follows: For every action sequence s, execute s starting at p and q, separately. Then, compare the states reached from p and those from q.

Equivalence relations can be distinguished according to the following three factors:

1) Which states are reached and how - this leads us to define different types of reachability (Section 2.3).
2) What properties (e.g., observable or unobservable) the reached states have.
3) Whether the states reached from p are compared with those reached from q individually or collectively.

In the following, we shall see how these factors influence our definition of equivalence relations.

2.3 REACHABILITY IN LABELED TRANSITION SYSTEMS

In this section, we define three types of reachability which we think are new in the literature. They depend on the composition (purely observable or a mixture of observable and unobservable actions) of the action sequence. The first two types, strong reachability --> and weak reachability ==>, differ from the third type, exhibited behavior reachability $\Longrightarrow$, in that the third type is also defined in terms of the observability of the reached states.

Definition 2.5 (-->, strong reachability)

For two states p and $q \in K$ and an action $a \in \Sigma$, q is said to be *strongly-reachable* from p by a, denoted as $p -- a --> q$, iff $(p, a, q) \in \Delta$.

For two states p and $q \in K$ and a sequence of actions $s = a_1a_2...a_n$ $(n \geq 1) \in \Sigma^+$, $p -- s --> q$ means that there exists a sequence of states $p_1...p_{n-1}$ such that $p -- a_1 --> p_1$, ..., $p_{n-1} -- a_n --> q$.

Definition 2.6 (==>, weak reachability)

For two states p and $q \in K$ and an action sequence $s \in \Sigma_{obs}^*$, q is said to be *weakly-reachable* from p by s, denoted as $p == s ==> q$, iff

1) if $s = \varepsilon$ (the empty sequence), there exists a sequence of states $p_1...p_{n-1}$ such that that $p -- \tau --> p_1, ..., p_{n-1} -- \tau --> q$; or

2) if $s = a_1 a_2 \ldots a_n$, there exists a sequence of states $p_1 p_2 \ldots p_{2n}$ such that

$p == \varepsilon ==> p_1 -- a_1 --> p_2 == \varepsilon ==> p_3 -- a_2 --> p_4 \ldots\ldots p_{2n-1} -- a_n --> p_{2n} == \varepsilon ==> q$.

Note that it is always true that $p == \varepsilon ==> p$ for any state p.

Definition 2.7 (≡=>, exhibited behavior reachability)

For two states p and $q \in K$ and an action sequence $s \in \Sigma_{obs}^*$, q is said to be *EB-reachable* from p by s, denoted as $p \equiv\equiv s \equiv\equiv> q$, iff $q \in K_O$ and $p == s ==> q$, where K_O is the observable subset of K.

For an action sequence s, the three types of reachability defined above can be distinguished as follows: Let PATH be the action sequence leading from state p to state q. In general, PATH is composed of segments of observable actions and segments of internal actions. For strong reachability, these actions, observable or internal, must be explicitly and definitely specified in s. As a result, PATH and state q are quite rigidly specified, except when some of the observable or internal actions in s can lead to different states. For weak reachability, however, only the observable actions of PATH are explicitly specified in s. The segments of internal actions are quite flexible. As a result, a wider scope of states will be reached. EB reachability is the same as weak reachability but imposes on the reached states the additional requirement that they must be observable. Hence, its scope of reached states is smaller than that for weak reachability.

Definition 2.8 ($p \sim s \sim>$, $p \sim s \sim>$ dead)

The notation $\sim s \sim>$ denotes either $-- s -->$ or $== s ==>$ or $\equiv s \equiv>$, as will be indicated from the context. Let $p \in K$ and $s \in \Sigma^+$ (or $s \in \Sigma_{obs}^*$). We say that $p \sim s \sim>$ if there exists a state p' such that $p \sim s \sim > p'$. Otherwise, we say $p \sim s \sim >$ dead.

Next, we extend the definitions of reachability from single states to sets of states.

Definition 2.9 (Reachable set of states)

For $P \subset K$, $s \in \Sigma^*$ (or $s \in \Sigma_{obs}^*$) and a reachability type $\sim\sim>$, the *set of states reachable from* P by s with respect to $\sim\sim>$ is defined as follows:

$Reach(P, s, \sim\sim>) = \{ p' \mid p' \in K, \exists\, p \in P$ such that $p \sim s \sim> p' \}$.

Definition 2.10

For $P, P' \subset K$, $P \sim s \sim> P'$ iff $P' = Reach(P, s, \sim\sim>)$.

Theorem 2.1

In general, for any $s \in \Sigma_{obs}^*$ and any P C K, Reach(P,s,-->) C Reach(P,s,$\Longrightarrow$) and Reach(P,s, $\Longrightarrow$) C Reach(P,s, $\Longrightarrow$). In particular, If Σ has no internal action τ, -->, $\Longrightarrow$ and $\Longrightarrow$ are all identical.

Proof: These results follow immediately from Definitions 2.5, 2.6 and 2.7. []

Example 2.1

This example illustrates the distinction among the three types of reachability defined above. Figure 2.1 shows an LTS with K = { 1,2,3,4,5,6,7,8,9 }, Σ = { a,b,c,d,τ }, Δ = { (1,a,2), (2,b,3), (2,τ,4), (4,b,5), (5,τ,6),(5,τ,7),(6,c,8),(7,d,9)} and $p_0 = 1$.

We have: Reach({1}, ab, –>) = {3}, Reach({1}, ab, $\Longrightarrow$) = {3,5,6,7} and Reach({1}, ab, $\Longrightarrow$} = {3,6,7}. Note that state 5 is unobservable and is therefore not EB-reachable from state 1. Next, we have: Reach({1}, a, –>) = {2}, Reach({1}, a, $\Longrightarrow$) = {2,4} and Reach({1}, a, $\Longrightarrow$} = {2,4}.

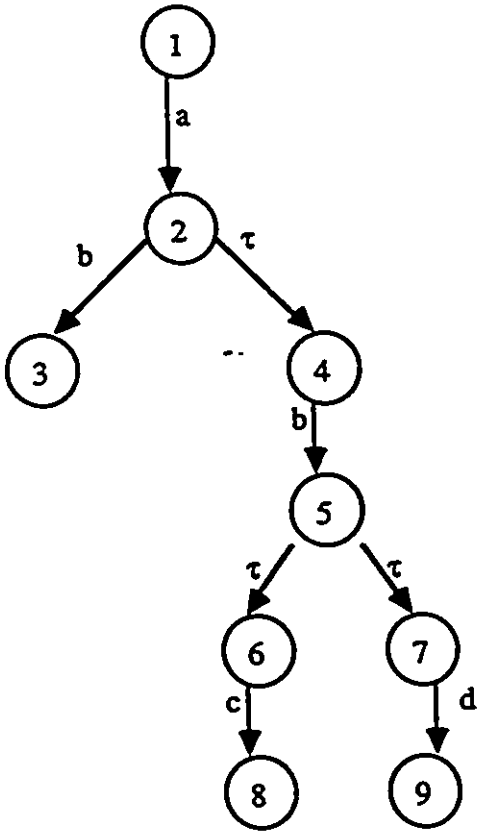


Figure 2.1 A labeled transition system for showing different types of reachability.

In the following sections, we review the definitions of several equivalence relations between two labeled transition systems LTS1 = < P, Σ, Δ_P, p_0> and LTS2 = < Q, Σ, Δ_Q, q_0 > and illustrate how the different types of reachability defined above can be used to characterize them.

2.4 STRONG OBSERVATIONAL EQUIVALENCE

The notion of strong observational equivalence (or simply "strong equivalence") is first defined in [MIL80]. Intuitively, strong equivalence of two LTSs requires them to be able to 'simulate' each other action by action, including both external and internal ones. Here, 'simulate' means that one LTS accepts whatever transition sequences the other one accepts. In other words, LTS1 and LTS2 are strongly equivalent if, starting from their initial states, they can perform the same sequences of actions (not necessarily observable) and then move to two strongly equivalent states.

Definition 2.11 ($\cong^S$, strong equivalence of two states) [MIL80, KAN83]

Let $p \in P$ and $q \in Q$.

i. $p \not\cong_0^S q$ is always true.

ii. $p \not\cong_k^S q$ iff, $\forall a \in \Sigma$, the following two conditions are true :

a) $\forall p' \in P$, if $p -- a --> p'$, then ($\exists q' \in Q : q -- a --> q'$ and $p' \not\cong_{k-1}^S q'$).

b) $\forall q'' \in Q$, if $q -- a --> q''$, then ($\exists p'' \in P : p -- a --> p''$ and $p'' \not\cong_{k-1}^S q''$).

iii. $p \cong^S q$ iff $p \not\cong_k^S q$ for all $k \geq 0$.

Definition 2.12 ($\cong^S$, strong equivalence of two LTSs)

LTS1 and LTS2 are *strongly equivalent*, denoted as LTS1 $\cong^S$ LTS2, iff $p_0 \cong^S q_0$.

In the literature, strong equivalence is also defined in terms of strong bisimulation [BOL89].

Example 2.2 [ELG89] (Figure 2.2)

LTS1 $\cong^S$ LTS2, as explained below: Since p_2, p_4, q_4, q_6 and q_8 are terminating states, we have $p_4 \cong^S q_4$, $p_4 \cong^S q_8$, $p_2 \cong^S q_2$ and $p_2 \cong^S q_6$. Since $\{p_3\} -- c --> \{p_4\}$, $\{q_3\} -- c --> \{q_4\}$ and $\{q_7\} -- c --> \{q_8\}$, $p_3 \cong^S q_3$ and $p_3 \cong^S q_7$. Also, since $\{p_1\} -- \tau --> \{p_3\}$, $\{q_1\} -- \tau --> \{q_3\}$ and $p_2 \cong^S q_2$, $p_1 \cong^S q_1$. Similarly, $p_1 \cong^S q_5$. Lastly, since $\{p_0\} -- a --> \{p_1\}$ and $\{q_0\} -- a --> \{q_1, q_5\}$, $p_0 \cong^S q_0$.

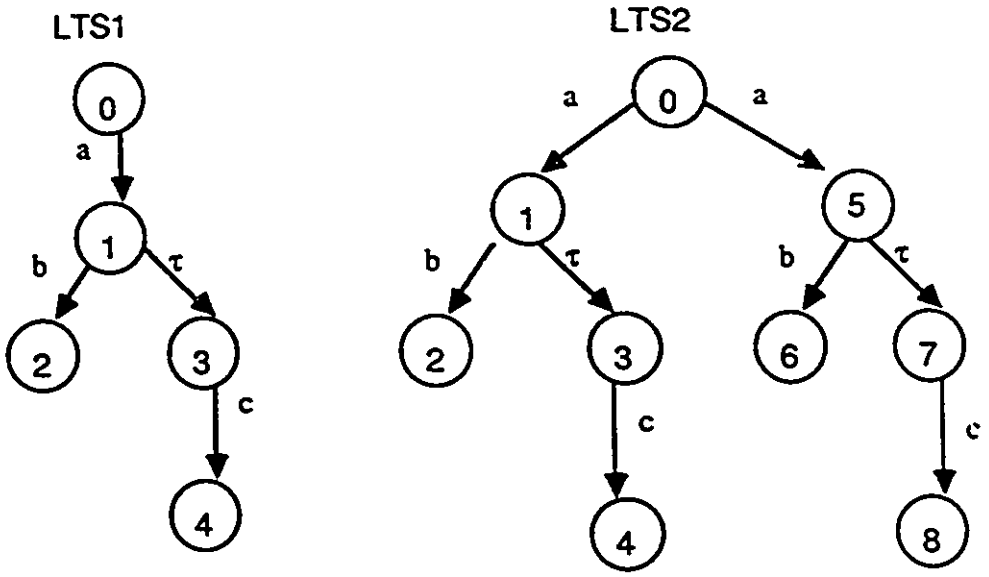


Figure 2.2 Two strongly equivalent LTSs.

2.5 WEAK OBSERVATIONAL EQUIVALENCE

According to Definition 2.11, we can see that the condition for strong equivalence is very restrictive for comparing the behaviors of LTSs. It requires that every action of one LTS, including both external and internal ones, must be simulated by an action of the other LTS. A less restrictive equivalence notion is weak equivalence [MIL80], which requires simulation only for their observable actions. Briefly, two LTSs are weakly equivalent if, starting from their initial states, they can perform exactly the same sequences of observable actions and then reach two weakly equivalent states.

Definition 2.13 ($\equiv^W$, weak equivalence of two states) [MIL80, KAN83]

Let $p \in P$ and $q \in Q$.

i. $p \not=_0^W q$ is always true.

ii. $p \not=_k^W q$ iff, $\forall a \in \Sigma_{obs} \cup \{\varepsilon\}$, the following two conditions are true :

a) $\forall p' \in P$, if $p = a \Longrightarrow p'$, then $(\exists q' \in Q : q = a \Longrightarrow q'$ and $p' \not=_{k-1}^W q')$.

b) $\forall q'' \in Q$, if $q = a \Longrightarrow q''$, then $(\exists p'' \in P : p = a \Longrightarrow p''$ and $q'' \not=_{k-1}^W p'')$.

iii. $p \equiv^W q$ iff $p \not=_k^W q$ for all $k \geq 0$.

Definition 2.14 ($\equiv^W$, weak equivalence of two LTSs)

LTS1 and LTS2 are *weakly equivalent*, denoted as LTS1 $\equiv^W$ LTS2, iff $p_0 \equiv^W q_0$.

In the literature, weak equivalence is also defined in terms of weak bisimulation [BOL89].

Example 2.3 [BOL89] (Figure 2.3)

LTS1 $\equiv^W$ LTS2, as explained below: Since p_2, p_4, q_2, q_4 and q_6 are terminating states, $p_2 \equiv^W q_2$, $p_4 \equiv^W q_4$ and $p_4 \equiv^W q_6$. Then, since $\{p_3\} == c ==> \{p_4\}$, $\{q_3\} == c ==> \{q_4\}$ and $\{q_5\} == c ==> \{q_6\}$, we have $p_3 \equiv^W q_3$ and $p_3 \equiv^W q_5$. Next, since $\{p_1\} == \varepsilon ==> \{p_3\}$, $\{q_1\} == \varepsilon ==> \{q_3\}$, $\{p_1\} == c ==> \{p_4\}$, $\{q_1\} == c ==>\{ q_4\}$, $\{p_1\} == b ==> \{p_2\}$ and $\{q_1\} == b ==> \{q_2\}$, we have $p_1 \equiv^W q_1$. Lastly, since $\{p_0\} == a ==> \{p_1, p_3\}$ and $\{q_0\} == a ==>\{q_1,q_3,q_5\}$, $p_0 \equiv^W q_0$.

However, LTS1 $\equiv^S$ LTS2 does not hold, because we have $\{p_0\}$ -- a --> $\{p_1\}$ in LTS1 and $\{q_0\}$ -- a --> $\{q_1,q_5\}$ in LTS2, but q_5 is not strongly equivalent to p_1.

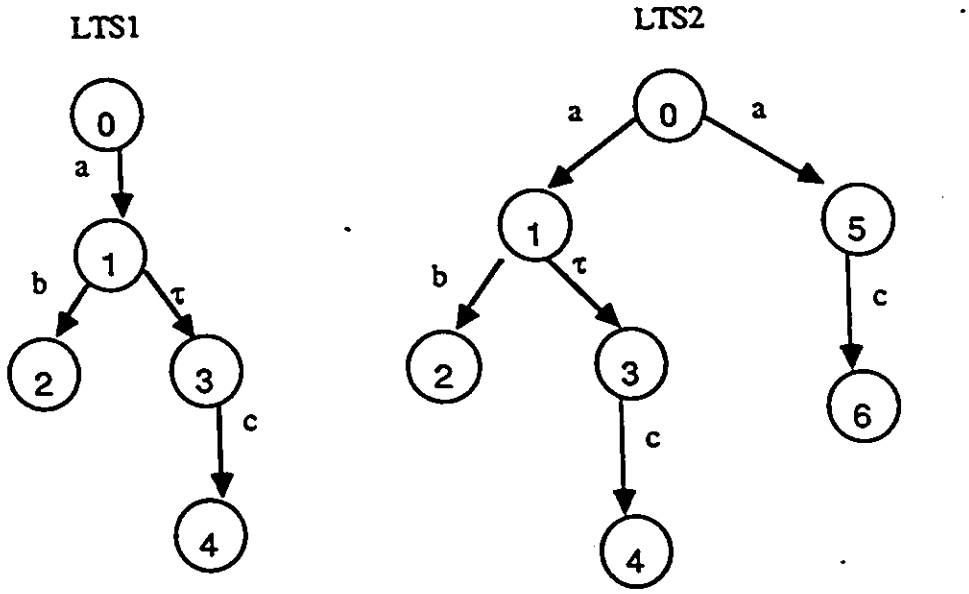


Figure 2.3 Two weakly equivalent LTSs which are not strongly equivalent.

2.6 EXHIBITED BEHAVIOR EQUIVALENCE

In strong equivalence and weak equivalence, both observable or unobservable states are reached and are compared. If the reached states are limited to the observable ones, we have the notion of exhibited behavior equivalence (or, in abbreviation, "EB equivalence") [CIN85, POM86]. Similar to weak equivalence, EB equivalence requires two LTSs to be able to "simulate" each other in their observable actions. In EB equivalence, only the observable reached states are compared while the unobservable states are never reached and are thus ignored. If two LTSs have no unobservable states, EB equivalence and weak equivalence are identical.

The original notion of EB equivalence was defined in terms of Petri nets [CIN85]. The author defined a firing rule for observable transitions such that their firing leads to markings which have at least one firable observable transition. The author also gave some transformation rules for proving EB equivalence of Petri nets [CIN85]. In the following, we redefine this concept in terms of LTSs and EB-reachability.

Definition 2.15 ($\equiv^{eb}$, exhibited behavior equivalence of two states)

Let $p \in P$ and $q \in Q$.

i. $p \not=_0^{eb} q$ is always true.

ii. $p \not=_k^{eb} q$ iff, $\forall a \in \Sigma_{obs} \cup \{ \varepsilon \}$, the following two conditions are true:

a) $\forall\ p' \in P$, if $p \equiv a \Longrightarrow p'$, then $(\exists\ q' \in Q : q \equiv a \Longrightarrow q'$ and $p' \not=_{k-1}^{eb} q')$.

b) $\forall\ q'' \in Q$, if $q \equiv a \Longrightarrow q''$, then $(\exists\ p'' \in P : p \equiv a \Longrightarrow p''$ and $q'' \not=_{k-1}^{eb} p'')$.

iii. $p \equiv^{eb} q$ iff $p \not=_k^{eb} q$ for all $k \geq 0$.

Definition 2.16 ($\equiv^{eb}$, exhibited behavior equivalence of two LTSs)

LTS1 and LTS2 are *EB equivalent*, denoted as LTS1 $\equiv^{eb}$ LTS2, iff $p_0 \equiv^{eb} q_0$.

Example 2.4 [POM86] (Figure 2.4)

LTS1 $\equiv^{eb}$ LTS2, as explained below: Since p_4, p_5, q_4 and q_5 are terminating states, $p_4 \equiv^{eb} q_4$ and $p_5 \equiv^{eb} q_5$. Furthermore, since $\{p_2\} \equiv b \Longrightarrow \{p_4\}$, $\{p_3\} \equiv c \Longrightarrow \{p_5\}$, $\{q_2\} \equiv b \Longrightarrow \{q_4\}$ and $\{q_3\} \equiv c \Longrightarrow \{q_5\}$, we have $p_2 \equiv^{eb} q_2$ and $p_3 \equiv^{eb} q_3$. Lastly, since $\{p_0\} \equiv a \Longrightarrow \{p_2, p_3\}$ and $\{q_0\} \equiv a \Longrightarrow \{q_2,q_3\}$, $p_0 \equiv^{eb} q_0$.

However, it is not true that LTS1 $\equiv^{w}$ LTS2, because we have $\{p_0\} = a \Longrightarrow \{p_1,p_2,p_3\}$ in LTS1 and $\{q_0\} = a \Longrightarrow \{q_2,q_3\}$ in LTS2, but neither $p_1 \equiv^{w} q_2$ nor $p_1 \equiv^{w} q_3$.

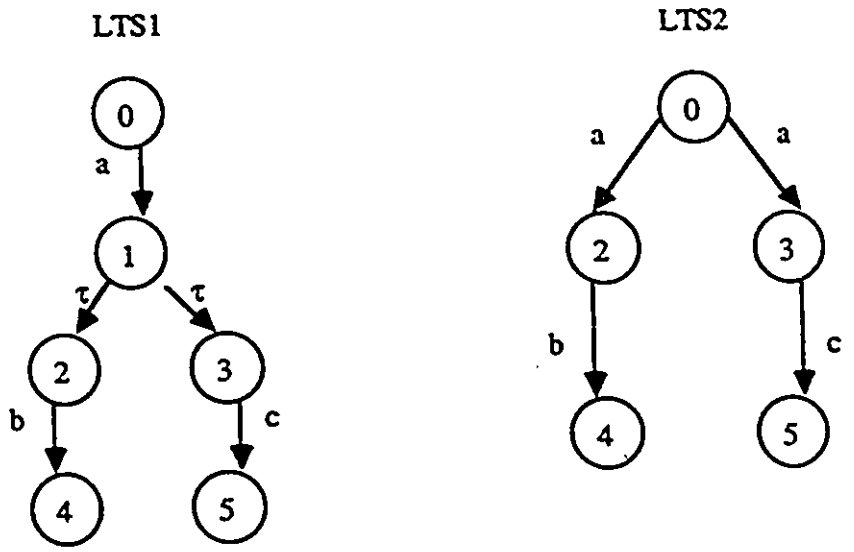


Figure 2.4 Two EB equivalent LTSs which are not weakly equivalent.

2.7 FAILURE EQUIVALENCE

Eb equivalence is defined on the basis of weak reachability and the observability of the reached states. In this section, we study a kind of equivalence, namely, failure equivalence, which is based on weak reachability and the refusing capability of the reached states.

Definition 2.17 (refusal set)

For $p \in K$ and $Z \subset \Sigma_{obs}$, Z is said to be a *refusal set* of p, if, for every $a \in Z$, $p == a ==>$ dead. Also, Refusal(p) = { Z | Z is a refusal set of p } denote the set of all refusal sets of p.

max-Refusal(p) is the largest set in Refusal(p).

Refusal(p) includes all observable actions which will lead p to deadlock .

Example 2.5 (Figure 2.5)

Let Σ_{obs} = {a, b, c} and state p_1 be the initial state of LTS. Refusal(p_1) = {{b}, {c}, {b,c}} and max-Refusal(p_1) = {b,c}.

In Figure 2.5, each state is shown with its max-Refusal set attached.

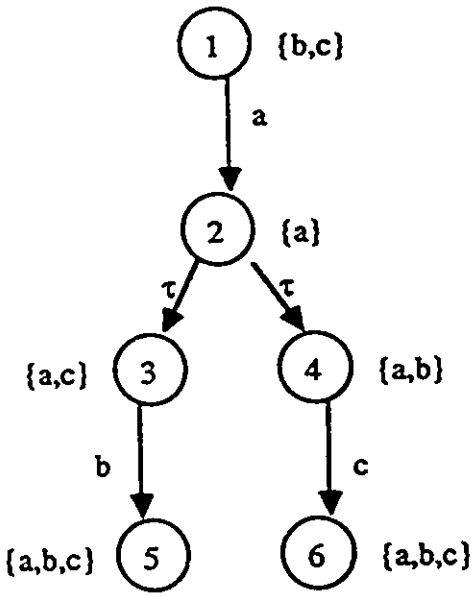


Figure 2.5 An example showing the refusal sets of an LTS .

Definition 2.18 ($\cong^f$, failure equivalence of two states)

Let $p \in P$ and $q \in Q$. p and q are said to be *failure equivalent*, denoted as $p \cong^f q$, iff, $\forall s \in \Sigma_{obs}^*$ and $\forall Z \subset \Sigma_{obs}$, the following conditions are satisfied:

1) If $\exists p' \in P$ such that $p == s ==> p'$ and $Z \in$ Refusal(p'), then $\exists q' \in Q$ such that $q == s ==> q'$ and $Z \in$ Refusal(q').

2) If $\exists q'' \in Q$ such that $q == s ==> q''$ and $Z \in$ Refusal(q''), then $\exists p'' \in Q$ such that $p == s ==> p''$ and $Z \in$ Refusal(p'').

Intuitively, failure equivalence of two states requires them to be able to accept the same sequences of observable actions and to reach states which have the same "refusal sets" of observable actions.

Definition 2.19 ($\cong^f$, failure equivalence of two LTSs)

LTS1 and LTS2 are *failure equivalent*, denoted as LTS1 $\cong^f$ LTS2, iff $p_0 \cong^f q_0$.

Example 2.6 (Figure 2.6)

LTS1 $\cong^f$ LTS2 because, for all observable action sequences a, ab and ac, p_0 and q_0 have the same refusal sets at the reached states. For s = a, $p_0 == a ==> \{p_1, p_2\}$ and $q_0 == a ==> \{q_1, q_2, q_3, q_4\}$. States p_1 and q_2 have the same refusal sets {{a},{c},{a,c}}, p_2 and q_4 have the same

refusal sets {{a},{b},{a,b}}. Both q_1 and q_3 have refusal sets {a} and it is also refused by p_1 or p_2. Additionally, for s = ab, p_0 == ab ==> {p_3} and q_0 == ab ==> {q_5, q_6}; for s = ac, p_0 == ac ==> {p_4} and q_0 == ac ==> {q_7, q_8}. State p_3, p_4, q_5, q_6, q_7, q_8 have the same refusal sets {a,b,c}.

Since a, ab and ac are the only observable sequences in both LTS1 and LTS2, $p_0 \equiv^f q_0$.

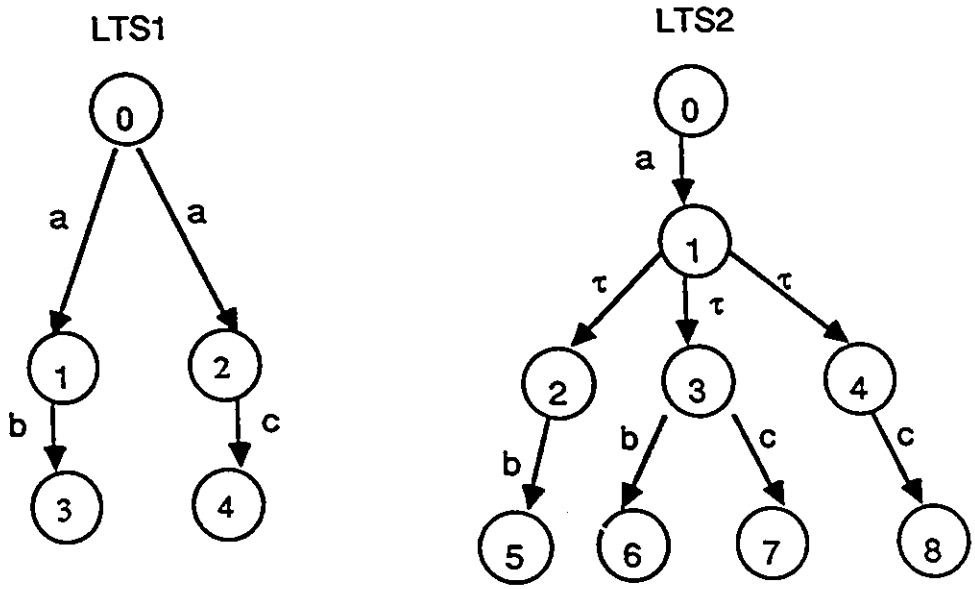


Figure 2.6 Two failure equivalent LTSs.

2.8 TESTING EQUIVALENCE

In order for p_0 and q_0 to be strong, weak or EB equivalent, it requires that, for every state in Reach({p_0},a,~~>), there exists a state in Reach({q_0},a,~~>) such that these two states are equivalent under the same equivalence relation. Note that these reached states are compared individually. By comparing the two sets Reach({p_0},a,~~>) and Reach({q_0},a,~~>) collectively, we can describe another useful equivalence relation, namely, testing equivalence. For this purpose, we will concentrate only on strongly convergent LTSs, that is, systems which cannot perform any infinite sequence of internal actions. For non-strongly convergent systems, testing equivalence [DEN84] is stronger than failure equivalence [HOA81] in its ability to distinguish two systems. They are equivalent in the case of strongly convergent systems. Testing (Failure) equivalence presents a rather general approach for comparing the behavior of distributed processes by tabulating the possible effects of the interactions between observers and processes. Processes are distinguished w.r.t. their ability in passing tests and inability not to fail tests. Testing (Failure)

equivalence differs from the equivalence relations studied previously in that it is not sensitive to where an indeterministic choice is made.

Definition 2.20 (weakly observable actions)

For $p \in K$, the set of *weakly observable actions* at p is defined as follows :

$O_w(p) = \{ a \mid a \in \Sigma_{obs}, \exists p' \in K$ such that $p == a ==> p' \}$.

Definition 2.21 (Outs(P), MinOuts(P))

For $P \subset K$, $Outs(P) = \{ O_w(p) \mid p \in P \}$. MinOuts(P) is the largest subfamily of Outs(P) such that, if $L_1, L_2 \in$ Outs(P) and L_1 is a proper subset of L_2, then $L_2 \notin$ MinOuts(P).

Definition 2.22 ($\cong^t$, testing equivalence of two states)

Let $p \in P$ and $q \in Q$. p and q are said to be *testing equivalent*, denoted as $p \cong^t q$, iff,

$\forall s \in \Sigma_{obs}^*$, MinOuts(Reach({p},s,==>)) = MinOuts(Reach({q},s,==>)) .

Intuitively, the testing equivalence of two states requires that the same set of observable actions sequences can be executed at both states and that their sets of reached states must have the same group of "acceptable sets" of observable actions.

Definition 2.23 ($\cong^t$, testing equivalence of two LTSs)

LTS1 and LTS2 are said to be *testing equivalent*, denoted as LTS1 $\cong^t$ LTS2, iff $p_0 \cong^t q_0$.

The following theorem describes the relationship between failure equivalence and testing equivalence in a strongly convergent LTS.

Theorem 2.2

For every $p \in K$, $s \in \Sigma_{obs}^*$ and $L \subset \Sigma_{obs}$, $L \in$ MinOuts(Reach({p},s,==>)) iff $\exists p' \in K$ such that $p == s ==> p'$ and $\Sigma_{obs} - L =$ max-Refusal(p').

Proof: It follows directly from the definitions of MinOuts(P) and max-Refusal(p) (Definitions 2.17 and 2.21). []

Theorem 2.2 states that if L is an *acceptance set* of actions which represents the weakly observable actions after LTS executes the action sequence s at p, then $\Sigma_{obs} - L$ is a *refusal set* of actions which cannot be executed.

Example 2.7 (Figure 2.7)

In the following, it will be shown that MinOuts(Reach({p_0},s,==>)) = MinOuts(Reach({q_0},s,==>)) for each of the observable action sequences s = a, s = ab, s = abc and s = abd. Hence, $p_0 \equiv^t q_0$.

For s = a, we have Reach({p_0},a,==>) = { p_1 }, Reach({q_0},a,==>) = { q_1, q_2, q_3 }, and MinOuts(Reach({p_0},a,==>)) = MinOuts(Reach({q_0},a,==>)) = {{b}}.

For s = ab, we have Reach({p_0},ab,==>) = { p_2, p_3 }, Reach({q_0},ab,==>) = { q_4, q_5 }, and MinOuts(Reach({p_0},ab,==>)) = MinOuts(Reach({q_0},ab,==>)) = { {c},{d} }.

For s = abc or s = abd, we have Reach({p_0},s,==>) = { p_0 }; Reach({q_0},s,==>) = { q_0 } and MinOuts(Reach({p_0},s,==>)) = MinOuts(Reach({q_0},s,==>)) = {{a}}.

However, neither $p_0 \equiv^w q_0$ nor $p_0 \equiv^{eb} q_0$. In fact, we have Reach({p_0},a,==>) = {p_1} and Reach({q_0},a,==>) = {q_1 , q_2, q_3}, but neither $p_1 \equiv^w q_2$ nor $p_1 \equiv^w q_3$. Also, we have Reach({p_0},a,==>) = {p_1} and Reach({q_0},a,==>) = {q_2, q_3}, but neither $p_1 \equiv^{eb} q_2$ nor $p_1 \equiv^{eb} q_3$.

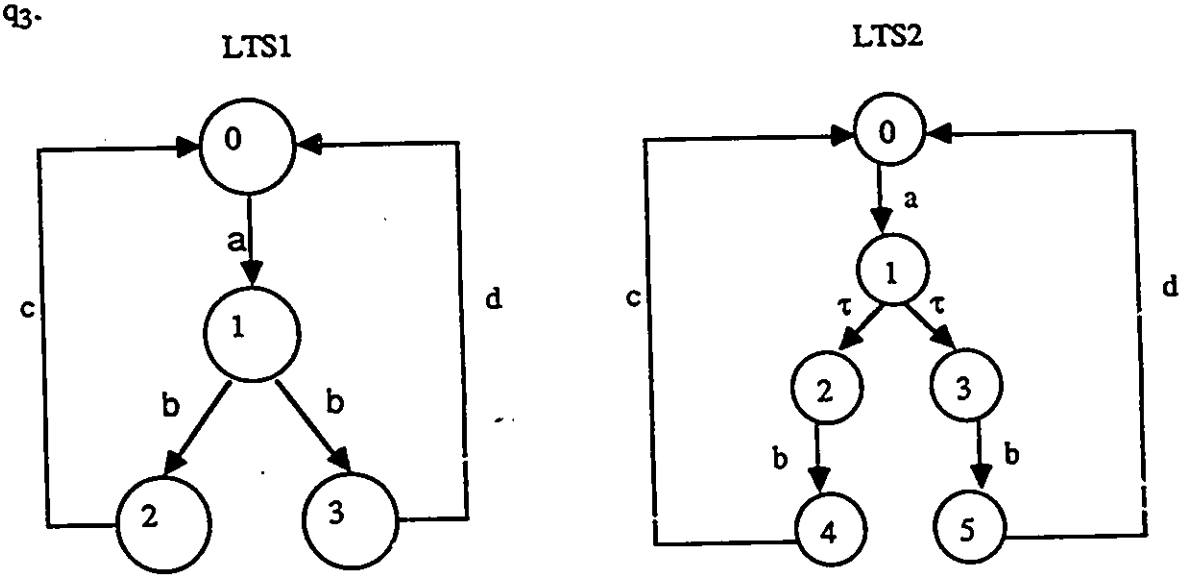


Figure 2.7 Two testing equivalent LTSs which are neither weakly equivalent nor EB equivalent.

In the literature, testing equivalence is also defined in terms of sets such as Must and MinMust. Their relationships are explained below.

Definition 2.24 (Must)

For P C K and L C Σ_{obs}, P *Must* L iff, $\forall p \in P$, $L \cap O_W(p) \neq \emptyset$.

Definition 2.25 (Must(P), MinMust(P))

For P C K, Must(P) = { L | L C Σ_{obs}, P Must L }

MinMust(P) is the largest subfamily of Must(P) such that, if L_1, $L_2 \in$ Must(P) and L_1 is a proper subset of L_2, then $L_2 \notin$ MinMust(P).

Theorem 2.3 Suppose P and Q C K. The following three conditions imply one another:

1) Must(P) = Must(Q)

2) MinMust(P) = MinMust(Q)

3) MinOuts(P) = MinOuts(Q)

Proof : See [BOL89]. []

2.9 TRACE EQUIVALENCE

In this section, we introduce a weaker equivalent relation, namely, trace equivalence. Two LTSs are considered to be trace equivalent if they can execute the same set of sequences of observable actions starting from their initial states. Trace equivalence differs from the equivalent relations investigated previously in the fact that, after the two LTSs have executed a sequence of observable actions, it does not require the two reachable states to satisfy any additional conditions.

Definition 2.26 (Traces)

Traces(p) = { s | s $\in$ Σ_{obs}^*, $\exists$ p' $\in$ K such that p $\Longrightarrow$ s $\Longrightarrow$ p' }.

Definition 2.27 ($\equiv^{tr}$, trace equivalence of two LTSs)

LTS1 and LTS2 are said to be *trace equivalent*, denoted as LTS1 $\equiv^{tr}$ LTS2, iff Traces(p_0) = Traces(q_0).

Theorem 2.4

If p $\equiv^{t}$ q then Traces(p) = Traces(q).[]

Proof: Omitted because of its triviality. []

Example 2.8 (Figure 2.8) We have LTS1 $\equiv^{tr}$ LTS2 because Traces(p_0) = Traces(q_0) = {a, ab, ac}. It can be proved that LTS1 and LTS2 are not strong, weak, exhibited behavior and testing equivalent.

Example 2.8 will be repeated as Example 4.1 under a unified definition of equivalence.

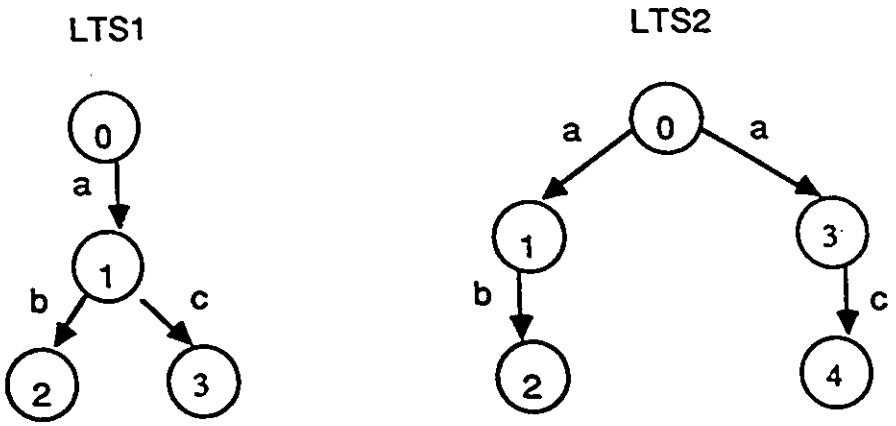


Figure 2.8 Two trace equivalent LTSs which are not strong, weak , EB and testing equivalent.

Chapter 3

REDUCTION BETWEEN EQUIVALENCE RELATIONS

3.1 INTRODUCTION

Chapter 2 provides a review on several equivalence relations for LTSs. In this chapter, we study the interrelationships between these equivalence relations under a transformation of the LTS. In fact, when one LTS is transformed to another LTS', a type of equivalence relation in LTS may be reduced to a different type in LTS'. Also, the transformation should be "property preserving", i.e., some properties of the LTS should be semantically maintained in LTS' under certain criteria.

In the following sections, we first consider those transformations which lead to three types of reduction: (1) weak equivalence reduced to strong equivalence, (2) EB equivalence reduced to weak equivalence, and (3) testing equivalence reduced to extended trace equivalence. Then, we discuss the property-preservation issues under these transformations.

This chapter includes our following original contributions:

1) We propose a rule (Transformation Rule R2) for transforming one LTS to another so that verification of EB equivalence can be reduced to verification of weak equivalence (Theorem 3.3, Collary 3.1, Lemma 3.1, Theorem 3.4).
2) We show the properties which are preserved under transformations (Theorems 3.2, Lemma 3.2, Theorem 3.5).
3) We provide a general property satisfied by the transformations (Theorems 3.7, 3.8, 3.9 and 3.10).

3.2 REDUCTION FROM WEAK EQUIVALENCE TO STRONG EQUIVALENCE [KAN83]

It is possible to verify the weak equivalence of two states of an LTS by showing that they are strongly equivalent in another LTS', which is derived from LTS by using the following transformation.

Transformation Rule R1: (changing weak-reachability to strong-reachability) [KAN83]

Transform LTS = $< K, \Sigma, \Delta, p_0 >$ to LTS' = $< K, \Sigma_{obs} \cup \{ \varepsilon \}, \Delta', p_0 >$, where $\Delta' = \Delta \cup \{ (p,a,q) \mid p, q \in K, a \in \Sigma_{obs} \cup \{ \varepsilon \}, p = a \Rightarrow q \text{ in LTS} \}$.

The only difference between LTS' and LTS is in their transition functions Δ' and Δ. In simple words, Δ' is obtained by adding the transition (p,a,q) to Δ whenever q is weakly-reachable from p by a in LTS. This is equivalent to changing every $p == a ==> q$ to $p -- a --> q$.

Example 3.1 (Figure 3.1)

According to Rule R1, since $p_0 == a ==> p_2$ is in LTS, transition (p_0,a,p_2) is added. Similarly, since $p_1 == b ==> p_3$ is in LTS, transition (p_1,b,p_3) is added.

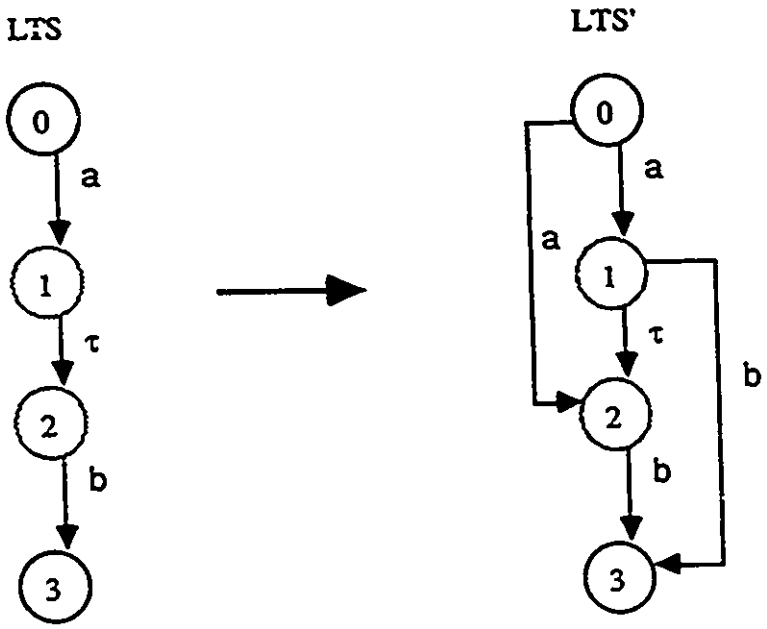


Figure 3.1 Changing LTS to LTS' by Transformation Rule R1.

Theorem 3.1 [KAN83]

Suppose LTS' is transformed from LTS by applying Rule R1. $p \cong^W q$ in LTS iff $p \cong^S q$ in LTS'.

Proof: This follows from the fact that LTS and LTS' have the same state space and action set and that, for any states p and q, $p == a ==> q$ in LTS iff $p -- a --> q$ in LTS'. □

Based on Transformation Rule R1 and Theorem 3.1, verification of weak equivalence in LTS can be done in LTS' as follows: 1) Transform LTS to LTS' by Rule R1. 2) Verify strong equivalence in LTS'.

The following theorem states some equivalence relations between LTS and LTS'.

Theorem 3.2

Suppose LTS' is transformed from LTS by applying Rule R1. LTS and LTS' are weakly, testing and trace equivalent.

Proof: Note that the definitions of weak, testing and trace equivalences are all based on weak reachability ==>. Let X = Reach({p},a, ==>) in LTS and X' = Reach({p},a,==>) in LTS'. By considering the facts that p == a ==> q in LTS iff p -- a --> q in LTS', that p -- a --> q is the same as p == a ==> q in LTS' and that LTS and LTS' have the same state space, we can conclude that, for every state $p \in K$ and every action $a \in \Sigma_{obs} \cup \{ \varepsilon \}$, $X = X'$. Furthermore, for every state $p \in K$, p (in X) $\equiv^w$ p in (X') and MinOuts(X) = MinOuts(X'). In particular, these cases hold for initial state p_0. It follows from Definitions 2.13, 2.15 and 2.22 that LTS $\equiv^x$ LTS', where $\equiv^x$ stands for weak, testing or trace equivalence. []

3.3 REDUCTION FROM EB EQUIVALENCE TO WEAK EQUIVALENCE

It is possible to verify the EB equivalence of two states in one LTS by showing that they are weakly equivalent in another LTS', which is derived from LTS by the transformation Rule R2 to be described below. But, let us define the following term first.

Definition 3.1 (isolated state)

A state $p \in K$ is said to be *isolated* if there do not exist any state $q \in K$ and action $a \in \Sigma$ such that (p, a, q) or (q, a, p) $\in \Delta$.

The following rule cannot be used for non-strongly convergent LTSs which have ε-loops (i.e., loops containing only transition τ) originating and ending at the same unobservable state. This kind of unobservable states cannot be eliminated from LTSs by the rule. It has been proved [POM86] that, in the non-strongly convergent case, there exists two EB equivalent LTSs which are not testing equivalent. For our application, under the assumption that it is possible to observe the beginning and ending of a normal execution (see the assumption of Definition 2.3 and Definition 2.23), we consider strongly convergent LTSs only.

Transformation Rule R2 (Eliminating unobservable states from an LTS)

Transform LTS = < K, Σ, Δ, p_0 > to LTS' = < K_O, Σ, Δ', p_0 >, where K_O is the subset of observable states of K and Δ' is obtained from Δ by applying the following steps repeatedly until it is no longer possible.

Step 1. In Δ, for every (p,a,q), where $a \in \Sigma$ and q is an unobservable state, execute the following operations:

a) If (p,a,q) is the only transition ending at q, then delete (p,a,q) and all (q,τ, r), where

$r \in$ Reach(q,τ,-->); otherwise, delete just (p,a,q).

b) For every $r \in$ Reach(q,τ,-->), add (p,a,r) into Δ.

Step 2. Delete q from K if q is isolated.

The main difference between LTS and LTS' is in their transition functions Δ and Δ'. Δ' is obtained from Δ by deleting all unobservable states and their incident transitions, and by adding the transition (p,a,r) whenever r is EB-reachable from p by a and there exists at least one unobservable state q such that $p = a \Longrightarrow q$ and $q -- \tau --> r$ in LTS.

Essentially, Rule R2 transforms indeterminism caused by unobservable states to indeterminism caused by the multiple occurrences of observable actions. It reflects the characteristics that EB equivalence ignores unobservable states.

Example 3.2 (Figure 3.2)

Assume that a1, a2, b1, and b2 are observable actions. State p_3 is the only unobservable state in LTS. First, consider $(p_1,a1,p_3)$ in Δ. Since $(p_2,a2,p_3)$ is also in Δ and Reach$(p_3,\tau,-->) = \{p_4, p_5\}$, $(p_1,a1,p_3)$ is deleted (Step 1.a of Rule R2) while $(p_1,a1,p_4)$ and $(p_1,a1,p_5)$ are added (Step 1.b of Rule R2). Now, since $(p_2,a2,p_3)$ has become the only transition leading to p_3 in Δ, $(p_2,a2,p_3)$, (p_3,τ,p_4) and (p_3,τ,p_5) are all deleted (Step 1.a of Rule R2) while $(p_2,a2,p_4)$ and $(p_2,a2,p_5)$ are added (Step 1.b of Rule R2). Lastly, p_3 is deleted because it becomes isolated by this time.

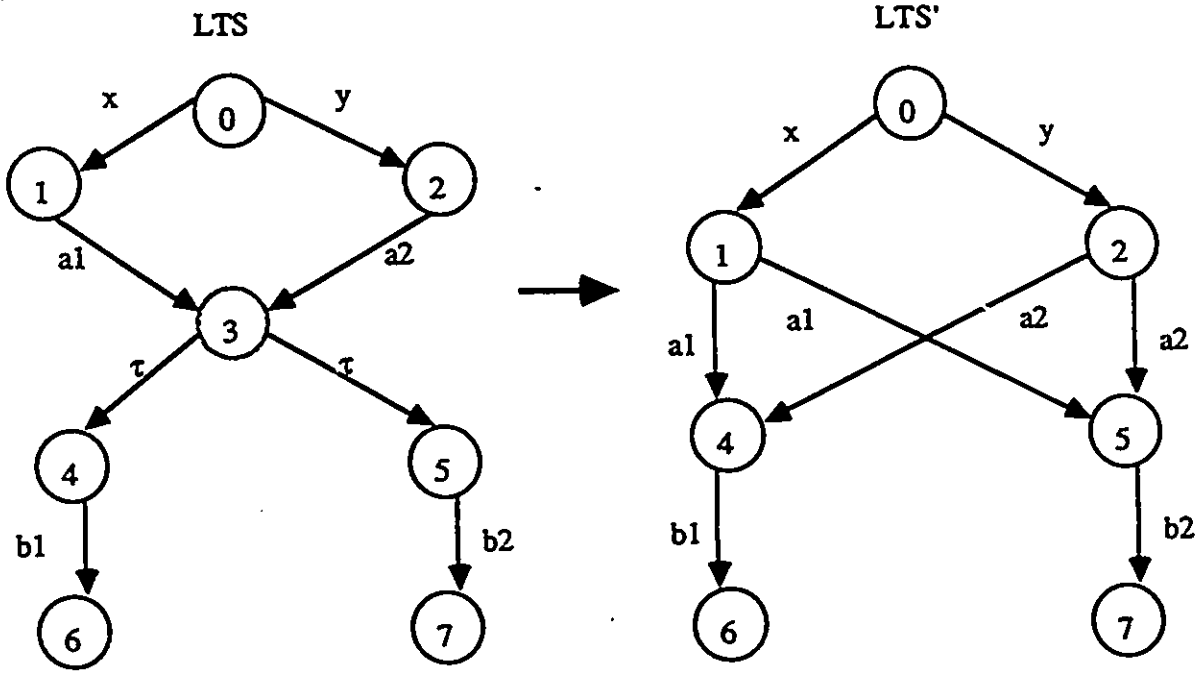


Figure 3.2 An example of changing LTS to LTS' by Transformation Rule R2.

The following theorem states that Transformation Rule R2 preserves weak reachability "==>" between two observable states. Note that this is not true for unobservable states, because LTS' does not have unobservable states.

Theorem 3.3 Suppose LTS' is transformed from LTS by applying Rule R2. For every $p, q \in K_O$ and every $a \in \Sigma_{obs} \cup \{ \varepsilon \}$, $p == a ==> q$ in LTS iff $p == a ==> q$ in LTS'.

Proof: For every $(p,a,q) \in \Delta'$, we call it "new" if it is added into Δ during the transformation; otherwise we call it "old".

(<==) $p == a ==> q$ in LTS' means that there exists a sequence of states $q_1, q_2, ..., q_m \in K_O$ such that $p = q_1 -- \tau --> q_2$, $q_2 -- \tau --> q_3$, ..., $q_j -- a --> q_{j+1}$, ..., $q_{m-1} -- \tau --> q_m = q$. Consider each $q_i -- \tau$ (or a) $--> q_{i+1}$, where $1 <= i <= m-1$. If it is "old", it also exists in LTS. If it is "new", k ($>= 0$) unobservable states $q_{i,1}, q_{i,2}, ..., q_{i,k}$ in LTS may have been deleted between q_i and q_{i+1}. So before the transformation, we must have $q_i -- \tau$ (or a) $--> q_{i,1}$, $q_{i,1} -- \tau --> q_{i,2}$, ..., $q_{i,k} -- \tau -- q_{i+1}$ in LTS. That is, $q_i == \tau$ (or a) $==> q_{i+1}$ in LTS. By combining all these transitions, we have $p == a ==> q$ in LTS.

(==>) we are going to prove that the fact $p == a ==> q$ in LTS implies that $p == a ==> q$ in LTS' by induction on m, where m is the number of unobservable states occurring on the path $p == a ==> q$ in LTS.

i. $m = 0$, which means there exists a sequence of states $q_1, q_2, ..., q_m$, such that $p = q_1 -- \tau --> q_2$, $q_2 -- \tau --> q_3$, ..., $q_j -- a --> q_{j+1}$, ..., $q_{m-1} -- \tau --> q_m = q$ in LTS, where every $q_i -- \tau$ (or a) $--> q_{i+1}$ is "old" and every q_i ($i = 2, 3, ..., m-1$) $\in K_O$. So after transformation, $p == a ==> q$ exists in LTS'.

ii. Assume for m-1, $p == a ==> q$ in LTS implies $p == a ==> q$ in LTS'.

iii. Suppose that there are m unobservable states in the trace $p == a ==> q$ in LTS. We consider the following two cases :

a) There is no observable state on the path from p to q (see Figure 3.3). Based on Transformation Rule R2, we will add $p -- a --> q$ in LTS' and all the m unobservable states will be deleted. so $p == a ==> q$ exists in LTS'.

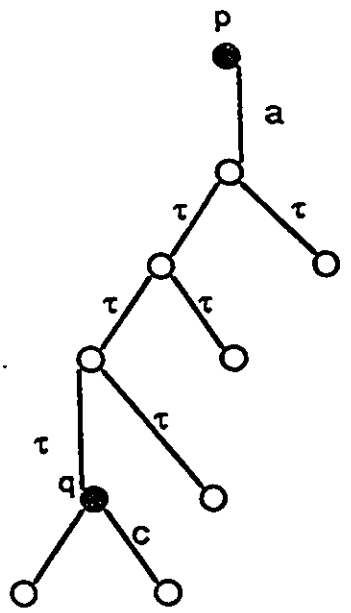


Figure 3.3 All states on the path between p and q are unobservable .

b) Suppose q^* is the last observable state in the trace (see Figure 3.4). Then either ($p == a ==> q^*$ and $q^* == \varepsilon ==> q$) or ($p == \varepsilon ==> q^*$ and $q^* == a ==> q$) holds in LTS. We now prove that if $p == a ==> q^*$ and $q^* == \varepsilon ==> q$ in LTS then $p == a ==> q$ in LTS'. Similar proof can be used in the case where $p == \varepsilon ==> q^*$ and $q^* == a ==> q$ in LTS.

If all m unobservable states are between q^* and q, then according to Transformation Rule R2, $q^* -- \tau -> q$ will be added in LTS'. Since $p == a ==> q^*$ in LTS, so $p == a ==> q^*$ is true in LTS' (where m = 0). By combining these two traces, we have $p == a ==> q$ in LTS'.

If there is at least one unobservable state between p and q^*, by induction, we have both $p == a ==> q^*$ and $q^* == \varepsilon ==> q$ in LTS'. This implies $p == a ==> q$ in LTS'. []

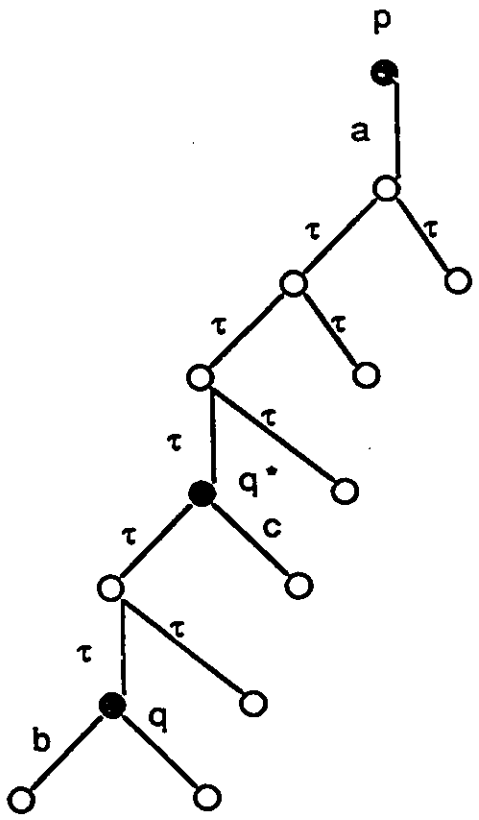


Figure 3.4 There is at least one observable state q^* on the path between p and q.

The above result can be extended to sequences of observable actions, as stated in the following corollary.

Corollary 3.1 Suppose LTS' is transformed from LTS by applying Rule R2. For every $p, q \in K_O$ and every $s \in \Sigma_{obs}^*$, $p = s \Longrightarrow q$ in LTS iff $p = s \Longrightarrow q$ in LTS'.
Proof: By induction on the length of s. []

Lemma 3.1 Suppose LTS' is transformed from LTS by applying Rule R2. For every $p, q \in K_O$ and every $k >= 0$, $p \not=_k^{cb} q$ in LTS iff $p \not=_k^{w} q$ in LTS'.
Proof : By induction on k, Theorem 3.3 and the fact that , $\forall$ p, q $\in K_O$ and $\forall a \in \Sigma_{obs} \cup \{ \varepsilon \}$, $p \equiv a \Longrightarrow q$ in LTS iff $p = a \Longrightarrow q$ in LTS'. []

Theorem 3.4 Suppose LTS' is transformed from LTS by applying Rule R2. For every $p, q \in K_O$, $p \equiv^{cb} q$ in LTS iff $p \equiv^{w} q$ in LTS'.
Proof: This follows from Lemma 3.1 and the definition of $\equiv^{cb}$. []

Based on Transformation Rule R2 and Theorem 3.4, verification of EB equivalence in LTS can be carried out in LTS' as follows: 1) Transform LTS to LTS' by Rule R2. 2) Verify weak equivalence in LTS'.

Next, we shall show some properties between LTS and LTS'.

Lemma 3.2 Suppose LTS' is transformed from LTS by applying Rule R2. For every $s \in \Sigma_{obs}^*$, MinOuts(Reach($\{p_0\}$,s,==>)) in LTS = MinOuts(Reach($\{p_0\}$,s,==>)) in LTS'.
Proof: By Definition 2.3, the initial state p_0 is observable and hence will not be deleted by Rule R2. Let R = Reach($\{p_0\}$,s,==>)) in LTS and R' = Reach($\{p_0\}$,s,==>)) LTS'. According to Transformation Rule R2, every $q \in R'$ is observable. By Corollary 3.1, $q \in R$ and hence $R' \subset R$. Suppose, in LTS, $R = R' \cup X$, where X is the set of unobservable states. By Definition 2.21, Outs(R) = Outs(R') $\cup$ Outs(X) and Outs(X) = $\{O_W(p^*) \mid p^* \in X\}$. For every unobservable state p^* in X, since p^* is not the terminating state, there must exist a subset X' of R', such that for every state $p \in X'$, $p_0 == s ==> p^*$, $p^* == \varepsilon ==> p$ and $O_W(p) \subset O_W(p^*)$. (See Figure 3.5). By the definition of MinOuts(R), we know that $O_W(p^*)$ will be eliminated from Outs(R). Hence in LTS, MinOuts(R) = MinOuts(R'). []

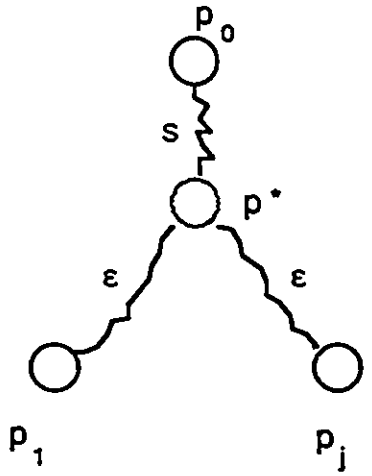


Figure 3.5 Construction of MinOuts(Reach($\{p_0\}$,s,==>) in LTS.

Theorem 3.5 Suppose LTS' is transformed from LTS by using Rule R2. LTS and LTS' are EB, testing and trace equivalent.

Proof: (For EB equivalence) By Theorem 3.3, for any observable states p and q, $p == a ==> q$ in LTS iff $p == a ==> q$ in LTS'. By the facts that $p == a ==> q$ iff $p \equiv a \equiv\!\!> q$ in LTS' and that LTS and LTS' have the same observable state space, it follows that LTS $\equiv^{eb}$ LTS'.

(For testing equivalence) By Lemma 3.2 and the definition of $\equiv^{t}$.

(For trace equivalence) For every sequence $s = a_1a_2...a_n \in Trace(p_0)$ in LTS, there exists a sequence of states $p_1, p_2, ..., p_n \in K_O$, such that $p_0 == a_1 ==> p_1$, $p_1 == a_2 ==> p_2$, ..., $p_{n-1} == a_n ==> p_n$. By Theorem 3.3, $p_i == a ==> p_{i+1}$ in LTS iff $p_i == a ==> p_{i+1}$ in LTS'. Thus, $s \in Trace(p_0)$ in LTS'. The reverse can be proved in a similar manner. []

3.4 REDUCTION FROM TESTING EQUIVALENCE TO EXTENDED TRACE EQUIVALENCE [BOL89]

It is possible to verify whether LTS1 and LTS2 are testing equivalent or not by showing whether LTS1' and LTS2' satisfy an extended version of trace equivalence, called *extended trace equivalence*, where LTS1' and LTS2' are derived from LTS1 and LTS2 by using the following transformation rule, respectively.

Transformation Rule R3: (changing an indeterministic LTS to a deterministic LTS)

Transform $LTS = < K, \Sigma, \Delta, p_0 >$ to $LTS' = < K', \Sigma_{obs}, \Delta', p'_0 >$ by the following steps.

Step 1. (changing weak reachability to strong reachability)

Obtain $LTS'' = < K, \Sigma, \Delta'', p_0 >$ from LTS by Transformation Rule R1 (Section 3.2).

Step 2. (changing an indeterministic LTS to a deterministic LTS)

Obtain $LTS' = < K', \Sigma_{obs}, \Delta', p'_0 >$ from LTS'' by applying a classical "subset construction" algorithm for transforming an indeterministic finite state machine to a deterministic one which has the same sequences of observable actions [AHO88], where K' is the set of states transformed from K. Each state in K' represents a subset of states of LTS'' and p'_0 represents the subset including p_0. Δ' is a transition relation in $K' \times \Sigma_{obs} \times K'$. $\Delta' = \{([p], a, [q]) \mid [p], [q] \in K'$, where [p] and [q] represent the subsets of K, respectively, $a \in \Sigma_{obs}$ and $[p] == a ==> [q]$ in LTS''$\}$ (See definition 2.10). After LTS' is obtained, the internal action τ is removed from action set Σ.

Step 3. (assigning labels to the states of LTS')

For every state p of LTS', if $\exists\, s \in \Sigma_{obs}^*$ such that $p'_0 -- s --> p$, then assign the label MO to p, where MO = MinOuts(Reach($\{p_0\}$,s,==>)) in LTS''.

Definition 3.2 ($\equiv^{et}$, extended trace equivalence)

Suppose $LTS1' = < P', \Sigma_{obs}, \Delta_{P'}, p'_0 >$ and $LTS2' = < Q', \Sigma_{obs}, \Delta_{Q'}, q'_0 >$ are transformed from LTS1 and LTS2 by using Rule R3, respectively. p'_0 and q'_0 are said to be *extended trace equivalent*, in notation, $p'_0 \equiv^{et} q'_0$, iff the following two conditions are satisfied :

i. $Trace(p'_0) = Trace(q'_0)$; and

ii. for every $s \in \Sigma_{obs}^*$, every $p' \in P'$ and every $q' \in Q'$, where $p'_0 -- s --> p'$ in $LTS1'$ and $q'_0 -- s --> q'$ in $LTS2'$, states p' and q' have the same MO-label.

Theorem 3.6 [BOL89]

Suppose $LTS1'$ and $LTS2'$ are transformed from LTS1 and LTS2 by using Rule R3, respectively. $LTS1 \equiv^{t} LTS2$ iff $LTS1' \equiv^{et} LTS2'$.

Proof : This follows directly from Definitions 2.23 and 3.2. []

Example 3.3 (Figure 3.6)

Figure 3.6 shows two LTSs LTS1 and LTS2 and their extended LTSs $LTS1'$ and $LTS2'$ obtained by applying Rule R3. In this example, since no τ actions are involved, the two intermediate LTSs $LTS1''$ and $LTS1''$ are the same as LTS1 and LTS2, respectively. The MO-labels are attached to the states of $LTS1'$ and $LTS2'$. Although $Trace(p'_0) = Trace(q'_0)$, it is not true that $LTS1' \equiv^{et} LTS2'$, because p'_1 and q'_1 have different labels. Hence, it is not true that $LTS1 \equiv^{t} LTS2$.

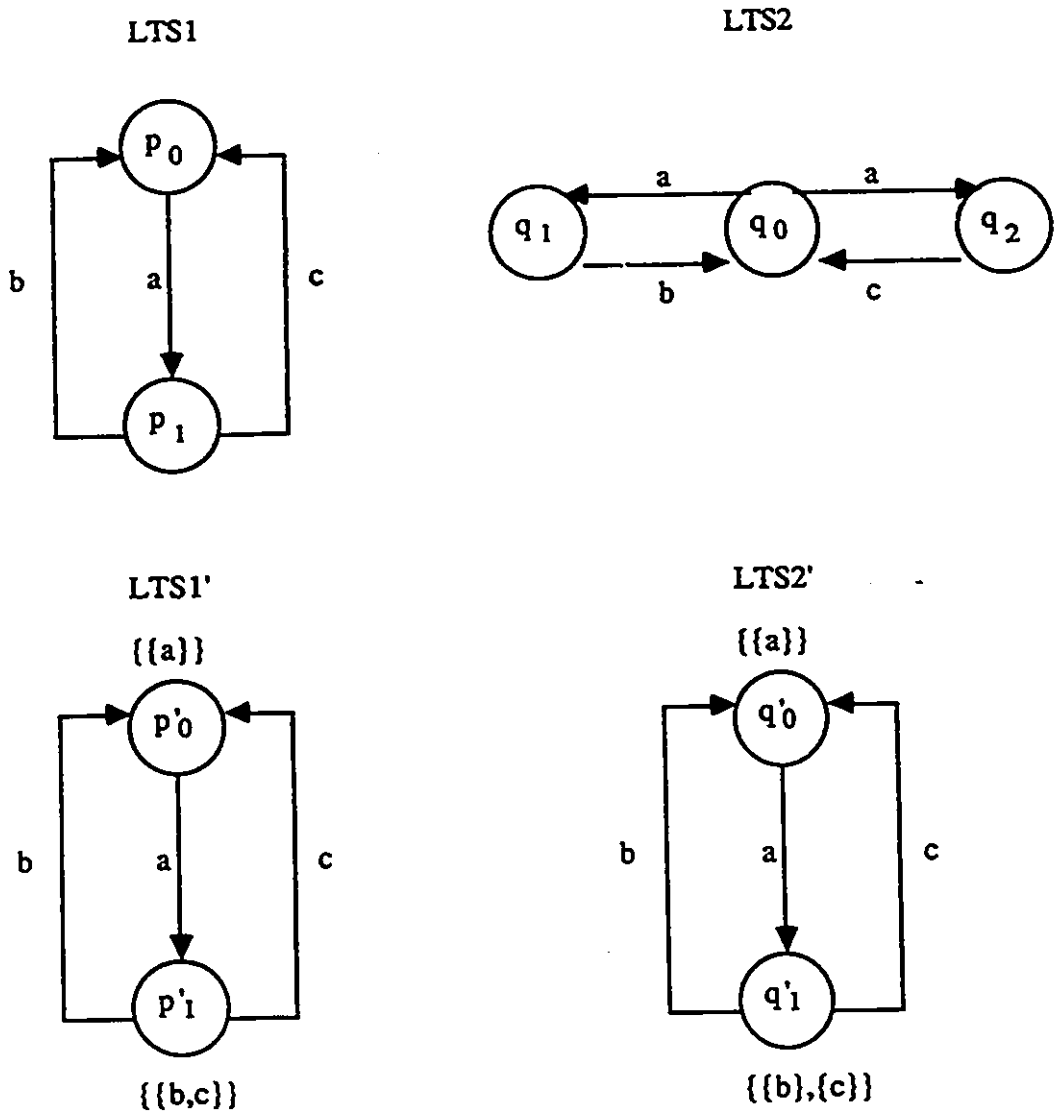


Figure 3.6 Two LTSs which are not testing equivalent.

3.5 PRESERVING PROPERTIES UNDER TRANSFORMATION

In the previous sections, we have seen that the verification of weak, EB and testing equivalences between two LTSs can be reduced to the verification of strong, weak and extended trace equivalences between two LTS's which are derived by using Transformation Rules R1, R2 and R3, respectively. Also, we have proved the existence of some properties between LTS and LTS'. In this section, we formalize and generalize these results as a problem called Correctness Preserving Transformation (CPT). Here, CPT means that after using a transformation rule, some important features of the original LTS have to be kept in the transformed system. i.e., they should be "semantically equivalent" under certain criteria. The CPT problem has been investigated for the

design and verification of communication protocols, for example, in the Petri-net model [CIN85] and LOTOS [LAN90]. In this section, we discuss the problem of equivalence-preserving transformations between LTSs.

Definition 3.3 (Satisfaction)

For a set S of LTSs, let R : S –> S be a transformation rule. R is said to *satisfy equivalence relation* $\equiv^x$ in S iff, $\forall$ LTS $\in$ S, LTS $\equiv^x$ R(LTS).

Theorem 3.7 Suppose $\equiv^x$ and $\equiv^y$ are two equivalence relations, where $\equiv^x \subset \equiv^y$, and R is a transformation rule. If R satisfies $\equiv^x$, then R satisfies $\equiv^y$.
Proof: Since $\equiv^x \subset \equiv^y$, it follows that LTS $\equiv^x$ R(LTS) implies LTS $\equiv^y$ R(LTS), for every LTS $\in$ S. []

Definition 3.4 (Preservation)

For a set S of LTSs, let R : S --> S be a transformation rule. R is said to *preserve equivalence relation* $\equiv^x$ in S iff, $\forall$ LTS1, LTS2 $\in$ S, R(LTS1) $\equiv^x$ R(LTS2) if LTS1 $\equiv^x$ LTS2.

Theorem 3.8 Suppose $\equiv^x$ is an equivalence relation and R is a transformation rule. If R satisfies $\equiv^x$, then R preserves $\equiv^x$.
Proof: Suppose LTS1 $\equiv^x$ LTS2. Since R satisfies $\equiv^x$, it follows that LTS1 $\equiv^x$ R(LTS1) and LTS2 $\equiv^x$ R(LTS2). Since $\equiv^x$ is an equivalence relation, it follows that R(LTS1) $\equiv^x$ R(LTS2). []

Theorem 3.9 Transformation Rule R1 satisfies and preserves weak, testing and trace equivalences in S.
Proof: It follows from Theorem 3.2 and Definition 3.3 that R1 satisfies weak, testing and trace equivalences. By Theorem 3.8, these equivalence relations are preserved by R1. []

Theorem 3.10 Transformation Rule R2 satisfies and preserves EB, testing and trace equivalences in S.
Proof: It follows from Theorem 3.5 and Definition 3.3 that R2 satisfies EB, testing and trace equivalences. By Theorem 3.8, these equivalence relations are preserved by R2. []

Chapter 4

A UNIFIED DEFINITION AND COMPARISON OF EQUIVALENCE RELATIONS

4.1 INTRODUCTION

This chapter presents the main theoretical results of this thesis. It includes two parts. In the first part (Section 4.2), by extracting three essential and common characteristics of the various equivalence relations defined in Chapter 2, we give a unified definition for them. Based on this definition, we illustrate by several examples the fact that the various equivalence relations differ from one another in their types of reachability and in the observable properties of the reached sets of states. In the second part (Section 4.4), we compare the strength of the various equivalence relations. Some of the equivalence relations have stronger conditions than the others. This means that their partitions of the universe of states into equivalent classes are more refined than those based on the others. These conditions can be strengthened or weakened to reflect various conditions for observation.

This chapter includes our following original contributions:

1) We propose a unified definition of equivalence relation (Definition 4.2).

2) We create many examples for illustrating the unified definition and for comparing the strength of the equivalence relations (Sections 4.3 and 4.4).

3) We provide a proof that EB equivalence is stronger than testing equivalence (Theorem 4.3).

4) We provide a disproof of the conclusion existing in literature that EB equivalence is stronger than weak equivalence [POM86] (Theorem 4.4).

4.2 A UNIFORM APPROACH FOR DEFINING EQUIVALENCE RELATIONS

In the literature, various equivalence relations have been defined by using different concepts and approaches. However, in Chapter 2, we have been able to single out the following three important properties for describing these relations :

1) The type of reachability of the action sequences.

2) The property of the reached set of states.

3) The relationship imposed between the reached sets of states.

In this section, based on the above properties, we give a unified definition for the various equivalence relations. This definition is based on our observation that, for the types of equivalence

relation studied in Chapter 2, verifying the equivalence of two labeled transition systems LTS1 = < P, Σ, Δ_P , p> and LTS2 = < Q, Σ, Δ_Q , q> can be done by the following uniform process.

A uniform process for verifying equivalence relations

First, depending on the equivalence relation $\equiv^x$ to be verified, select a type of reachability $\sim\sim>$ (out of $\rightarrow$, $\Longrightarrow$ and $\Longrightarrow$), a domain D of action sequences (from a subset of either Σ^+ or Σ_{obs}^*) and a criterion C_i from the set described below. Then, for each $s \in D$, carry out Step 1 and Step 2 below. $p \equiv^x q$ iff the result of Step 2 is affirmative.

Step 1: Find Reach({p},s,$\sim\sim>$) and Reach({q},s,$\sim\sim>$).

Step 2: Show whether Reach({p},s,$\sim\sim>$) and Reach({q},s,$\sim\sim>$) satisfy criterion C_i or not.

Set of possible criteria:

C1. For every $p' \in$ Reach({p},s,$\sim\sim>$), $\exists\, q' \in$ Reach({q},s,$\sim\sim>$) such that $p' \equiv^x q'$. Also, for every $q'' \in$ Reach({q},s,$\sim\sim>$), $\exists\, p'' \in$ Reach({p},s,$\sim\sim>$) such that $p'' \equiv^x q''$.

C2. MinOuts(Reach({p},s,$\sim\sim>$) = MinOuts(Reach({q},s,$\sim\sim>$).

C3. Both Reach({p},s,$\sim\sim>$) and Reach({q},s,$\sim\sim>$) are not empty.

The following terminology is needed in order to give a unified definition of equivalence relations.

Definition 4.1 (identity of sets of states)

For two subsets A and B of the state space of a LTS, different types of *identity* $=^x$ can be defined as follows:

1. A and B are said to be *strongly-identical*, denoted as $A =^s B$, iff, $\forall\, a \in A, \exists\, b \in B$ such that $a \equiv^s b$; and vice versa.
2. A and B are said to be *weakly-identical*, denoted as $A =^w B$, iff, $\forall\, a \in A, \exists\, b \in B$ such that $a \equiv^w b$; and vice versa.
3. A and B are said to be *EB-identical*, denoted as $A =^{eb} B$, iff, $\forall\, a \in A, \exists\, b \in B$ such that $a \equiv^{eb} b$; and vice versa.
4. A and B are said to be *testing-identical* , denoted as $A =^t B$, iff MinOuts(A) = MinOuts(B).
5. A and B are said to be *trace-identical*, denoted as $A =^{tr} B$, iff both A and B are not empty.

Based on the three types of reachability defined in Chapter 2 and the identities defined above, we can now give the following unified definition of the various equivalence relations.

Definition 4.2 (unified definition of equivalence relations)

Two states p and q of a LTS are said to be *x-equivalent* iff, $\forall s \in D$, Reach({p}, s, x-reachable) $=^x$ Reach({q}, s, x-reachable), where the domain D of the action sequences, the type of reachability 'x-reachable', the type of identity '$=^x$', and the way they are associated are defined according to a given set of rules.

In general, the unified definition is dependent on the given set of rules. Its essense is that its three properties characterize the different aspects of observing the behavior of an LTS.

i) The domain describes the scope of action sequences to be observed.

ii) The type of reachability describes the various ways of leading from one state to the others where further observation is to be made.

iii) The identity relation summarizes the the relationship between the two reached sets of states.

Note that, in the case of strong, weak and EB equivalences, Definition 4.2 is recursive. Strictly speaking, therefore, we have to use their k-equivalences.

As particular cases of this unified definition, the five equivalence relations investigated in Chapter 2 are summarized in Figure 4.1. It shows how the different types of reachability, action sequences and properties of the reached states are used to distinguish the various equivalence relations.

Equivalence relation	Domain of action sequences	Type of reacha-bility	Criterion for equivalence based on the identity relation between X and Y
Strong	$s \in \Sigma^+$	--> strong	$X =^s Y$
Weak	$s \in \Sigma_{obs}^*$	==> weak	$X =^w Y$
Exhibited behaviour	$s \in \Sigma_{obs}^*$	≡≡> eb	$X =^{eb} Y$
Testing	$s \in \Sigma_{obs}^*$	==> weak	$X =^t Y$, i.e., MinOuts(X) = MinOuts(Y)
Trace	$s \in \Sigma_{obs}^*$	==> weak	$X =^{tr} Y$, i.e., X and Y are not empty

Note : X = Reach({p},s,~~>); Y = Reach({q},s,~~>)

Figure 4.1 Three properties used for characterizing five equivalence relations under the unified definition.

4.3 EXAMPLES ON THE VERIFICATION OF EQUIVALENCE BASED ON THE UNIFIED DEFINITION

In this section, several examples are presented for illustrating how to verify an equivalence relation based on the uniform approach described in Section 4.2. These examples clearly show that the different equivalence relations depend on the three common characteristics. These examples are also used for comparing these equivalence relations in Section 4.4.

Example 4.1 (Figure 4.2)

In the following, let $s \in \Sigma_{obs}^*$, ~~> denote ==>, X = Reach({p_0},s,==>) and Y = Reach({q_0},s,==>).

1) LTS1 $\equiv^{tr}$ LTS2 holds for the following reason: For each of the sequences s = a, s = ab, and s = ac, we have $X \neq \emptyset$ iff $Y \neq \emptyset$. Hence, $X =^{tr} Y$.

2) LTS1 $\equiv^{t}$ LTS2 does not hold for the following reason: For s = a, we have $X = \{p_1\}$, MinOuts(X) = {{b,c}}, $Y = \{q_1,q_3\}$ and MinOuts(Y) = {{b},{c}}. Hence, $X =^{t} Y$ does not hold.

After action a, both b and c are acceptable at state p_1. However, only b (resp., c), but not both, is acceptable at state q_1 (resp., q_3). This means that the reachable sets $\{p_1\}$ and $\{q_1,q_3\}$ have different deadlock phenomena. To see this, consider a process S which accepts only action b after action a. LTS1 will never deadlock when communicating with S, but LTS2 will deadlock if, the system transfers to q_3 after action a.

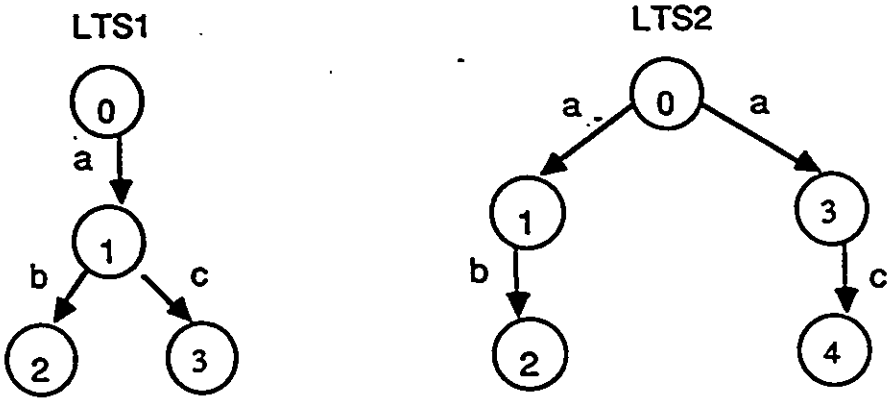


Figure 4.2 Two LTSs which are trace equivalent but not testing equivalent [PEH89].

Example 4.2 (Figure 4.3)

In the following, let $s \in \Sigma_{obs}^*$, X = Reach({p_0},s,~~>) and Y = Reach({q_0},s,~~>), where ~~> denotes $\Longrightarrow$ for testing equivalence and denotes $\Longrightarrow$ for EB equivalence.

1) LTS1 $\equiv^{t}$ LTS2 holds for the following reason: For s = a, we have $X = \{p_1,p_2,p_3\}$, MinOuts(X) = {{b},{c}}, $Y = \{q_1,q_2,q_3,q_4\}$ and MinOuts(Y) = {{b},{c}}. For s = ab and s = ac, we have MinOuts(X) = MinOuts(Y) = {$\emptyset$}. Hence, $X =^{t} Y$.

2) LTS1 $\equiv^{eb}$ LTS2 does not hold for the following reason: For s = a, we have $X = \{p_2,p_3\}$ and $Y = \{q_2,q_3,q_4\}$. But, $p \equiv^{eb} q_3$ does not hold for $q_3 \in Y$ and for every $p \in X$. Hence, $X =^{eb} Y$ does not hold.

The main point of this example is that, because of the internal operation τ following the interaction a, LTS1 and LTS2 reach two different sets of observable states $X = \{p_2,p_3\}$ and $Y = \{q_2,q_3,q_4\}$, respectively. State q_3 of Y has a different deadlock phenomenon from p_2 and p_3. Therefore, $p_0 \equiv eb\ q_0$ does not hold.

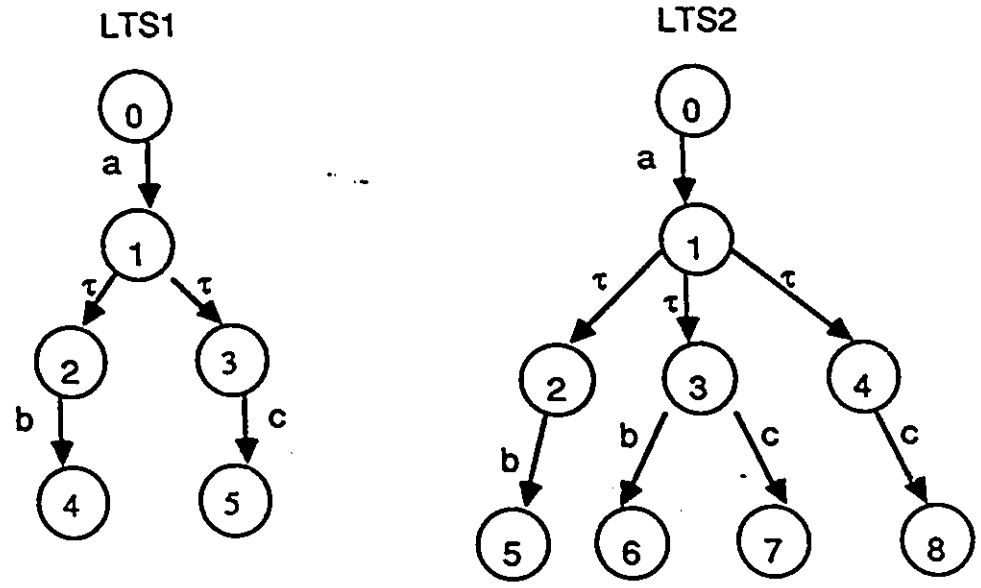


Figure 4.3 Two LTSs which are testing equivalent but not EB equivalent [POM86].

Example 4.3 (Figure 4.4)

In the following, let $s \in \Sigma_{obs}^*$, $X = Reach(\{p_0\},s,\sim\sim>)$ and $Y = Reach(\{q_0\},s,\sim\sim>)$, where $\sim\sim>$ denotes $\Longrightarrow$ for both testing equivalence and weak equivalence and denotes $\equiv\Longrightarrow$ for EB equivalence.

1) LTS1 $\equiv^t$ LTS2 holds for the following reason: For s = a, we have $X = \{p_1,p_2,p_3\}$, $MinOuts(X) = \{\{b\},\{c\}\}$, $Y = \{q_1,q_3\}$ and $MinOuts(Y) = \{\{b\},\{c\}\}$. For s = ab or s = ac, we have $MinOuts(X) = MinOuts(Y) = \{\emptyset\}$. Hence, $X =^t Y$.

2) LTS1 $\equiv^{eb}$ LTS2 holds for the following reason: For s = a, we have $X = \{p_2,p_3\}$, $Y = \{q_1,q_3\}$, $p_2 \equiv^{eb} q_1$ and $p_3 \equiv^{eb} q_3$. Similar properties hold for s = ab and s = ac. Hence, $X =^{eb} Y$.

3) LTS1 $\equiv^w$ LTS2 does not hold for the following reason: For s = a, we have $X = \{p_1,p_2,p_3\}$ and $Y = \{q_1,q_3\}$. But, $p_1 \equiv^w q$ does not hold for every $q \in Y$. Hence, $X =^w Y$ does not hold.

In LTS1, indeterminism is caused by the internal action τ. In LTS2, indeterminism is caused by the multiple occurrences of the external action a, and after a, either b or c is acceptable but never both. Thus, p_1 can be simulated by neither q_1 nor q_3.

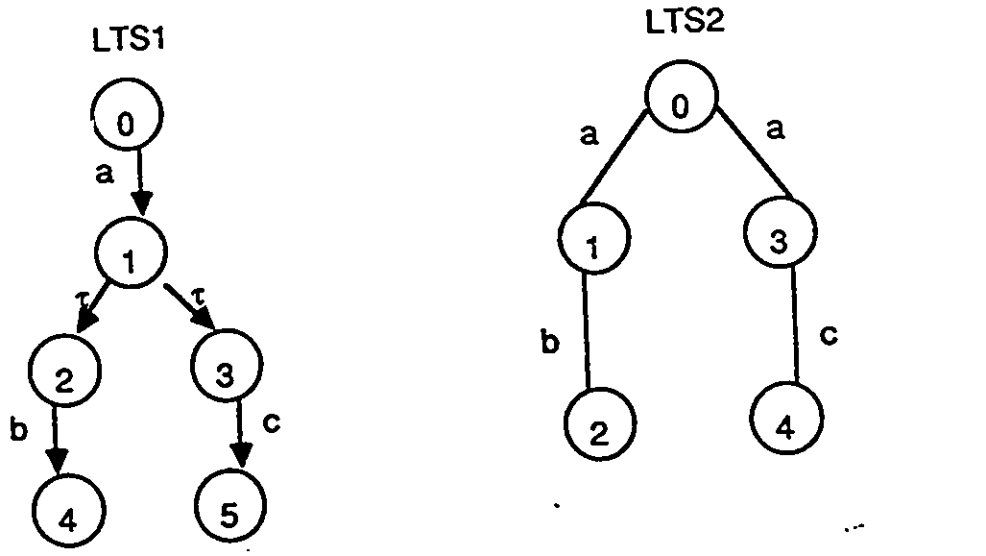


Figure 4.4 Two LTSs which are testing equivalent and EB equivalent but not weakly equivalent [POM86].

Example 4.4 (Figure 2.3)

In the following, let $X = Reach(\{p_0\},s,\sim\sim>)$ and $Y = Reach(\{q_0\},s,\sim\sim>)$, where $\sim\sim>$ denotes $\Longrightarrow$ for weak equivalence and denotes $\rightarrow$ for strong equivalence.

1) Let $s \in \Sigma_{obs}^*$, LTS1 $\equiv^w$ LTS2 holds for the following reason: For s = a, we have $X = \{p_1,p_3\}$, $Y = \{q_1,q_3,q_5\}$, $p_1 \equiv^w q_1$, $p_3 \equiv^w q_3$, and $p_3 \equiv^w q_5$. Similar properties hold for s = ab and s = ac. Hence, $X =^w Y$.

2) Let $s \in \Sigma^*$, LTS1 $\equiv^s$ LTS2 does not hold for the following reason: For s = a, we have $X = \{p_1\}$ and $Y = \{q_1,q_5\}$. But, $p_1 \equiv^s q_5$ does not hold. Hence, $X =^s Y$ does not hold.

Example 4.5 (Figure 4.5)

Let $s \in \Sigma_{obs}^*$, $X = Reach(\{p_0\},s,\sim\sim>)$ and $Y = Reach(\{q_0\},s,\sim\sim>)$, where $\sim\sim>$ denotes $\Longrightarrow$ for weak equivalence and denotes $\Longrightarrow$ for EB equivalence.

1) LTS1 $\equiv^w$ LTS2 holds for the following reason: For s = a, we have $X = \{p_1,p_2,p_3\}$, $Y = \{q_1,q_2,q_3\}$, $p_1 \equiv^w q_1$, $p_2 \equiv^w q_2$, and $p_3 \equiv^w q_3$. Similar properties hold for s = ab, s = ac and s = ad. Hence, $X =^w Y$.

2) LTS1 $\equiv^{eb}$ LTS2 does not hold for the following reason: For s = a, we have $X = \{p_1,p_2,p_3\}$ and $Y = \{q_2,q_3\}$. But, $p_1 \equiv^{eb} q$ does not hold for every $q \in Y$. Hence, $X =^{eb} Y$ does not hold.

Initially, action a is acceptable in both systems. However, after the interaction a, LTS1 reaches the observable state p_1 in which actions d, b and c are all acceptable. But, in the observable state q_2 or q_3 of LTS2, not all of d, b or c can be accepted. Hence, $p_0 \equiv^{eb} q_0$ does not hold.

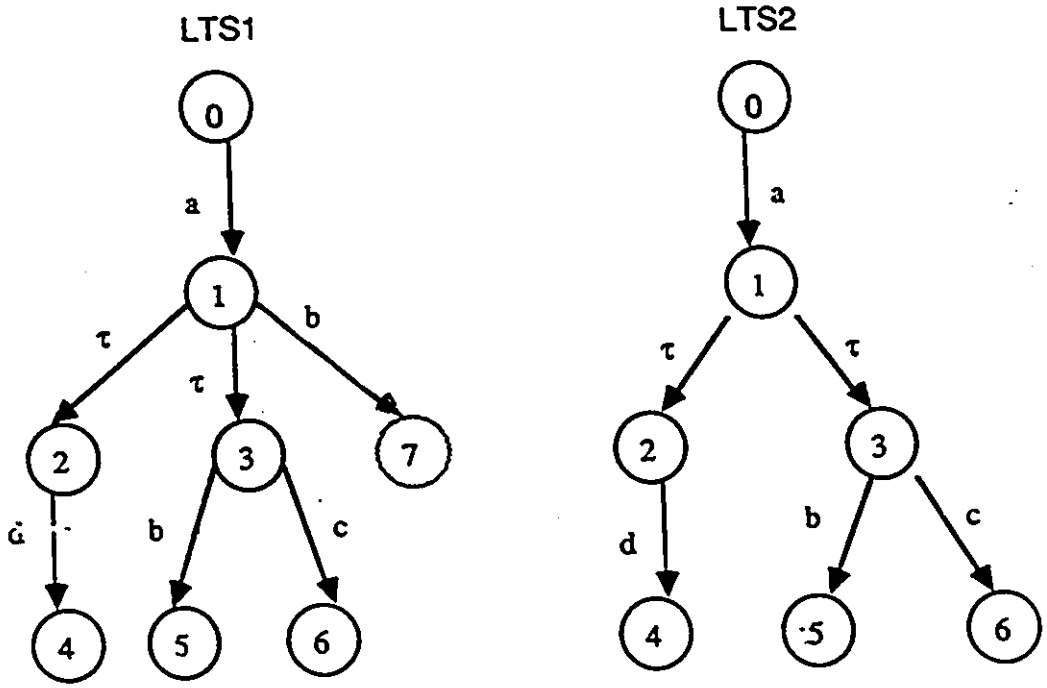


Figure 4.5 Two LTSs which are weakly equivalent but not EB equivalent.

4.4 COMPARISON ON THE STRENGTH OF EQUIVALENCE RELATIONS

In this section, we compare the strength of the various equivalence relations. By means of a counter example, we also disprove the conclusion existing in literature that EB equivalence is stronger than weak equivalence [POM86].

Definition 4.3 ($\geq$, stronger than; $>$, strictly stronger than; $=$, equal; $\times$, overlapping)

Let $\equiv^x$ and $\equiv^y$ be two equivalence relations.

1) $\equiv^x \geq \equiv^y$ iff, for any two LTSs LTS1 and LTS2, LTS1 $\equiv^x$ LTS2 implies LTS1 $\equiv^y$ LTS2.

2) $\equiv^x > \equiv^y$ iff $\equiv^x \geq \equiv^y$ and there exists at least one pair of LTSs LTS1 and LTS2 such that LTS1 $\equiv^y$ LTS2 but LTS1 $\equiv^x$ LTS2 does not hold.

3) $\equiv^x = \equiv^y$ iff $\equiv^x \geq \equiv^y$ and $\equiv^y \geq \equiv^x$.

4) $\equiv^x \times \equiv^y$ iff neither $\equiv^x \geq \equiv^y$ nor $\equiv^y \geq \equiv^x$ holds. $\equiv^x$ and $\equiv^y$ are said to be "overlapping".

Theorem 4.1 $\equiv^s > \equiv^w$.

Proof: It follows directly from Definitions 2.11 and 2.13 that strong reachability --> is a particular case of weak reachability ==>. Example 4.4 shows that two weakly equivalent LTSs may not be strongly equivalent. []

Theorem 4.2 $\equiv^w > \equiv^t$.

Proof: Suppose LTS1 $\equiv^w$ LTS2. It follows from the definition of weak equivalence that, for any sequence $s \in \Sigma_{obs}^*$, if $p_0 == s ==> p'$, $\exists\ q'$ such that $q_0 == s ==> q'$ and $p' \equiv^w q'$. $p' \equiv^w q'$ means that the set of observable actions acceptable at p' = the set of observable actions acceptable at q'. That is, MinOuts(Reach($\{p_0\}$,s,==>)) = MinOuts(Reach($\{q_0\}$,s,==>). Hence, LTS1 $\equiv^t$ LTS2. Example 4.3 shows that two testing equivalent LTSs may not be weakly equivalent. []

Theorem 4.3 $\equiv^{eb} > \equiv^t$.

Proof: Suppose LTS1 $\equiv^{eb}$ LTS2. Let LTS1' and LTS2' be transformed from LTS1 and LTS2 by using Rule R2, respectively. Since Rule R2 preserves EB equivalence (Theorem 3.11), LTS1' $\equiv^{eb}$ LTS2'. Also, LTS1' $\equiv^w$ LTS2' because no unobservable states exist in LTS1' and LTS2'. By Theorem 4.2, it follows LTS1' $\equiv^t$ LTS2'. By Theorem 3.6, LTS1' $\equiv^t$ LTS1 and LTS2' $\equiv^t$ LTS2, respectively. Hence, LTS1 $\equiv^t$ LTS2. Example 4.2 shows that two testing equivalent LTSs may not be EB equivalent. []

It has been claimed in the literature [POM86] that weak equivalence is stronger than EB equivalence. The following theorem nullifies such a claim.

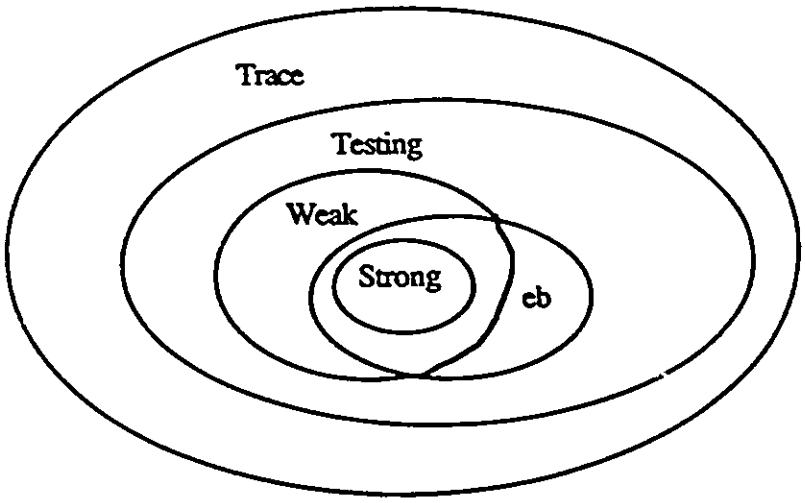
Theorem 4.4 : $\equiv^w \succ\prec \equiv^{eb}$.

Proof: Example 4.3 shows that two EB equivalent LTSs may not be weakly equivalent. Hence, EB equivalence is not stronger than weak equivalence. Example 4.5 shows that two weakly equivalent LTSs may not be EB equivalent. Hence, weak equivalence is not stronger than EB equivalence. []

Theorem 4.5 : $\equiv^t > \equiv^{tr}$.

Proof: From the definition of testing equivalence, $\equiv^t$ implies $\equiv^{tr}$. Example 4.1 shows that two trace equivalent LTSs may not be testing equivalent. []

Figure 4.6 shows the partial order of the strength of the five equivalence relations under consideration.



Notation : The stronger equivalence relations are represented as subsets of the weaker ones.

Figure 4.6 Comparison on the strength of five equivalence relations

Chapter 5

ALGORITHMS FOR VERIFYING EQUIVALENCE RELATIONS

5.1 INTRODUCTION

Chapters 2 and 3 provide the definitions of equivalence relations and reduction methods between them. In this chapter, we review some existing algorithms and also propose a new algorithm for verifying these relations. We first introduce the problem called "relational coarsest partition" [KAN83]. Its solution is useful for the verification of strong, weak, EB and testing equivalences. Then, we show: (1) strong equivalence can be verified by solving a set of relational coarsest partition problems; (2) weak equivalence can be verified by solving an instance of the strong equivalence problem; (3) EB equivalence can be verified by solving an instance of the weak equivalence problem; and (4) testing equivalence can be verified by solving an instance of the trace equivalence problem, which in turn can be formulated as a " relational coarsest partition" problem. The algorithms for verifying strong, weak and testing equivalences can be found in the literature [KAN83, BOL89].

This chapter includes our following original contributions:

1) We propose a new algorithm for verifying EB equivalence.
2) We provide a proof that the problem of EB equivalence verification can be solved in polynomial time.

An equivalence relation is defined in terms of the initial states of two different LTSs. However, the verification algorithms are applied to two different states of the same LTS. Hence, we have to define the union of two LTSs in order to apply a verification algorithm. Obviously, a necessary condition for equivalence is that the two LTSs have the same action set. In the following definition, without loss of generality, we may assume that the two LTSs have disjoint state spaces but the same action set.

Definition 5.1 (Union of LTSs)

Let $LTS1 = < P, \Sigma, \Delta_P, p_0 >$ and $LTS2 = < Q, \Sigma, \Delta_Q, q_0 >$ be two LTSs with disjoint state spaces but the same action set. The *union* of LTS1 and LTS2 is the labeled transition system $LTS = < K, \Sigma, \Delta, - >$, where $K = P \cup Q$, $\Delta = \Delta_P \cup \Delta_Q$ and the initial state of LTS is immaterial (as far as application of the verification algorithms is concerned).

5.2 THE RELATIONAL COARSEST PARTITION PROBLEMS (RCP) [KAN83]

In this section, we describe the RCP problem and some existing algorithms for solving it. These algorithms are used in Section 5.3 for verifying strong equivalence. Based on these algorithms, we can derive efficient algorithms for verifying weak and EB equivalences (Sections 5.4 and 5.5).

It is very unlikely that efficient (i.e., polynomial-time) algorithms will ever be found for the other equivalence relations, such as trace equivalence (similar to traditional language equivalence), and testing (failure) equivalence. Both problems have been shown to be PSPACE-complete [KAN83] (Section 5.6).

A *partition* of a set S consists of nonempty disjoint subsets called *blocks* of S whose union is S itself. The relational coarsest partition (RCP) problem is described below:

The Relational Coarsest Partition (RCP) Problem

Given a set S, an initial partition $\Pi^0 = \{ E_1, E_2, \ldots, E_p \}$ of S and a function $F : S \rightarrow 2^S$, derive a new partition $\Pi' = \{ B_1, B_2, \ldots, B_q \}$ of S such that

i. Π' is consistent with Π^0 in the sense that every B_k is a subset of some E_j ;

ii. for every pair p, q in block B_k and any block B_j of Π', $F(p) \cap B_j \neq \emptyset$ iff $F(q) \cap B_j \neq \emptyset$; and

iii. Π' has the smallest number of blocks.

The RCP problem is said to be *bounded fanout* if there exists a constant C such that, for every $p \in S$, $|F(p)| <= C$, where |X| denotes the number of elements of X. Obviously, for finite LTSs, any RCP problem is bounded fanout.

The function F can be represented as a directed graph G whose node set is S and arc set is { (p, q) | p, q ∈ S, q ∈ F(p)}. The size of RCP is (N, M), where N is the number of nodes and M is the number of arcs of the graph. In the particular case where F: S → S and thus M = N, the RCP problem can be solved by an O(NlogN) algorithm [AHO74]. In general, we can use the algorithms described below [KAN83].

Algorithms for solving the RCP problem

An obvious solution to the RCP problem is to start with the initial partition $\Pi' = \Pi^0 = \{ E_1, E_2, \ldots, E_p \}$ and refine the blocks of Π' as follows. Let B_i be a block of Π'. For each $p \in B_i$, examine F(p). We partition B_i so that two elements p and q of B_i are put in the same subblock if

and only if, for any block B_j of Π', $F(p) \cap B_j$ and $F(q) \cap B_j$ are either both empty or both non-empty. This naive method is formally described below.

Algorithm RCP-1 (naive) [KAN83]
Input : A set S, a function $F : S \rightarrow 2^S$ and an initial partition Π^0 of S.
Output : A partition Π' which is a solution to the RCP problem.
Method :
Begin
Starting with i = 0, repeat the following process until $\Pi^i = \Pi^{i+1}$ for certain i. Then, let $\Pi' = \Pi^i$.
Process: Suppose $\Pi^i = \{ B_1, B_2, \ldots, B_p \}$. Obtain Π^{i+1} as follows: Partition each B_k of Π^i into sub-blocks $B_{k1}, B_{k2}, \ldots, B_{kr}$ in such a way that $p, q \in B_k$ are grouped into B_{km} iff, for every block B_j of Π^i, $F(p) \cap B_j$ and $F(q) \cap B_j$ are either both empty or both nonempty.
end

Lemma 5.1 Algorithm RCP-1 solves the RCP problem of size (N, M) in O(NM) time.
Proof: See [KAN83]. []

The complexity of the above naive algorithm can be improved substantially by a generalization of the divide-and-conquer method [AHO74]. The generalization appears in the form of an algorithm for solving the bounded fanout RCP problem in $O(C^2 N \log N)$ [KAN83]. First, we describe the concept behind this algorithm.

For any block B of a given partition Π^i, define C non-overlapping and possibly empty blocks, $F_0^{-1}(B), F_1^{-1}(B), \ldots, F_{C-1}^{-1}(B)$, where $F_j^{-1}(B) = \{ b \mid F(b) \cap B \neq \emptyset$ and there are exactly j other blocks B' of Π^i such that $F(b) \cap B' \neq \emptyset \}$.

Now, instead of partitioning a block B_k based on the image set F(b) of b in B_k as in the naive method, B_k is partitioned into at most C+1 blocks based on $F_i^{-1}(B)$ as follows:

$B_{ki} = \{ b \mid b \in B_k$ and $b \in F_i^{-1}(B) \}$, i = 0, 1, ..., C-1, and

$B_{kc} = B_k - \cup_i B_{ki}$, where the subscript i of $\cup_i$ and B_{ki} runs from 0 to C-1.

Once we have partitioned B_k with respect to B, we need not partition B_k further with respect to B unless B itself is split. However, if B itself is split into C+1 pieces $B^0, B^1, \ldots, B^c$, then Similar to [AHO74], we need not partition B_k further with respect to all of $B^0, B^1, \ldots, B^c$ but with respect to the C smallest of them.

The above method is formally described below.

Algorithm RCP-2 (divide-and-conquer) [KAN83]

Input : A set S, a function $F : S \rightarrow 2^S$, an initial partition Π^0 of S and $C = \max \{|F(p)| \mid p \in S\}$.

Output : A partition Π' which is a solution to the RCP problem.

Method :

```
Begin
BLOCKS := Π^0; WAITING := Π^0;  /*BLOCKS is the current partition. WAITING is the set of
                                    blocks waiting to be partitioned*/
While WAITING not empty do
    Begin
      Select and delete a block B_j from WAITING
      Compute F_i^-1(B_j) for i = 0,1,...,C-1
      Suppose BLOCKS = { B_1,B_2,.....,B_p }
      For k = 1,2,..,p,
      Begin
        If B_k satisfies the following conditions: a) B_k is not a subset of
        F_i^-1(B_j), for every i; and b) B_k ∩ (∪_i F_i^-1(B_j)) ≠ ∅,
        then
        Begin
          Delete B_k from BLOCKS
          Partition B_k into C+1 new subblocks B_ki = { b | b ∈ B_k and b ∈ F_i^-1(B_j) }, i = 0, 1,
          ...., C-1, and B_kC = B_k - ∪_i B_ki .
          Put the nonempty new blocks into BLOCKS.
          If B_k is in WAITING then delete it from WAITING and add all nonempty new
          subblocks to WAITING else add the C smallest subblocks of B_k to WAITING;
        end /* end if */
      end /* for */
  end /* while */
    Return with Π' = BLOCKS.
end
```

Theorem 5.1 The bounded fanout RCP problem is solved by Algorithm RCP-2 in time $O(C^2 N \log N)$, where N is the number of elements of S.

Proof: See [KAN83]. □

5.3 AN ALGORITHM FOR VERIFYING STRONG EQUIVALENCE

In this section, we show that, for a given LTS = < K, Σ, Δ, ->, the problem of verifying the strong equivalence of two of its states can be reduced to solving $|\Sigma|$ instances of RCP iteratively. In each iteration, the function F is defined in terms of an action $a \in \Sigma$ as $F_a : S \rightarrow 2^S$, where $F_a(p) = \{q \mid (p,a,q) \in \Delta\}$. In the first iteration, the initial partition Π^0 is {K}; and, in the ith iteration, the initial partition is the outcome of the (i-1)th iteration.

Algorithm Verifier-strong [KAN83]
Input : An LTS = < K, Σ, Δ, ->, where $\Sigma = \{a_1, a_2, ..., a_{|\Sigma|}\}$, and $p, q \in K$.
Output : A partition Π' of K for making the following verdict.
Verdict : $p \equiv^s q$ iff p and q belong to the same block of Π'.
Method :
Begin
    $\Pi := \{K\}$.
    repeat
        For evety F_{ai} (i =1,2,.., $|\Sigma|$), $\Pi := RCP(K, F_{ai}, \Pi)$.
        (Note: The above notation means that the newly generated partition Π is the output of Algorithm RCP-1 or Algorithm RCP-2 with inputs K, every F_{ai} and previous partition)
    until no changes are made on Π.
    $\Pi' := \Pi$.
End.

Theorem 5.2 For two states p and q of an LTS with N states and M transitions, Algorithm Verifier-strong verifies $p \equiv^s q$ by solving $|\Sigma|$ RCP problems.

Proof: (sketchy) We provide a proof different from the one given in [BOL87]. Suppose p and q belong to the same block of Π'. Then, solving the $|\Sigma|$ RCP problems means that, for every $a \in \Sigma$, if p -- a --> p' and q -- a --> q', then p' and q' belong to the same block of Π'. By considering each block of Π' as an equivalent class for strong equivalence, this means that if p' and q' are strongly equivalent, so are p and q. []

Corollary 5.1 [KAN83] For two states p and q of an LTS with N states and M transitions, $p \equiv^s q$ can be verified in $O(|\Sigma| NM)$ time. In particular, if there are at most C transitions departing every state, $p \equiv^s q$ can be verified in $O(|\Sigma| C^2 N \log N)$ time.

Proof: By Lemma 5.1, Theorem 5.1 and Theorem 5.2. []

5.4 AN ALGORITHM FOR VERIFYING WEAK EQUIVALENCE

In Section 3.1, we prove that the weak equivalence of two states of an LTS can be verified by showing that they are strongly equivalent in LTS', which is derived from LTS by using Transformation Rule R1. In the transformation, the main step is to compute Δ'. Based on the definition of $\Longrightarrow$, $p \Longrightarrow a \Longrightarrow q$ iff ($p \Longrightarrow \varepsilon \Longrightarrow p'$, $p' - a - > q'$ and $q' \Longrightarrow \varepsilon \Longrightarrow q$). Δ' can be computed as follows:

For each pair of states p and q, if there exist $p', q' \in K$ and $a \in \Sigma_{obs}$ such that (p', a, q') in Δ, $p \Longrightarrow \varepsilon \Longrightarrow p'$ and $q' \Longrightarrow \varepsilon \Longrightarrow q$, then transition (p,a,q) is added into Δ'.

The following algorithm and theorem show that the problem of weak equivalence verification can be solved in $O(N^3)$.

Algorithm Verifier-weak [KAN83]

Input : An LTS = < K, Σ, Δ, - > and p, q $\in$ K.

Output : A partition Π' of K for making the following verdict.

Verdict : $p \equiv^w q$ iff p and q belong to the same block of Π'.

Method :

Begin

Obtain LTS' = < K, Σ, Δ', - > from LTS by applying Transformation Rule R1.

Π' := Verifier-strong (LTS', Π^0), where $\Pi^0 = \{K\}$.

(Note: The above notation means that Π' is the output of Algorithm Verifier-strong with inputs LTS' and Π^0).

End.

Theorem 5.3 [KAN83] For two states p and q of an LTS with N states, Algorithm Verifier-weak checks $p \equiv^w q$ in $O(N^3)$.

Proof: We first use a transitive closure algorithm [AHO74] to compute the transition relation $\Longrightarrow a \Longrightarrow$ from Δ, $a \in \Sigma$. This requires $O(N^3)$ time and results in a new LTS with at most $O(N^2)$ more edges than the original one.

Theorem 3.1 says that $p \equiv^w q$ in the original LTS iff $p \equiv^s q$ in the new LTS. By Corollary 5.1, the verification of $p \equiv^s q$ requires $O(|\Sigma| C^2 N \log N)$ time in the new LTS. Thus, the time complexity of Verifier-weak is dominated by the transitive closure operation. []

5.5 AN ALGORITHM FOR VERIFYING EB EQUIVALENCE

In Section 3.2, we prove that the EB equivalence of two states of an LTS can be verified by showing that they are weakly equivalent in LTS', which is derived from LTS by using Transformation Rule R2. Based on this transformation, the following algorithm can be used to verify EB equivalence of two states.

Algorithm Verifier-EB

Input : An LTS = < K, Σ, Δ, - > and p, q $\in$ K_O, where K_O is the observable subset of K.

Output : A partition Π' of K_O for making the following verdict.

Verdict: $p \equiv^{eb} q$ iff p and q belong to the same block of Π'.

Method :

Begin

Obtain $LTS' = <K_O, \Sigma, \Delta', - >$ from LTS by applying Transformation Rule R2.

$\Pi' :=$ Verifier-weak (LTS').

(Note: The above notation means that Π' is the output of Algorithm Verifier-weak with input LTS').

End.

Theorem 5.4 For two states p and q of an LTS with N states, Algorithm Verifier-EB checks $p \equiv^{eb} q$ in $O(N^3)$ time.

Proof : We first identify all unobservable states. This step requires $O(N^2)$ time. Then, we delete all such states by using Transformation Rule R2. It needs $O(N^3)$ time to do so since there are at most O(N) different unobservable states on the path between any pair of states p and q and we have to inspect $O(N^2)$ pairs of states. Lastly, we have to decide if $p \equiv^{w} q$ in LTS'. This requires $O(N^3)$ time. []

5.6 AN ALGORITHM FOR VERIFYING TESTING EQUIVALENCE [BOL89]

In Section 3.4, we prove that the testing equivalence of LTS1 and LTS2 can be verified by showing that LTS1' and LTS2' are extended trace equivalent, where LTS1' and LTS2' are derived from LTS1 and LTS2 by using Transformation Rule R3, respectively. The following algorithm solves the testing equivalence verification problem.

Algorithm Verifier-testing [BOL89]

Input : An LTS = $< K, \Sigma, \Delta, ->$ and $p, q \in K$.

Output : A partition Π' of K' and $p', q' \in K'$ for making the following verdict.

(Note: $LTS' = <K', \Sigma_{obs}, \Delta', ->$ is transformed from LTS by using Transformation Rule R3, p' and q' represent the subset of K including p and q, respectively.

Verdict: $p \equiv^t q$ iff p' and q' belong to the same block of Π'.

Method :

Begin

Obtain $LTS' = <K', \Sigma_{obs}, \Delta', ->$ from LTS by applying transformation Rule R3.

$\Pi' :=$ Verifier-strong (LTS', Π^0), where Π^0 is the initial partition of K' in which the states of K' are grouped together according to their MO-labels.

(Note: The above notation means that Π' is the output of Algorithm Verifier-strong with inputs LTS' and Π^0).

End.

It has been proved [KAN83] that the verification of failure equivalence between two LTSs is a *PSPACE-complete* problem. Since testing equivalence is identical with failure equivalence in strongly convergent LTSs, the verification of testing equivalence is also PSPACE-complete.

Theorem 5.6 Let p, q be two states of an LTS. Checking whether $p \equiv^f q$ is *PSPACE-complete.*

proof: See [KAN83]. []

By Theorem 2.2 and Theorem 5.6, we have the following result:

Corollary 5.2 Let p, q be two states of LTSs. Checking whether $p \equiv^t q$ is a PSPACE-complete problem.

Chapter 6

IMPLEMENTATION OF ALGORITHMS FOR VERIFYING EQUIVALENCE RELATIONS

6.1 INTRODUCTION

In this chapter, we present the second major contribution of this thesis: Implementation of the verification algorithms described in Chapter 5 in a system called LTS-EVS (Labeled Transition System Equivalence Verification). At the current stage of implementation, LTS-EVS accepts as inputs the matrix representation of two indeterministic labeled Petri-nets, derives their state-transition matrices representing the two LTSs internally and outputs the equivalent classes of their union. States which are equivalent should belong to the same equivalent class. LTS-EVS can be extended to accept the state-transition matrices derived from other models, such as finite state machine and LOTOS.

LTS-EVS is an interactive menu-driven system. It can be used as an independent tool or be integrated as a subsystem of UO-GLOTOS [CHE89a], a graphical LOTOS environment for the specification, verification and testing of distributed systems. LTS-EVS is implemented with the C language on a SUN Workstation under the UNIX system.

6.2 PETRI-NETS WITH INDETERMINISTIC BEHAVIOR

In this section, we introduce a labeled Petri-net model, one of the input forms (in fact, the only input form already implemented) to LTS-EVS. It can be used to describe the behavior of indeterministic systems. This model has been used in UO-GLOTOS [CHE90b] for expressing the control flow of LOTOS specifications.

Definition 6.1 (IPN, indeterministic labeled Petri-net) [CHE90b]

An indeterministic labeled Petri-net IPN is a 6-tuple $< P, T, F, M_0, L, h >$, where

P is a non-empty set of places;

T is a set of transitions, where $P \cap T = \emptyset$;

F is a set of arcs, i.e., $F \subseteq (P \times T) \cup (T \times P)$;

M_0 is the initial marking;

$L = L_e \cup \{\tau\} \cup L_s$ is a set of labels, where L_e is the set of external labels, τ is the internal label and L_s is the set of special labels (such as terminations); and

h: is a mapping $T \rightarrow L$.

Remember that a marking is a vector the value of whose jth element denotes the number of tokens existing in the jth place of IPN. In comparing an IPN with an LTS = < K, Σ, Δ, p_0 >, we see that the set of all reachable markings from the initial marking M_0 corresponds to K; L corresponds to Σ; $L_e \cup L_s$ corresponds to Σ_{obs}; the set {M[t >M' | M and M' are markings, $t \in T$, $h(t) \in L$} corresponds to the transition relation Δ; and M_0 corresponds to p_0.

The techniques of transforming LOTOS specifications to IPN have been developed in [CHE90b]. LTS-EVS includes a subsystem for deriving the state-transition matrices from the matrix representation of an IPN.

6.3 FUNCTIONS OF LTS-EVS

Functionally, LTS-EVS is divided into the following two subprograms (Figure 6.1) :

(1) DERIVATION first accepts the matrix representation of two IPNs and derives their state-transition matrices (marking transition graphs) from the input IPNs. These state-transition matrices represent the labeled transition systems internally.

(2) EQUIVALENCE VERIFIER outputs the equivalent classes according to the equivalence relation specified by the user.

In the following sections, some implementational detail of these two subprograms is given.

6.4 SUBPROGRAM - DERIVATION

DERIVATION derives the Marking Transition Graph from the input matrix representation of an IPN by using the depth-first search technique and a hashing technique. The data structures, algorithms, and functions of DERIVATION are described in the following subsections.

6.4.1 Data structures used in DERIVATION

DERIVATION uses three main data structures: the Marking Transition Graph matrix MT; the Hash_Map, a bit state space for reducing the search time for new states; and a stack for depth-first search. These structures are briefly described below.

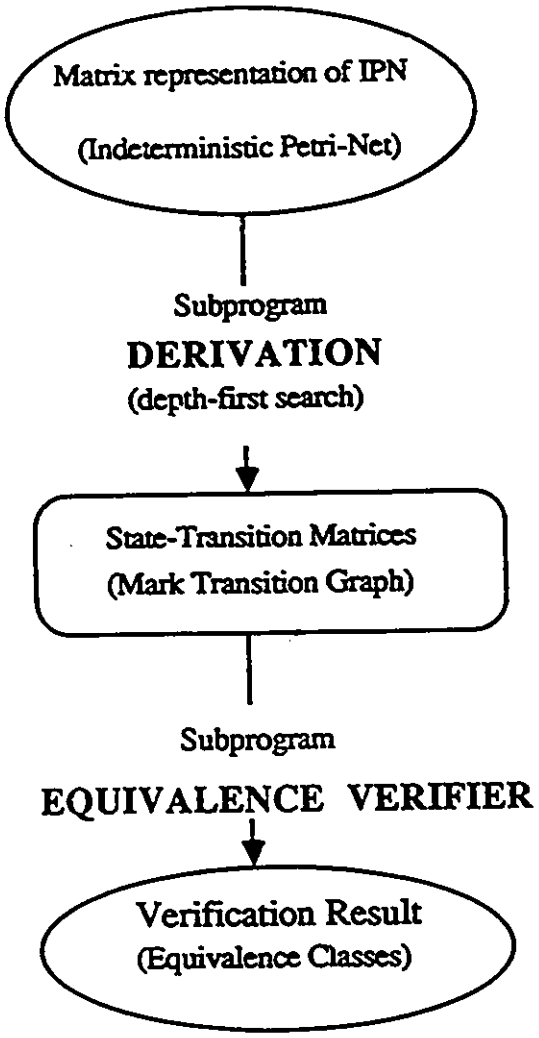


Figure 6.1 Functional diagram of Equivalence Verification System LTS-EVS.

1) Marking Transition Graph matrix MT

MT is the output matrix of DERIVATION. This matrix represents the markings (states) and their transition relation. It is an M×M matrix, where M is the number of markings in the Marking Transition Graph. MT[i, j] contains the set of actions $\{a_1,a_2,..,a_n\}$, any of which is firable at the ith marking and leads to the jth marking

Field	Data type	Usage
tran	integer	the set of transitions between markings
len	integer	the size of tran

Figure 6.2 The two fields of an element of matrix MT.

2) Hash_Map

This bit state space is used for hashing a marking vector to the state space. The use of a hashing technique in DERIVATION greatly reduces the search time for deciding if a newly generated marking is new or not. For each marking vector, we first hash it as a 16-bit hash value vector. In this vector, the ith bit is a number transformed from the ith segment (each consists of 16 bytes) of the marking vector. Then, we use the folding method in hashing to hash this 16-bit vector to a fixed size bit state space Hash_Map (1..2^{16}=65536). For a given marking M, Hash_Map(hash(M)).mark = 1 means that M is an "old" state otherwise M is a "new" state. If M is a new state, Hash_Map(hash(M)).order = i means M is ith new state in Marking Transition Graph. (See Figure 6.4)

Field	Data type	Usage
mark	integer	flag of new state
order	integer	sequential number assigned to new state

Figure 6.3 The two fields of each marking vector in Hash_Map.

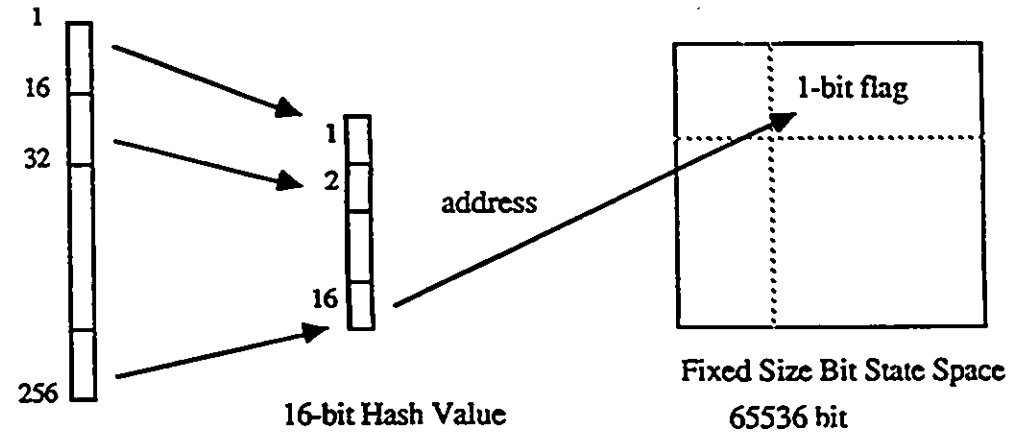


Figure 6.4 The hash processing used in subprogram DERIVATION

6.4.2 Algorithm used in DERIVATION

Algorithm TRANS_GRAPH is the only algorithm within DERIVATION. It is used to produce a Marking Transition Graph from the matrix representation of the given IPN. The depth-

first search technique is used to generate all the reachable markings from its initial marking. The algorithm is iterative. In each iteration, it generates all the markings reachable from a current marking M. Then, these reached markings are mapped into a bit state space to check if they are new or not. If a reached marking is new, it is assigned a sequential number and its flag is changed. After that, the transition relation between the current marking M and the reached markings is built up and stored in the Marking Transition Graph matrix. A pseudo-code description of the algorithm is given below.

Algorithm TRANS_GRAPH

Input : D^- and D^+, the matrix representation of an IPN.

Output : MT, the matrix representation of the Marking Transition Graph of the IPN.

Method:

```
Begin
    Let M = initial marking of IPN
    Put M into MT
    Push M into stack
    While stack <> Ø do
      Begin
        Pop up a marking from stack and assign it to M
        Find all firable transitions X at M
        Generate all the reachable markings M' from M based on the computation
        M' = M+X.D, where D = D- - D+
        For each reached marking M' do:
         Begin
           Check whether M' is a new marking or not by hashing it into MT
           if M' is new
              Begin
                put M' into MT
                push M' into stack
              end /*if*/
           Save the transition relation between M and M' in MT
         end /*For*/
    end /*While*/
End /*METHOD*/
```

6.5 SUBPROGRAM - EQUIVALENCE VERIFIER

By accepting as input the state-transition matrices of two LTSs produced by DERIVATION, EQUIVALENCE VERIFIER verifies if the two LTSs are equivalent or not by generating all the equivalent classes and checking if the two initial states belong to the same class or not. In fact, it implements those verification algorithms described in Chapter 5. In this section, we describe the data structures used in EQUIVALENCE VERIFIER and the control flow among the various verification algorithms.

6.5.1 Data structures used in EQUIVALENCE VERIFIER

EQUIVALENCE VERIFIER uses the following main data structures : MT, the matrix representation of an LTS (Section 6.3.1) and a set of the equivalent classes PAI generated by EQUIVALENCE VERIFIER. Each equivalent class is represented by a linear table. Also, a closure matrix CLO for computing the weak reachability, a set of the deterministic states DSTATE in which each state consists of a set of the indeterministic states in LTS, a MO-label table MO_L for each of the deterministic states are used in EQUIVALENCE VERIFIER, where DSTATE and MO_L are used in the verification of testing equivalence $\equiv^t$.

1) The set of the equivalent classes PAI.

This set is the output of EQUIVALENCE VERIFIER. PAI[i] is a linear link table which represents the ith equivalent class (block). During the partition, each block is built up or split in the PAI.

Field	Data type	Usage
state	integer	the state belong to Block
father	pointer	point to front state
son	pointer	point to back state
link	array	a Block, linear table of states
len	integer	the length of the link

Figure 6.5 The five fields of each block in the equivalence class set PAI.

2) The set of the deterministic states DSTATE

In the verification of testing equivalence, the first step is to transform an indeterministic LTS into an deterministic LTS. Each state in the deterministic LTS consists of a set of the equivalent states in the indeterministic LTS. DSTATE is the set of states in the

deterministic LTS. DSTATE[i] is the ith deterministic state which is a linear table including all the equivalent states of the indeterministic LTS and DS_LEN[i] is the size of DSTATE[i].

3) The MO_label table MO_L

During the determinization of an LTS, we first generate a new deterministic state p from the indeterministic LTS; Then, we compute MO-label for state p. MO-label is a set of sets with respect to the all the firable transitions at state p. For example, suppose in the deterministic LTS, $p == a \Longrightarrow p'$, then the set $O_w(p')=\{a_1, a_2,..., a_m\}$ is in MO-label of state p, where $a, a_1, a_2,..., a_m \in \Sigma_{obs}$. In the verification of testing equivalence, the initial partition of the states is obtained by grouping together those states with the same MO-label.

Field	Data type	Usage
state	integer	the reachable state
act	actionlist	the set of all firable actions from state i.e., act = O_w(state)
len	integer	the size of act

Figure 6.6 The three fields of the weakly observable actions of a state.

Field	Data type	Usage
outlabel	out	the set of outs for each action.
len	integer	the size of outlabel.

Figure 6.7 The fields of MO-label of a deterministic state.

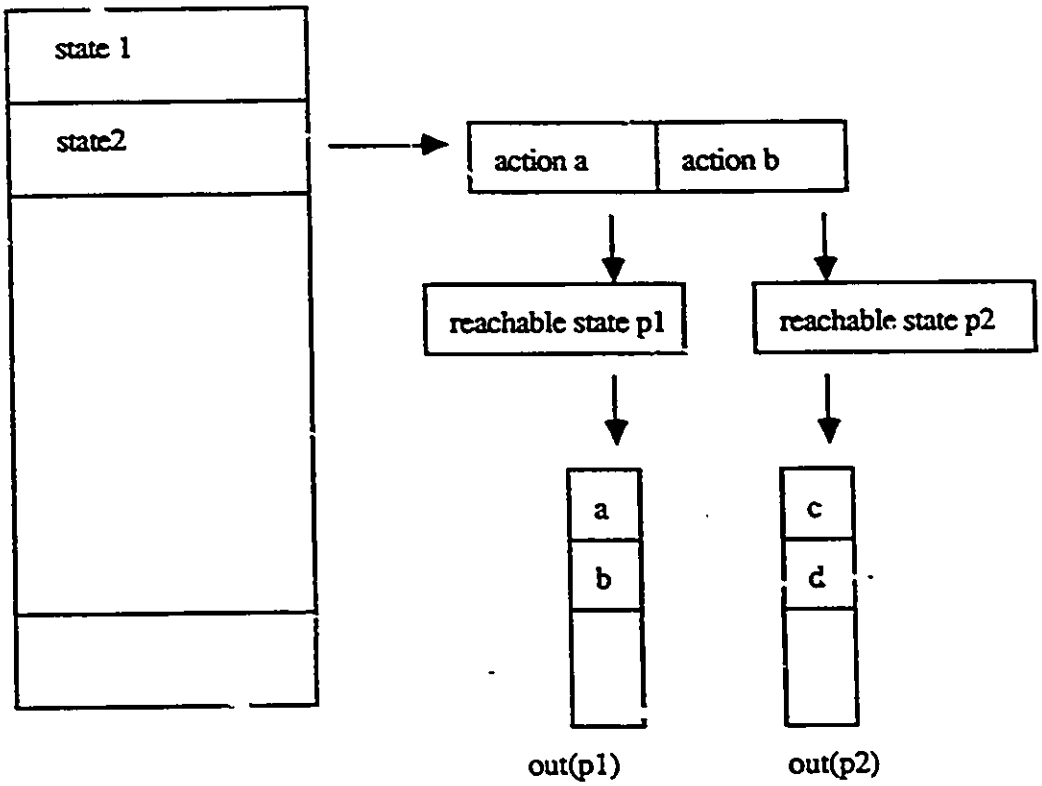


Figure 6.8 The data structures used for an MO-label.

6.5.2 Implementational control flow among various verification algorithms

The main algorithm used in EQUIVALENCE VERIFIER is Algorithm RCP-2 of Section 5.2. This algorithm can be used directly for the verification of strong equivalence. Since weak equivalence can be reduced to strong equivalence (by Transformation Rule R1), EB equivalence can be reduced to weak equivalence (by Transformation Rule R2), and testing equivalence can be reduced to the strong equivalence (by Transformation Rule R3), all the algorithms are basically dependent on RCP-2. Figure 6.9 shows the implementational control flow among these algorithms.

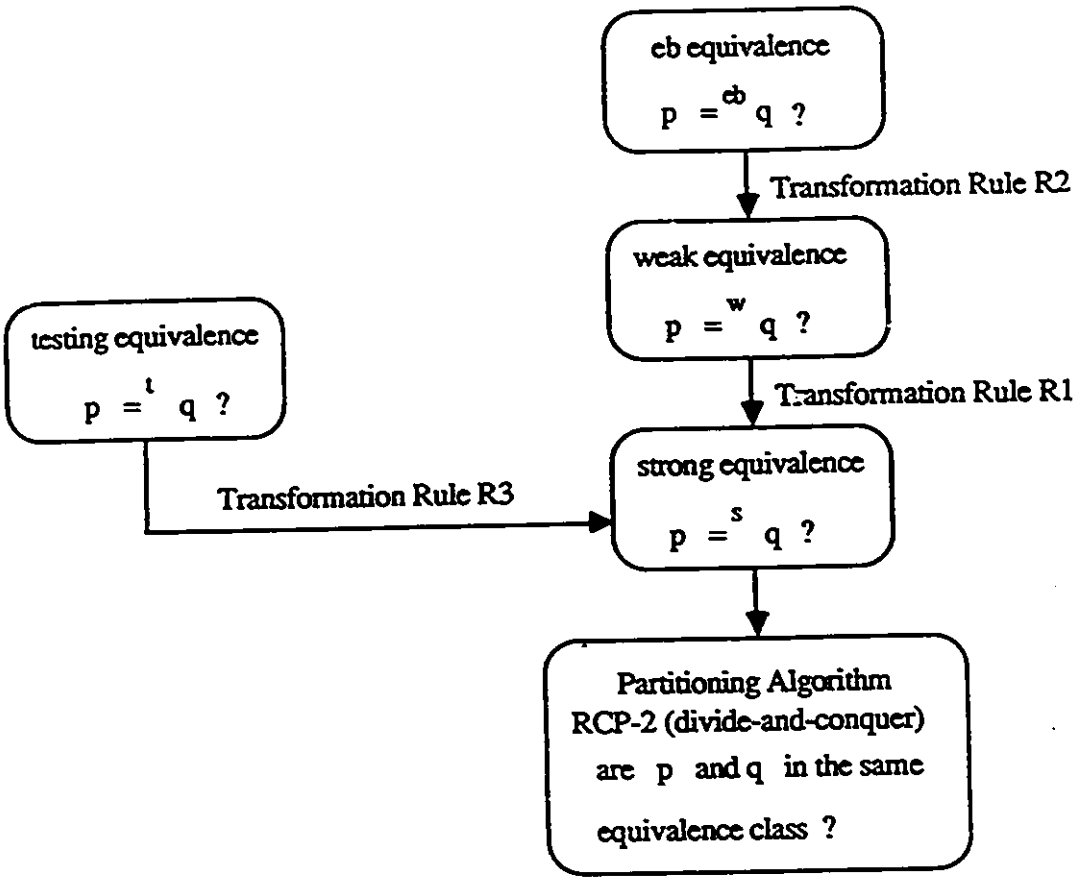


Figure 6.9 The implementational control flow among various algorithms.

6.6 SOME EXPERIMENTAL RESULTS

In this section, we show part of our experimental results of applying LTS-EVS by two examples. In these examples, the inputs to LTS-EVS are two labeled transition systems LTS1 and LTS2 and a user-specified equivalence relation and the outputs are the equivalence classes.

Example 6.1

In this example, the weak equivalence of two LTSs is verified by applying LTS-EVS (Figure 6.10). The inputs are the state-transition matrices of LTS1 and LTS2. Algorithm Verifier-weak first obtains LTS1" and LTS2" by using Transformation Rule R1, then outputs the equivalent classes as follows:

Class #1 = $\{p_3, p_5, p_6, q_2, q_4\}$

Class #2 = $\{p_2, p_4, q_3\}$

Class #3 = $\{p_1, q_1\}$

Class #4 = $\{p_0, q_0\}$

Since p_0 and q_0 belong to the same equivalent class, LTS1 and LTS2 are weakly equivalent.

Example 6.2

In this example, the testing equivalence of two LTSs is verified by applying LTS-EVS (See Figure 6.10). The inputs are the state-transition matrices of LTS1 and LTS2. Algorithm Verifier-testing first obtains LTS1" and LTS2" by using Transformation Rule R1 (Step 1 in Rule R3), then obtains LTS1' and LTS2' by a "subset construction" algorithm [AHO88] (Step 2 in Rule R3). Each subset of states in LTS1" (resp., LTS2") becomes a state of LTS1' (resp., LTS2'). For example, $\{p_1, p_2, p_4\}$ of LTS1" becomes state p_1' of LTS1'.

In LTS1", Reach($\{p_0\}$, a, ==>) = $\{p_1, p_2, p_4\}$, Outs($\{p_1, p_2, p_4\}$) = $\{O_w(p_1), O_w(p_2), O_w(p_4)\}$ = $\{\{b, c\}, \{c\}\}$. Thus, MinOuts(Reach($\{p_0\}$, a, ==>)) = $\{\{c\}\}$. By the Step 3 of Transformation Rule R3, state p_1' of LTS1' is assigned the MO-label $\{\{c\}\}$. Similarly, in LTS1', $p_0' = \{p_0\}$, $p_1' = \{p_1, p_2, p_4\}$, $p_2' = \{p_3\}$, $p_3' = \{p_5, p_6\}$. p_0' has MO-label $\{\{a\}\}$, p_1' has MO-label $\{\{c\}\}$, p_2' and p_3' have MO-label $\{\{\emptyset\}\}$. In LTS2', $q_0' = \{q_0\}$, $q_1' = \{q_1, q_3\}$, $q_2' = \{q_2\}$, $q_3' = \{q_4\}$. q_0' has MO-label $\{\{a\}\}$, q_1' has MO-label $\{\{c\}\}$, q_2' and q_3' have MO-label $\{\{\emptyset\}\}$.

After obtaining LTS1' and LTS2', Algorithm Verifier-testing verifies if they are extended trace equivalent (Definition 3.2) or not. The following is the output of Verifier-testing:

Class #1 = $\{p_2', p_3', q_2', q_3'\}$

Class #2 = $\{p_1', q_1'\}$

Class #3 = $\{p_0', q_0'\}$

Since p_0' and q_0', and thus p_0 and q_0 belong to the same equivalent class, LTS1 and LTS2 are testing equivalent.

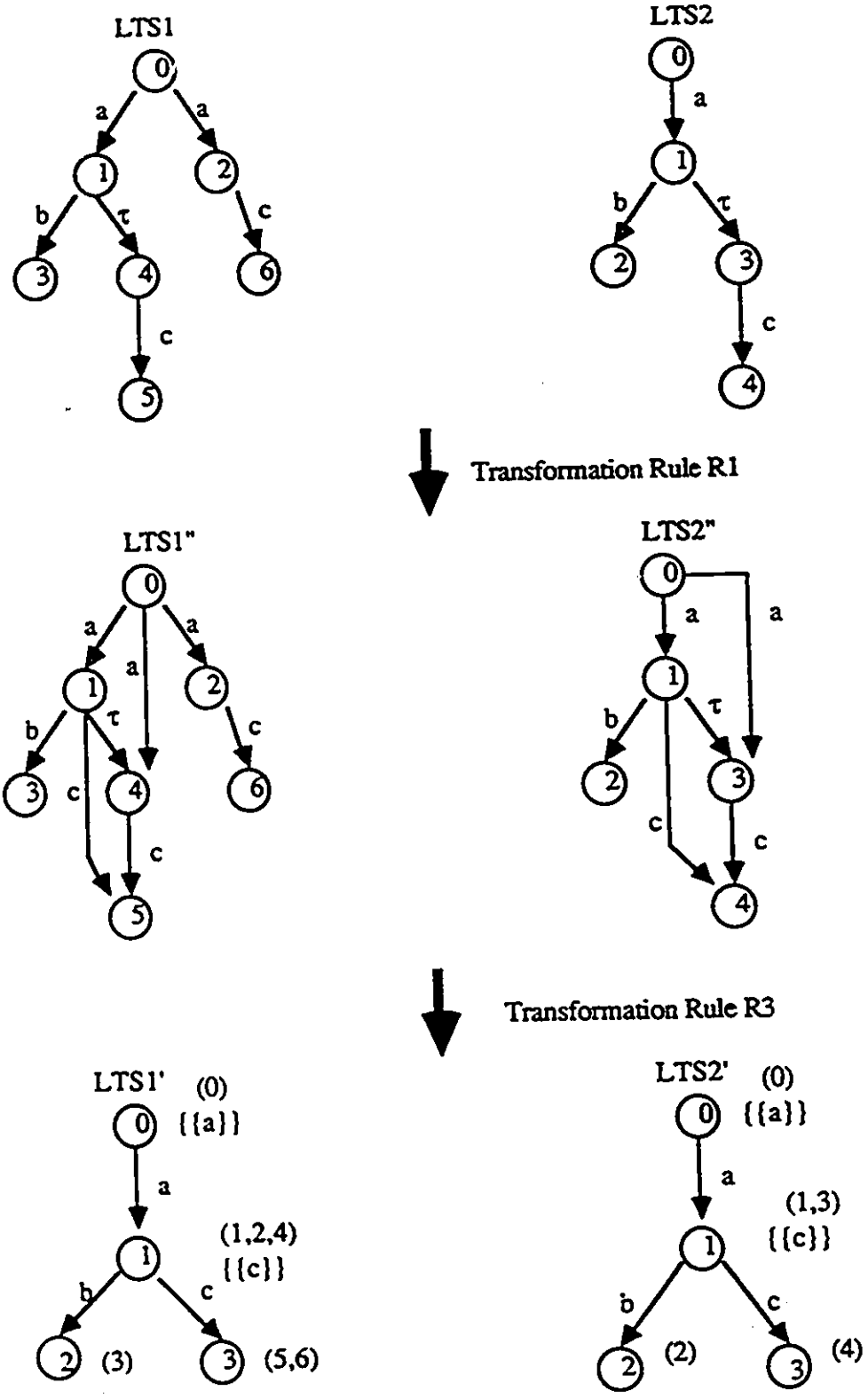


Figure 6.10 Experimental results of verifying weak equivalence and testing equivalence.

Example 6.3 (Figure 6.11)

This example shows two LTSs which are EB equivalent but not weakly equivalent. This can be shown by comparing the outputs of Verifier-EB and Verifier-weak as follows.

The output of Verifier-EB is (Note that unobservable states p_7 and p_8 have been eliminated from LTS2 after transformation by Rule R2) :

Class #1 = {p_3, p_{11}, p_{12}, p_{15}, p_{16}, q_2, q_7, q_8}

Class #2 = {p_9, p_{13}, q_5}

Class #3 = {p_{10}, p_{14}, q_6}

Class #4 = {p_5, p_6, q_4}

Class #5 = {p_2, p_4, q_3}

Class #6 = {p_1, q_1}

Class #7 = {p_0, q_0}

Since p_0 and q_0 belong to the same equivalent class, LTS1 and LTS2 are EB equivalent.

The output of Verifier-weak is :

Class #1 = {p_3, p_{11}, p_{12}, p_{15}, p_{16}, q_2, q_7, q_8}

Class #2 = {p_9, p_{13}, q_5}

Class #3 = {p_{10}, p_{14}, q_6}

Class #4 = {q_4}

Class #5 = {p_7, p_8}

Class #6 = {q_3}

Class #7 = {p_5, p_6}

Class #8 = {q_1}

Class #9 = {p_2, p_4}

Class #10 = {p_1}

Class #11 = {p_0}

Class #12 = {q_0}

Since p_0 and q_0 do not belong to the same equivalent class, LTS1 and LTS2 are not weakly equivalent.

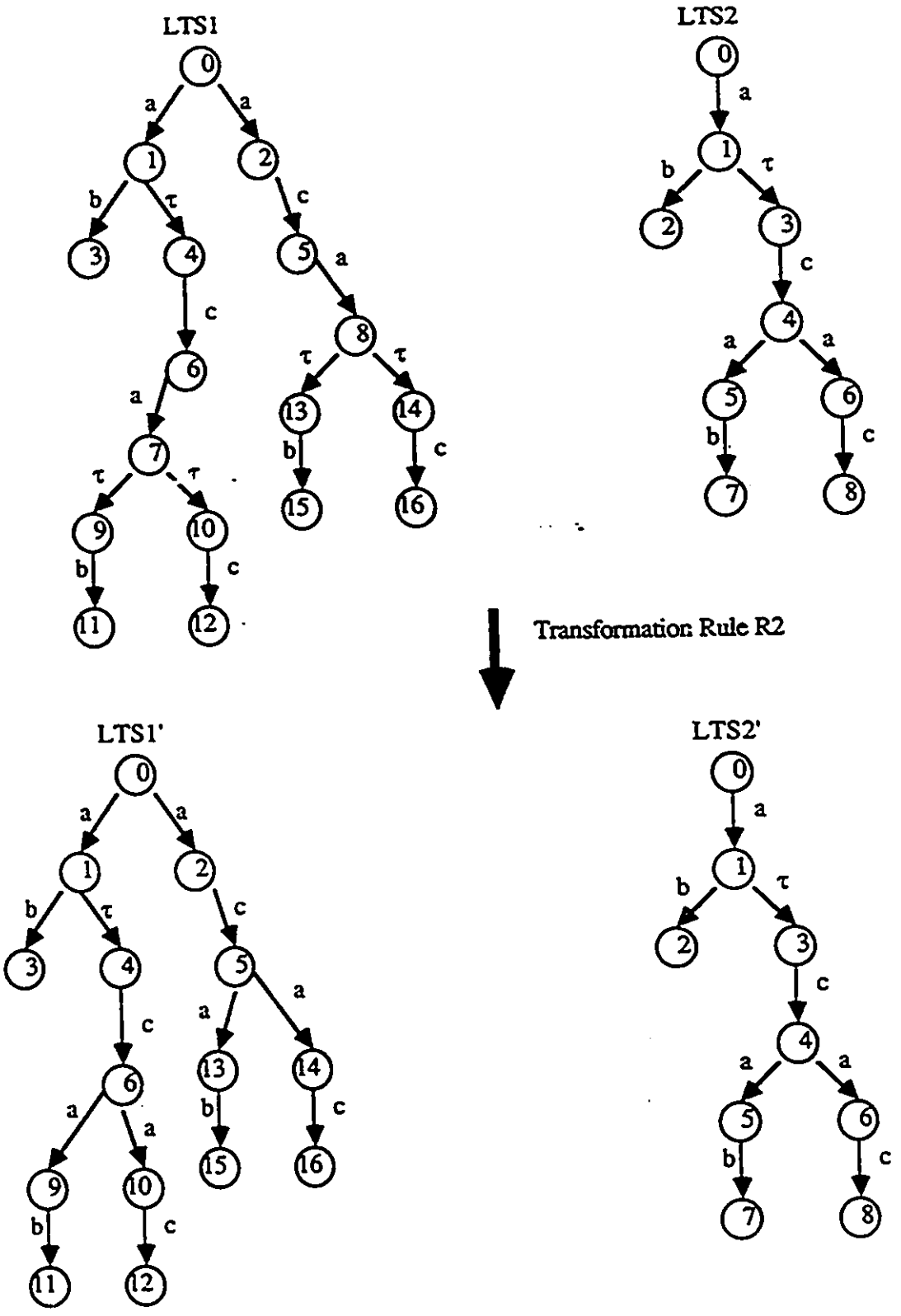


Figure 6.11 An experimental result of EB equivalence.

Chapter 7

SUMMARY AND FUTURE WORKS

In this thesis, we have reported our research work concerning some of the problems of verifying distributed systems which exhibit indeterministic behavior. Our main results include the proposal of a unified definition of equivalence relation and the implementation of the various verification algorithms in a system called LTS-EVS on a Sun Workstation under the UNIX system. This unified view is based on several new types of reachability defined for indeterministic systems. In the literature, strong, weak, and testing equivalences are defined on labeled transition systems and EB equivalence is defined on petri-nets [MIL80, DEN84, KAN83, CIN85]. These definitions are based on different angles and lack a sense of uniformity. Our unified definition seems to be the first attempt to unify these concepts. In addition, there are many relatively less important results: EB equivalence is redefined in terms of labeled transition systems. A new transformation rule by which EB equivalence in an LTS can be reduced to weak equivalence in another LTS is proposed. A new polynomial algorithm is presented for verifying EB equivalence. A proof that EB equivalence is stronger than testing equivalence in non-strongly convergent labeled transition systems and a disproof of the conclusion existing in literature that EB equivalence is stronger than weak equivalence [POM86] are provided. Also, many properties which are satisfied and preserved under different transformations are derived.

We can foresee some future research developments based on our work, as described briefly below.

At present, LTS-EVS is implemented as an independent system. A set of small-sized LTSs have been used for checking its performance. In the future , we shall integrate LTS-EVS into UO-GLOTOS so that it can be used for verifying large LOTOS specifications. More detail follows.

UO-GLOTOS is an interactive menu-driven LOTOS-based graphical system for the specification, verification and testing of communication protocols. UO-GLOTOS has been developed under the direction of Cheung [CHE89a] for research in LOTOS. UO-GLOTOS is based on a limited set of simple graphical patterns for representing the LOTOS constructs and a hierarchical structure for representing the control flow of a LOTOS specification. LTS-EVS can be integrated into UO-GLOTOS as a subsystem. The interface between LTS-EVS and UO-GLOTOS is IPN, the petri-net representation of a LOTOS specification. An organizational diagram of the entire system UO-GLOTOS is shown in Figure 7.1.

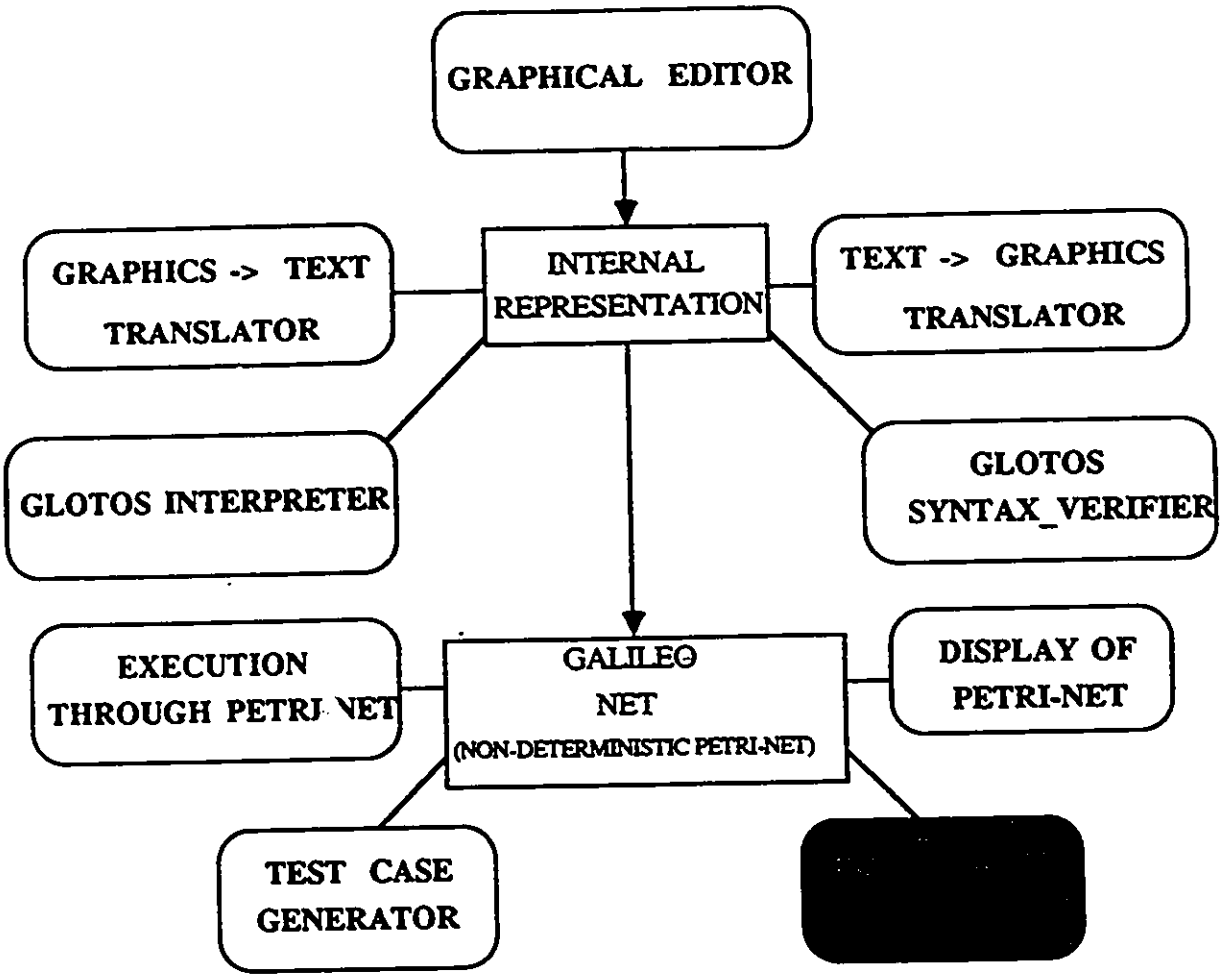


Figure 7.1 A LOTOS-based graphical environment for distributed computing research at University of Ottawa (The shaded component is LTS-EVS)

The verification process of some equivalence relations requires the exhaustive state-by-state comparison of reachable sets. This leads to the problems of state explosion as in classical reachability analysis. A computational limitation is very serious for efficient implementation. Testing equivalence seems more flexible than the other three types of equivalence. It verifies the system behavior by comparing the actions acceptable by the reachable sets collectively. This approach 'hides' the indetermininstic behavior of the systems at the individual states. Unfortunately, the verification of testing equivalence is a PSPACE-complete problem [KAN83], meaning that it is unlikely to find a polynomial algorithm to solve the problem. Comparison problem is particularly serious for LOTOS specifications, for example, in which nested parallel operators are present and thus the size of the underlying LTS may grow exponentially. Two approaches may be adopted to deal with such a situation. One is to exploit partial or selective

verification methods similar to selective testing. Another is to develop a refinement technique for describing the behavior of systems at different abstract levels. This may require a semantic model based on modules. Several researchers have been working in a similar direction [CHE88, CIN85, GAL90]. Once we have obtained a model which describes the states in terms of processes and describes the concurrent behavior of the systems by using a real concurrency semantics, we may hope that verification and testing of LOTOS specifications of large sizes can be done within a reasonable time limit. Another direction for future work in the area is to find new computationally tractable equivalence relations whose strength is between weak equivalence (EB equivalence) and testing equivalence.

The study of properties preserved under transformation rules is another area for future research. The objective is to define a set of transformation rules for facilitating stepwise refinement design and simplifying the process of verification. These transformation rules should preserve important functionalities and structures. Several papers have discussed this issue for LOTOS [VIS88, LAN90].

Lastly, the use of these equivalence relations for conformance testing is a promising area of research. At present, all the major testing sequence generation methods are based on equivalence of deterministic finite state machines. Generation of testing sequences based on the equivalence of indeterminstic systems is a widely open field.

APPENDIX BASIC LOTOS

LOTOS (Language fOr Temporal Ordering Specification), an FDT developed within the Open System Interconnection (OSI) Architecture, is now an ISO standard [ISO8807]. The full LOTOS syntax has essentially two parts: the control part which describes the interaction among processes by a set of operators and the data part which consists of data structures. The control part uses a semantic model based on a modification of Milner's Calculus of Communicating System (CCS) [MIL80] to explain the behavior of LOTOS specifications. In LOTOS, a distributed system is described in terms of a set of interacting processes. A process is able to perform internal actions and communicate with other processes via actions at communications access point called gates. An action is an elementary unit of synchronization among processes. A process can be imagined as a black box capable of communicating with other processes. The mechanisms inside this box are unobservable and denoted by i. In LOTOS, processes can be defined in terms of the externally observable actions of the processes. The data structures are based on ACT ONE [EHR85], a specification language for the structural design of data types with equational axioms and initial semantics.

In this thesis, we consider only basic LOTOS, a subset of the full LOTOS constructs. It can be used to specify only the control part of a system. In basic LOTOS, processes interact with one another by pure synchronization, i.e., interactions without value passing. The syntax of basic LOTOS behavior expressions and related inference rules, which are used to explain these expressions, are listed in Figure 8.1.

Name	Behavior expression	Axioms or inference rules
inaction	stop	none
action prefix		
- unobservable	i; B	i; B --τ --> B
- observable	a; B	a ∈ L --- a; B --a --> B
choice	B1 [] B2	B1--a-->B1' --- B1 [] B2 --a--> B1'

$$\frac{B2 \text{ --a-->} B2'}{B1 \text{ [] } B2 \text{ --a--> } B2'}$$

parallel composition

- general case $B1 \mid [g_1,\ldots,g_n] \mid B2$

$$\frac{B1 \text{--a-->} B1',\ a \in \{g_1,\ldots,g_n,\delta\}}{B1 \mid [g_1,\ldots,g_n] \mid B2 \text{ --a-->} B1' \mid [g_1,\ldots,g_n] \mid B2}$$

$$\frac{B2 \text{--a-->} B2',\ a \in \{g_1,\ldots,g_n,\delta\}}{B1 \mid [g_1,\ldots,g_n] \mid B2 \text{ --a-->} B1 \mid [g_1,\ldots,g_n] \mid B2'}$$

$$\frac{B1 \text{--a-->} B1',\ B2 \text{--a-->} B2',\ a \in \{g_1,\ldots,g_n,\delta\}}{B1 \mid [g_1,\ldots,g_n] \mid B2 \text{ --a-->} B1' \mid [g_1,\ldots,g_n] \mid B2'}$$

- pure interleaving $B1 \,|||\, B2$

$$\frac{B1 \mid [\,] \mid B2 \text{ --a--> } B'}{B1 \,|||\, B2 \text{ --a--> } B'}$$

- full synchronization $B1 \,||\, B2$

$$\frac{B1 \mid [g_1,\ldots,g_n] \mid B2 \text{ --a-->} B'}{B \text{ --a--> } B'}$$

disabling $B1 \,[>\, B2$

$$\frac{B1 \text{--a-->} B1',\ a \neq \delta,}{B1 \,[>\, B2 \text{--a-->} B1' \,[>\, B2}$$

$$\frac{B1 \text{--}\delta\text{-->} B1'}{B1 \,[>\, B2 \text{--}\delta\text{-->} B1'}$$

		B2--a-->B2'
		B1 [> B2--a-- B2'
Successful termination	exit	exit--δ--> stop
hiding	B \ (g1,....gn)	B --a--> B', a ∈ {g1,....gn}
		B \ (g1,....gn) -- τ--> B' \ (g1,....gn)
		B --a--> B', a ∉ {g1,....gn}
		B \ (g1,....gn) --a--> B' \ (g1,....gn)

Figure 8.1 Axioms and inference rules for basic LOTOS behavior expressions

In Figure 8.1, L denotes the alphabet of observable actions; a, g_1,..g_n and δ denote actions; i and τ denote the internal action in behavior expressions and inference rules, respectively; B1 and B2 denote the behavior expressions.

From the inference rules listed in Figure 8.1, we can see that there are two cases of indeterminism in LOTOS specifications:

1) when the expression P = i; B is defined, the process P will have an internal behavior τ on unobservable action i.
2) When a hiding expression B \ (g_1,....,g_n) is used in parallel composition, it means that all the subsequent interactive actions g_1,....,g_n will be hidden. An external observer cannot perceive these evolutions. So the process will have internal behavior τ on all these hidden actions.

The following example is a LOTOS specification of the Connection Phase of the Class 0 Transport Protocol [BOL87a]. Figure 8.2 shows a labeled transition system of this LOTOS specification derived from the inference rules listed in Figure 8.1. We can see that action τ occurs once and twice in processes Calling and Called, respectively.

Example A LOTOS specification of the Connection Phase of Class 0 Transport Protocol.

```
process Connection_phase[CRQ,CI,CR,CC,DR,DI]: noexit :=
        ( i; Calling[CRQ,CI,CR,CC,DR,DI]
        [] Called[CRQ,CI,CR,CC,DR,DI]
        )
    where
      process Calling[CRQ,CI,CR,CC,DR,DI] () : exit :=
              CRQ; ( CC; exit [] DI; Connection-phase [CRQ,CI,CR,CC,DR,DI] )
      endproc

      process Called [ CRQ,CI,CR,CC,DR,DI] () : exit :=
              CI; ( i; CR; exit [] i; DR; Connection_phase [CRQ,CI,CR,CC,DR,DI] )
      endproc
endproc.
```

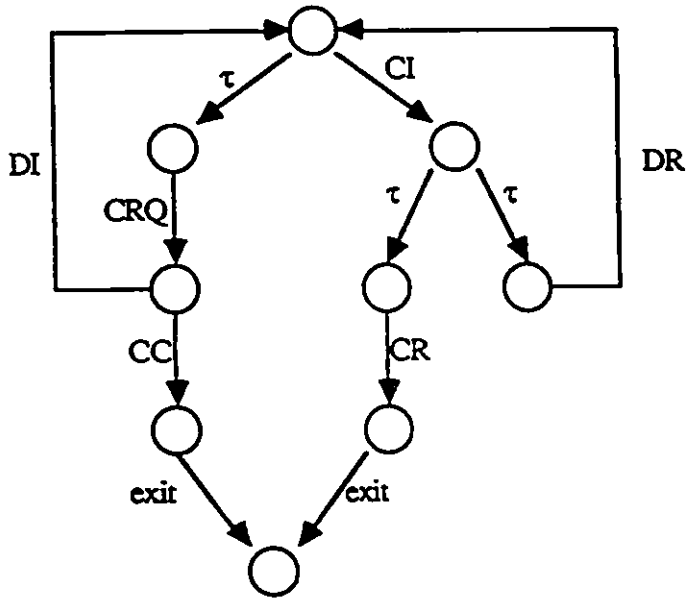


Figure 8.2 A LTS of the LOTOS specification of Connection Phase of the Class 0 Transport Protocol.

REFERENCES

[AHO74] A.V. Aho, J.E.Hopcroft and J.D.Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

[AHO88] A.V. Aho, R. Sethi, and J.D. Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, 1988.

[BOL87a] T. Bolognesi and E. Brinksma, "Introduction to the ISO Specification Language LOTOS", Computer Networks and ISDN Systems, 14 (1987), pp. 25-59.

[BOL87b] T. Bolognesi and S.A. Smolka, "Fundamental results for the verification of observational equivalence : a survey", Proc. Seventh International IFIP WG 6.1 Symposium on Protocol Specification, Testing, and Verification. Zurich, Switzerland, (1987), pp. 165-179.

[BOL89] T. Bolognesi and M. Caneve, "Equivalence verification : theory, algorithms and a tool", The Formal Description Techniques LOTOS, editors P.H.J. Van Eijk, C.A. Vissers and M. Diaz., Elsevier Science Publishers , 1989, pp. 303-326.

[BRI89] E. Brinksma, "A theory for the derivation of tests" , The Formal Description Techniques LOTOS, editors P.H.J. van Eijk, C.A. Vissers and M. Diaz. Elsevier Science Publishers 1989, pp.235-247.

[CHE88] T.Y. Cheung and Y. Zhu, "A Petri-net-based method for specifying distributed systems and deriving executable graphical LOTOS and ESTELLE", Tech. Report TR-88-04, Dept. of Computer Science, Univ. of Ottawa, (Feb. 1988).

[CHE89a] T.Y. Cheung, Y.C. Te, X. Ye and G.Q. Wang, "UO_GLOTOS: a syntax/system for representing, editing and translating graphical LOTOS", Proc. Second International Conference on Formal Description Techniques for Distributed systems and Communications Protocols, Vancouver, Canada, (1989), pp. 33-48.

[CHE89b] T.Y. Cheung, Y. Wu and X. Ye, "Generating test sequences and their degrees of indeterminism for distributed systems", Tech. Report TR-90-40, Dept. of Computer Science, Univ. of Ottawa, (Sept. 1990).

[CIN85] F.De Cindio, G.De Michelis, L. Pomello and C. Simone, "Exhibited-behavior equivalence and organizational abstraction in concurrent system design", Proc. Fifth International. Conf. on Distributed Comp. Systems, (1985), pp. 486-495.

[DEN84] R.De Nicola and M. Hennessy, "Testing equivalence for processes", Theoretical Computer Science 34, (1984), pp. 83-133.

[EHR85] H. Ehrig and B. Mahr, Fundamentals of Algebraic Specification1 : Equations and Initial Semantics, EATCS Monographs on Theoretical Computer Science, Springer Verlag, 1985.

[ELG89] H. Elgendy and R.L. Probert, "Equivalence of behavior expressions: A tutorial and comparison using LOTOS examples", Tech. Report TR-89-25, Dept. of Computer Science, Univ. of Ottawa, (July 1989).

[GAL90] S. Gallouzi and L. Logrippo, "An expressive trace theory for LOTOS", Tech. Report TR-90-29, Dept. of Computer Science, Univ. of Ottawa, (June 1990).

[HOA81] C.A.R. Hoare, S.D. Brookes, A.W. Roscoe, " A theory of communicating sequential processes", Tech. Report PRG-16, Oxford University Computing Labratory, Programming Research Group, Oxford, England, (May 1981).

[ISO7498] The Reference Model for Open System Interconnection, ISO International Standard IS7498.

[ISO8807] "Information processing system - Open System Interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behavior", ISO publication . 1988.

[KAN83] P.C. Kanellakis and S.A. Smolka, "CCS expression, finite state process, and three problems of equivalence", Proc. Second ACM Symposium on Principles of Distributed Computing, (1983), pp. 228-240.

[LAR86] K.G. Larsen, "Context-dependent bisimulation between Processes", Tech. Report CST-37-86, Dept. Computer Science, Univ. of Edinbourgh, (May 1986).

[LAN90] R. Langerak, "Decomposition of Functionality: a Correctness-Preserving LOTOS Transformation", Proc. Tenth IFIP WG 6.1 International Symposium on Protocol Specification, Testing and Verification, Ottawa, Canada, 1990. pp. 203-218.

[LED91] G.Leduc, "Conformance relation, associated equivalence, and new canonical tester in LOTOS", Proc. Eleventh International IFIP WG 6.1 Symposium on Protocol Specification, Testing and Verification, Stockholm, (1991), pp. 232-243.

[MAR89] S. Marchena and G. Leon, " Transformation from LOTOS specs to Galileo nets", Formal Description Techniques, K.J.Turner (editor), Elsevier Science Publishers, (1989), pp. 217-230.

[MIC87] Michael Hillerstrom, "Verification of CCS-process", M. Sc. Thesis in Computer Science, Aalborg University Center Institute of Electronic Systems, Jan, 1987.

[MIL80] Robin Milner, A Calculus of Communicating Systems, Lecture Notes of Computer Science, Vol. 92, Springer-Verlag, 1980.

[MIL90] R.E. Miller, "Protocol verification: The first ten years, the next ten years; Some personal observations", Proc. Tenth International IFIP WG 6.1 Symposium on Protocol Specification, Testing and Verification, Ottawa, Canada, (1990), pp. 201.

[PAI87] R. Paige and R.E. Tarjan, "Three pc_ution refinement algorithms", SIAM J. COMPUT, Vol. 16, No 6, (Dec 1987).

[PEH89] B. Pehrson, "Protocol verification for OSI", Sics. Tech. Report R89. Swedish Institute of Computer Science, Box 1263, S-164 28, Stockholm, Sweden.

[PET81] J.L. Peterson, Petri Net: Theory and the Modeling of System, Prentice-Hall, Inc., Englewood Cliffs. 1981.

[POM86] L. Pomello, "Some equivalence notions for concurrent systems: An overview", Advances in Petri Nets, Lecture Notes on Comp. Science, Vol. 222, Springer-Verlag, (1985), pp. 381-399.

[SHI89] N. Shiratori, H. Kaminaga, K. Takahashi, and S. Noguchi, "A verification method for LOTOS specification and its application", Proc. Ninth IFIP WG 6.1 International Symposium on Protocol Specification, Testing, and Verification, Enschede, The Netherlands, (1989).

[VIS88] C.A. Vissers, G. Scollo and M.V. Sinderen, "Architecture and specification style in formal descriptions of distributed systems", Protocol Specification, Testing and Verification VIII. S.Aggarwai and K.Sabnani (Editors), Elsevier Science Publishers B.V. (North-Holland). (1988), pp. 189-204.

[WEZ89] C.D. Wezeman, "The CO-OP method for compositional derivation of conformance testers", Proc. Ninth IFIP WG 6.1 International Symposium on Protocol Specification, Testing and Verification, Enschede, the Netherlands, (1989).