

Guided Symbolic Reachability using Partitioning

B. Tech Project Stage II Report

Submitted in partial fulfillment of the requirements
for the degree of

Bachelor of Technology

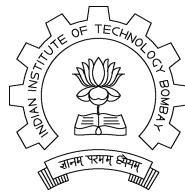
by

Varun Kanade

Roll No: 02005021

under the guidance of

Prof. Supratik Chakraborty



Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

Acknowledgments

I would like to thank my guide Prof. Supratik Chakraborty for his invaluable guidance, support and encouragement throughout the course of the work. The insights provided by him were very useful to understand in depth the topic concerned. I am grateful to Prof. Paritosh Pandya for providing timely help and support and making available useful references. My sincere thanks to Prof. S. Krishna for reading through the report and making valuable comments.

I would specially like to thank Dina Thomas, for her prompt replies and detailed explanations for even the most basic queries. Discussion and work with Sudeep Juvekar was very helpful in developing many of the ideas. Finally thanks to Barista! Many of the ideas germinated over the coffee there.

Varun Kanade
April 7, 2006

Abstract

Symbolic model checking is a technique used for the verification of finite state systems using BDDs. The state explosion problem refers to the very large growth in the number of states as the size of the system increases. Asynchronous systems consist of a set of transitions which are non-deterministically chosen and executed. The verification problem can be defined as determining whether a set of states is encountered starting from some set of initial states. The problem of verification thus involves finding the set of reachable states. The problem of large transition relations is solved by partitioning transition relations into clusters which depend on only few of the system variables. The theory of reachability expressions provides a theoretical framework for analyzing schedules in a system with partitioned transition relations. It also is helpful in proving the semantic equivalences between algorithms for reachability analysis. BDD size of the reachable states proves to be a bottleneck for BDD based model checking. Partitioning state space may overcome this problem. The theory of reachability matrices provides a method to allow transitions on a partitioned state space. The reachable states are now represented as a vector of BDDs whose union gives the set of reachable states. The effects and theory of partitioning state spaces are investigated in some detail.

Contents

Acknowledgments	i
Abstract	ii
Table of Contents	iii
List of Tables	iv
List of Figures	v
1 Introduction	1
2 Theory of Reachability Expressions	3
2.1 Syntax and Semantics of Reachability Expressions	3
2.2 Properties of Reachability Expressions	4
2.3 Some results of Reachability Expressions	5
2.4 Reachability Expressions and NuSMVDP	6
3 Experiments	7
3.1 Experiments with Fischer Processes	7
3.2 Experiments with Asynchronous Circuits	9
3.2.1 Layering the circuit topologically	11
3.2.2 Analyzing the Timed Transition Relation	12
4 Reachability Matrices	14
4.1 Some Definitions	14
4.2 Properties of Reachability Matrices	15
4.3 Some Results	16
5 Partitioning the Reachable States	21
5.1 Resizing State Vectors	21
5.2 Formulation 1	22
5.2.1 Some Results	22
5.2.2 Avoiding the Blow-up	24
5.3 Formulation 2	24
5.3.1 Some Results	25
6 Experiments with Reachability Matrices	27
6.1 Algorithm for Model Checking	27
7 Conclusions and Future Work	29

List of Tables

- 3.1 Performance comparison for Fischer process examples 8
- 3.2 More performance comparisons for Fischer process examples 8
- 3.3 Performance comparisons for circuit examples 11
- 3.4 More results on performance comparisons of circuit examples 12
- 3.5 Performance comparisons of circuit examples with layered clusters 12
- 3.6 More performance comparisons of circuit examples with layered clusters 13
- 3.7 Time required to generate models for ISCAS85 benchmarks 13

- 6.1 Performance comparison between partitioned and unpartitioned state space 28

List of Figures

- 3.1 Fischer Process Model 7
- 3.2 Model for gate in asynchronous circuits 9
- 3.3 Model for Inertial Delay 9
- 3.4 Model for bi-bounded pure delay 10

- 5.1 Diagram demonstrating application of formulation 1 24
- 5.2 Diagram demonstrating application of formulation 2 25

- 6.1 Algorithm for computing reachable state vector 27

Chapter 1

Introduction

Asynchronous systems consist of a set of processes which execute independently of each other. These systems can be modeled by network of timed automata[1]. The transitions of the various processes are carried out in an interleaved manner. At times a global transition is used to synchronize between the interacting processes. A transition of the system is a guarded action ($G \rightarrow A$). The guard G is a boolean combination of predicates on the state variables. Thus a transition can occur in a state only if the state satisfies the guard. An action A gives a valuation of the variables of the system, which gives the state that is reached after the transition occurs.

The reachability analysis of an asynchronous system involves choosing some enabled transition and executing it atomically, until no further transition can be applied. The safety verification problem reduces to searching the space of reachable states to detect if any undesirable state is reached by some execution. Model checking is the automatic verification of finite state concurrent systems. Symbolic model checking[4, 5] is a powerful technique for verification of very large systems. Symbolic model checking makes use of ordered binary decision diagrams[2](BDDs) to represent efficiently the set of states and also the transition relations. The sets of states and transition relations are expressed as boolean functions over the state variables.

The transition relation in the case of a network timed automaton can often be partitioned into transitions of the individual processes along with a global synchronization transition. The number of variables in the transitions of the individual processes is quite small, and it is efficient to use disjunctively partitioned transition relations as opposed to storing a single monolithic transition. The order in which these clusters of transitions are applied are observed to have significant impact on the size of the BDDs involved as well as the time required for reachability analysis. The theory of reachability expressions [7, 8] provides a framework to decide an optimal schedule for the transition clusters. Chapter 2 describes the theory of reachability expressions.

NuSMV is a tool developed for symbolic model checking[3]. NuSMVDP[6] is a tool developed at CFDVS, IIT Bombay to use reachability expressions to guide the reachability analysis. Along with the NuSMV code it also accepts a schedule for the transition clusters. A reachability expression gives a method of computation of final set of states from an initial set of states. Thus a reachability expression is regarded as a predicate transformer.

We demonstrate the effect of using various reachability expressions on some examples of timed systems. We verify Fischer's protocol for mutual exclusion. Our experiments suggest that use of reachability expressions significantly improves the efficiency of reachability analysis in several cases. Fischer's protocol is verified using classical techniques such as symbolic search using polyhedra or difference bound matrices for up to 20 processes. We could easily verify Fischer's protocol for 100 processes in reasonable time. Chapter 3 provides details of the experiments.

Asynchronous circuits serve as a very interesting set of examples for verification. We use gate level delay models for these circuits. The discrete transitions represent the transitions taking place at the wires between gates. The time transition synchronizes the clocks of all the gates. A detailed analysis of asynchronous circuit models is presented in chapter 3.

Symbolic model checking to some extent avoids the state-space explosion problem encountered in explicit state model checking. But our experience with very large asynchronous circuits shows that even with BDD representations, the BDDs representing the set of states grow very large. Thus we look at alternatives to get around this blow up in BDD size. We attempt to formalize a framework in which the set of states could themselves be partitioned in a way similar to the transition relations. This might overcome the problem of explosion in the size of the BDDs representing the set of states. Chapter 4 provides a theory of reachability matrices which generalize reachability expressions. Chapter 5 gives some methods to partition the state space. Chapter 6 shows some preliminary experiments based on reachability matrices. Chapter 7 concludes and points towards directions for future research.

Chapter 2

Theory of Reachability Expressions

In this chapter, we will introduce a notion of reachability expressions and study some of the algebraic properties of these. For a more complete set of results on reachability expressions refer to [7, 8]. We first define a state transition system.

Definition 2.1 State Transition System

A state transition system \mathcal{S} is defined as a quadruple $\mathcal{S} = (V, Q, Q_0, \gamma)$, where

- V is a finite set of variables
- Q is a set of states, each state is a particular assignment of variables in V to some specific value
- $Q_0 \subseteq Q$ is an initial set of states
- γ is a set of transitions of the form $(G \rightarrow A)$ where G is a guard which is a boolean combination of predicates over the state variables and A is an action giving a new assignment of variables

Each variable $v \in V$ has an associated domain D_v . A state in Q is thus an element of $\prod_{v \in V} D_v$. We also assume an additional property of the transition system, that for every state $q \in Q$ there is at least one transition $(G \rightarrow A) \in \gamma$ that can be applied to state q , i.e. q satisfies the guard G . The new state reached by application of this transition will have assignment of variables as defined in the corresponding action A . If some variable is not defined in A we assume it remains unchanged.

A cluster is defined as a non-empty subset $C \subseteq \gamma$. We define two special clusters as $\delta = (\text{True} \mapsto \text{skip})$ as the identity transition and Θ as the empty set of guarded action. An extended cluster of the transition system \mathcal{S} is defined to be either a subset of γ or δ or Θ .

An extended cluster C induces a transition relation on the set of states of \mathcal{S} denoted by R_C . We say that the pair $(q, q') \in R_C$ iff there exists a guarded action $(G \mapsto A) \in C$, such that the guard is enabled in q , and q' is the state obtained by applying the action A on q . Given an extended cluster C and a set of states $S \subseteq Q$, we define the *image* of S under C , denoted by $\text{Img}_C(S)$ to be the set $\{q' \in Q \mid \exists q \in S, R_C(q, q')\}$, i.e. the set of states reachable from S by one application of a guarded action in C .

2.1 Syntax and Semantics of Reachability Expressions

Let $\mathcal{T} = \{\mathbf{T}_1, \dots, \mathbf{T}_k\}$ be set of extended clusters of a state transition system \mathcal{S} . A *reachability expression* over \mathcal{T} is defined by the following grammar :

$$\begin{array}{lcl}
\text{RE} & \rightarrow & \text{RE} + \text{RE} \\
& & | \text{RE} \circ \text{RE} \\
& & | \text{RE}; \text{RE} \\
& & | * \text{RE} \\
& & | (\text{RE}) \\
& & | \mathbf{T}_1 | \dots | \mathbf{T}_k
\end{array}$$

Given a reachability expression σ , and a set $S \subseteq Q$ of states of \mathcal{S} , we want to define the notion of an *evaluation* of σ on S . This is done by taking images with various components of σ and applying some set operations on them. We will call the evaluation of σ on S as *image* of S under σ . We will now define the semantics of reachability expressions. The semantics are defined by the underlying transition system \mathcal{S} . The semantics of σ with respect to a state transition system $\mathcal{S} = (V, Q, Q_0, \gamma)$ is a mapping from 2^Q to 2^Q and is denoted by $\llbracket \sigma \rrbracket_{\mathcal{S}}$. If the context is clear, then we will just denote this by $\llbracket \sigma \rrbracket$. If S is a subset of Q , then $\llbracket \sigma \rrbracket$ can be inductively defined as follows :

- $\llbracket \mathbf{T}_i \rrbracket(S) = \text{Img}_{\mathbf{T}_i}(S)$, for all $\mathbf{T}_i \in \mathcal{T}$
- $\llbracket \sigma_1 + \sigma_2 \rrbracket(S) = \llbracket \sigma_1 \rrbracket(S) \cup \llbracket \sigma_2 \rrbracket(S)$
- $\llbracket \sigma_1 \circ \sigma_2 \rrbracket(S) = \llbracket \sigma_2 \rrbracket(\llbracket \sigma_1 \rrbracket(S))$
- $\llbracket \sigma_1; \sigma_2 \rrbracket(S) = \llbracket (\sigma_1 + \delta) \circ (\sigma_2 + \delta) \rrbracket(S)$
- $\llbracket (\sigma) \rrbracket(S) = \llbracket \sigma \rrbracket(S)$
- $\llbracket * \sigma \rrbracket(S) = \bigcup_{i=0}^{\infty} \llbracket (\sigma)^i \rrbracket(S)$

If σ_1 and σ_2 are two reachability expressions, we say that the expression σ_1 is covered by σ_2 ($\sigma_1 \sqsubseteq \sigma_2$) iff $\llbracket \sigma_1 \rrbracket_{\mathcal{S}}(S) \subseteq \llbracket \sigma_2 \rrbracket_{\mathcal{S}}(S)$ for all state transition systems \mathcal{S} , for all subsets S of Q and for all instantiations of symbolic clusters in \mathcal{T} with an extended cluster of \mathcal{S} . We say that $\sigma_1 = \sigma_2$ iff $\sigma_1 \sqsubseteq \sigma_2$ and $\sigma_2 \sqsubseteq \sigma_1$.

2.2 Properties of Reachability Expressions

It is worth to note here some algebraic properties of reachability expressions. These can be found in greater detail in [8] and so we omit all proofs here.

1. The set of reachability expressions form a commutative idempotent monoid with Θ as the identity element under $+$.
2. The set of reachability expressions form a monoid with δ as the identity element under \circ .
3. The set of reachability expressions form a semiring under the operations $+, \circ$ (distributive laws also hold in addition to the above two)

These algebraic properties are helpful in proving many results.

We now look at some properties related to the semantics of reachability expressions. From the definition of reachability expressions we can derive some straightforward properties. These are listed below without proof. The proofs can be found in [7, 8]

Property 2.1 If $\sigma_1 \sqsubseteq \sigma_2$ and $\sigma_3 \sqsubseteq \sigma_4$, then $(\sigma_1 \text{ op } \sigma_3) \sqsubseteq (\sigma_2 \text{ op } \sigma_4)$, where $\text{op} \in \{+, \circ, ;\}$

Property 2.2 $(\sigma_1; \sigma_2)^i \sqsubseteq (\sigma_1; \sigma_2)^{i+1}$ for all $i \geq 0$.

Property 2.3 If $\sigma_1 \sqsubseteq \sigma_2$, then $(\sigma_1)^i \sqsubseteq (\sigma_2)^i$ for all $i \geq 0$, and $(*\sigma_1) \sqsubseteq (*\sigma_2)$.

Property 2.4 $(*\sigma_1)^i = (*\sigma_1)$ for all $i \geq 0$, and $*(*\sigma_1) = (*\sigma_1)$.

Property 2.5 For all σ_1, σ_2 , $*(\sigma_1; \sigma_2) = *(\sigma_1; (*\sigma_2)) = *((*\sigma_1); \sigma_2) = *((*\sigma_1); (*\sigma_2))$.

2.3 Some results of Reachability Expressions

In this section, we will state some theorems related to reachability expressions. The proofs of these can be found in [7, 8]. The theorems given here provide some guidance in choosing an appropriate reachability expressions for reachability analysis.

Lemma 2.1 *Let $\mathcal{T} = \{\mathbf{T}_1, \dots, \mathbf{T}_k\}$ be the set of transition clusters of a transition system \mathcal{S} . Then*

1. $\ast(\mathbf{T}_1 + \dots + \mathbf{T}_k)$ computes the reachable states of \mathcal{S} .
2. For any reachability expression σ over \mathcal{T} , $\sigma \sqsubseteq \ast(\mathbf{T}_1 + \dots + \mathbf{T}_k)$

The first theorem states that two particular ways of computing the set of states is equivalent, thus they both compute exactly the same set of reachable states.

Theorem 2.1 *Let $\{\sigma_1, \dots, \sigma_k\}$, $k \geq 1$, be a finite set of reachability expressions. Then $\ast(\sigma_1 + \dots + \sigma_k) = \ast(\sigma_1; \dots; \sigma_k)$.*

The next theorem is an extension of the above theorem. The “ \ast ” denotes taking fixed points of certain operators. The next theorem gives the relation between two reachability expressions, while iteratively computing the fixed-points above.

Theorem 2.2 *Let $\{\sigma_1, \dots, \sigma_k\}$, $k \geq 2$, be a finite set of reachability expressions. Let $\sigma_Y = (\sigma_1 + \dots + \sigma_k)$ and $\sigma_Z = (\sigma_1; \dots; \sigma_k)$. Then the following hold :*

1. For all $n \geq 0$, $(\sum_{j=0}^n (\sigma_Y)^j) \sqsubseteq (\sigma_Z)^n$.
2. Furthermore, if $(\sigma_p)^2 \sqsubseteq \sigma_p$ for all p , then $(\sum_{j=0}^n (\sigma_Y)^j) \sqsubseteq (\sigma_Z)^{n - (\lfloor \frac{n}{k} \rfloor - 1)}$.

This theorem states what n iterations of the reachability expression $(\sigma_1 + \dots + \sigma_k)$ can compute, the reachability expression $(\sigma_1; \dots; \sigma_k)$ can do in n or less. The fixed-point is thus obtained faster in the second case. Under certain conditions, this bound is improved further. The condition $(\sigma_p)^2 \sqsubseteq \sigma_p$ may occur under several conditions. For example if $\sigma_p = \ast\widehat{\sigma}_p$, or if the variables assigned values in the actions of σ_p do not intersect with the set of variables in the guard of σ_p . This theorem suggests that if the complexity of one iteration of $(\sigma_1 + \dots + \sigma_k)$ is comparable to the complexity of one iteration of $(\sigma_1; \dots; \sigma_k)$, then it may be advantageous to use the latter expression, in the reachability analysis. This fact may be reasonable to expect, since the number of image computations and unions involved is same in both the cases.

The next theorem is useful in minimizing the number of image computations with respect to one very large cluster. A time transition cluster is typically found to be much larger than all other clusters. So minimizing the number of applications of such a cluster may reduce the time required for reachability analysis.

Theorem 2.3 *Let $\mathcal{S} = (V, Q, Q_0, \gamma)$ be a finite state transition system with extended clusters $\{\mathbf{T}_1, \dots, \mathbf{T}_k\}$, such that $\bigcup_{i=1}^k \mathbf{T}_i = \gamma$ and $\mathbf{T}_k \not\sqsubseteq \ast(\mathbf{T}_1 + \dots + \mathbf{T}_{k-1})$. Let $\sigma_X = (\mathbf{T}_1 + \dots + \mathbf{T}_{k-1})$, and $\widehat{\sigma} = (\ast\sigma_X) \circ \ast(\mathbf{T}_k; (\ast\sigma_X))$. Then the following hold*

1. $\llbracket \widehat{\sigma} \rrbracket(Q_0) = \text{reach}(\mathcal{S})$
2. For every $S \subseteq Q$ and reachability expression σ over $\{\mathbf{T}_1, \dots, \mathbf{T}_k\}$, let $N_k(\sigma, S)$ denote the number of times image under \mathbf{T}_k is computed during the evaluation of $\llbracket \sigma \rrbracket(S)$. If $\llbracket \sigma \rrbracket(Q_0) = \text{reach}(\mathcal{S})$, then $N_k(\widehat{\sigma}, Q_0) \leq N_k(\sigma, Q_0) + 1$.

We see that if we have a very large cluster and other clusters much smaller, it may be advantageous to use a reachability expression as given in the above theorem. This is because image computation depends directly on the size of the BDD. However this may result in excessive number of image computations with respect to other clusters. If this happens, then the advantage of minimizing the number of image computations under the largest cluster, may be lost. Hence the theorem should be used with some care. This is further discussed in chapter 3.

The next theorem tries to define some kind of an ordering in the clusters. Consider a case where the image computation under a cluster σ_i does not lead to any new states, until image computation has been done with respect to some other cluster σ_j . This kind of an ordering occurs naturally in circuits. The theorem states that some notion of an ordering can be useful in simplifying reachability expressions used to perform reachability analysis.

Theorem 2.4 *Let $\{\sigma_1, \dots, \sigma_k\}$ be a set of extended clusters and S be a set of states satisfying the following conditions:*

1. $(\sigma_i \circ ((\ast\sigma_{i+1}) \circ \dots \circ (\ast\sigma_k)) \circ \sigma_i) \sqsubseteq (\sigma_i \circ ((\ast\sigma_i) \circ \dots \circ (\ast\sigma_k)))$ for all $1 \leq i < k$.
2. There exists m , $1 \leq m \leq k$ such that
 - (a) $(\sigma_i \circ \sigma_j) \sqsubseteq (\sigma_i + \sigma_j)$ for all $1 \leq i, j \leq m$.
 - (b) $\llbracket \ast\sigma_i \rrbracket(S) = S$, for all $i > m$.

Then $\llbracket \ast(\sigma_1; \dots; \sigma_k) \rrbracket(S) = \llbracket (\ast\sigma_1); \dots; (\ast\sigma_k) \rrbracket(S)$.

The condition 1 in the above theorem formalizes the notion of ordering of dependencies between the clusters. In short it states that the effect of computing the image under expressions $\{\sigma_{i+1}, \dots, \sigma_k\}$ does not affect the computation of image under σ_i for all $1 \leq i < k$. The condition 2(a) asserts that the first few expressions do not depend on any other expression, while condition 2(b) states that unless the image of S is computed first under one of $\{\sigma_1, \dots, \sigma_m\}$, no new states are reached. This is an additional ordering requirement. Under these conditions the theorem allows the simplification of reachability expressions.

2.4 Reachability Expressions and NuSMVDP

NuSMVDP[6] is a tool developed at CFDVS, IIT Bombay. It hacks up NuSMV to allow support for reachability expressions. Details of the NuSMVDP may be found in [6]. The operators of NuSMVDP are not exactly reachability expressions. The operators used are `fixU`, `fix`, `img` and `union`. But a simple syntax translation scheme can be used to convert reachability expressions to this form.¹

RE	→	(RE ₁ + RE ₂)	{ RE.code = union(RE ₁ .code, RE ₂ .code); }
		(RE ₁ ∘ RE ₂)	{ RE.code = fix(fix(\$, RE ₁ .code, 1), RE ₂ .code, 1); }
		(RE ₁ ; RE ₂)	{ RE.code = fixU(fixU(\$, RE ₁ .code, 1), RE ₂ .code, 1); }
		∗RE ₁	{ RE.code = fixU(\$, RE ₁ .code, inf); }
		(RE ₁)	{ RE.code = RE ₁ .code }
		T _i	{ RE.code = img(\$, i) }

¹More details of this scheme are provided in the first stage report.

Chapter 3

Experiments

This chapter presents experiments with timed systems. We use different schedules suggested by reachability expressions to compare their performance. We use automaton models for Fischer's process and asynchronous circuits for verification.

3.1 Experiments with Fischer Processes

Fischer's Protocol is used to ensure mutual exclusion when a number of processes access a shared resource. Each process can be modeled using a timed automaton as shown in Fig. 3.1. The variable ck_i is the clock variable, and k is a variable shared across the processes. $k = 0$ indicates that the shared resource is free and $k = i$ indicates that the i^{th} process is using the shared resource. The integers a and b are constants, which in the experiments we set to 1 and 2 respectively. The transition clusters used were one for each individual process and a global time transition cluster.

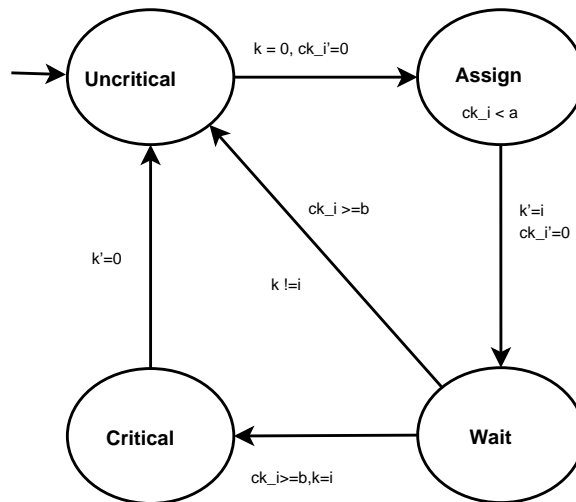


Figure 3.1: Fischer Process Model

Experimental Setup

In the case of Fischer's Protocol with n processes, let $\{\mathbf{T}_1, \dots, \mathbf{T}_n\}$ denote the set of clusters representing discrete transitions, one corresponding to each process. Let \mathbf{T}_t denote the global time transition cluster which synchronizes the clocks of all the processes and represents advancement of time. We may combine all the clusters together giving a monolithic transition cluster Γ . The reachability expression

$$S_0 = * \Gamma$$

mimics the monolithic transition relation as in the original NuSMV tool. If we use a disjunctively partitioned transition relation, we may represent this by a reachability expression.

$$S_1 = *(\mathbf{T}_1 + \dots + \mathbf{T}_n + \mathbf{T}_t)$$

The BDD size of the individual clusters is much smaller if we use the reachability expression S_1 in

Num	Max	\mathbf{T}_t	S_1			S_2			S_3		
			Time(s)	Itr	BDD	Time(S)	Itr	BDD	Time(S)	Itr	BDD
10	80	186	0.49	15	2318	0.14	7	1455	0.14	5	1455
20	91	261	19.13	25	17178	1.17	7	6715	1.02	5	6715
30	93	531	155.51	35	55663	4.31	7	15775	4.19	5	15775
40	102	706	698.12	45	129023	10.23	7	28635	10.65	5	28635
50	106	876	>30m	-	-	20.65	7	45295	21.671	5	45295
70	119	1221	>30m	-	-	62	7	90015	63.34	5	90015
100	117	1731	>30m	-	-	189.5	7	185595	194.12	5	185595

Table 3.1: Performance comparison for Fischer process examples

NuSMVDP. We see that this is essentially like the disjunctively partitioned transition relation as discussed in [4]. Theorem 2.2 shows that the reachability expression

$$S_2 = *(\mathbf{T}_1; \dots; \mathbf{T}_n; \mathbf{T}_t)$$

computes the same set of states as S_1 and will require lesser number of iterations under certain conditions.

Let $\sigma_x = (\mathbf{T}_1; \dots; \mathbf{T}_n)$ and define the reachability expression S_3 as

$$S_3 = (*\sigma_x) \circ *(\mathbf{T}_t; (*\sigma_x))$$

Also define a reachability expression S_4 as

$$S_4 = *(*\mathbf{T}_1; \dots; *\mathbf{T}_n; \mathbf{T}_t)$$

We evaluate the performance of the following reachability expressions in the case of Fischer's protocol. The results are displayed in tables 3.1 and 3.2. The experiments in this section were performed on a 3 GHz Intel 686 processor with 1 GB of main memory and running Fedora Core Linux 3.4.3-6.fc3.

Num	S_4			S_0			
	Time(s)	Itr	BDD	Γ	Time(s)	Itr	BDD
10	0.17	5	1932	3154	0.25	15	2592
20	1.42	5	8152	10499	7.71	25	17805
30	4.87	5	18372	21944	60.64	35	56980
40	13.03	5	2592	37489	229.51	45	131280
50	27.54	5	50812	57134	603.32	55	251955
70	88.48	5	99252	108724	>30m	-	-
100	282.87	5	201912	216859	>30m	-	-

Table 3.2: More performance comparisons for Fischer process examples

Tables 3.1 and 3.2 clearly show that unguided disjunctive partitioning as in S_1 is worse than even the monolithic transition relation S_0 . The number of iterations is same in each of the cases, however each iteration involves many more number of image computations in S_1 than in S_0 . Thus, in absence of guidance, partitioning is not an effective strategy. From the values under S_2 , S_3 and S_4 , we see that guidance does give significant improvement in performance. It is clear that S_2 performs better than S_1 in all the cases. This is what we expect from theorem 2.2. If we compare the reachability expressions S_2 and S_3 , we see that the number of iterations¹ decreases in the case of S_3 . An interesting observation is that the number of iterations in S_2 , S_3 and S_4 is actually independent of the number of processes. This is because the operator “;” ensures that all states reachable due to the discrete transitions are obtained in one pass over $\mathbf{T}_1, \dots, \mathbf{T}_n$, this however does not happen in S_1 .

¹Here the iterations are counted with respect to the outermost *

3.2 Experiments with Asynchronous Circuits

An asynchronous circuit is modeled by using timed automata as model for individual gates. The gates are modeled to have an inertial delay and a bi-bounded pure delay. The gate is modeled in three parts as follows (See figure 3.2).

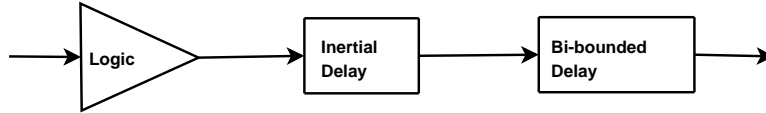


Figure 3.2: Model for gate in asynchronous circuits

- A *boolean logic block* that gives the boolean value of the output as a function of the boolean value of its inputs without introducing any delay.
- An *inertial delay element* which takes as input the output of the boolean logic block. Let D be the inertial delay of the element, then the output of this changes only if the input persists for at least D units of time

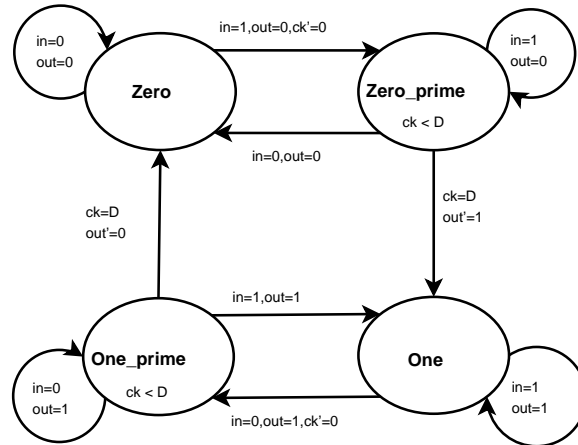


Figure 3.3: Model for Inertial Delay

- A *bi-bounded pure delay element* which takes as input, the output of the inertial delay element. Let l and u be the lower and upper bounds of the bi-bounded delay element, then the element delays each transition on its input non-deterministically by an amount t which is between l and u .

The inertial delay element with inertial delay D and a bi-bounded pure delay element with bounds l and u can be modeled by timed automata. The models are shown in figures 3.3 and 3.4. In these figures the states **Zero** and **One** are the stable states.

We model a given circuit as a network of such timed automata. To simplify the model, we assume the same values for the parameters D , l and u for each of the gates. We also force the condition $u < D$ which ensures that a change in the input is certainly visible at the output. When an output of a gate is connected to an input of another, we ensure that the transitions on these occur simultaneously. Also it is assumed that time flows synchronously for all gates. This is enforced by the time transition cluster.

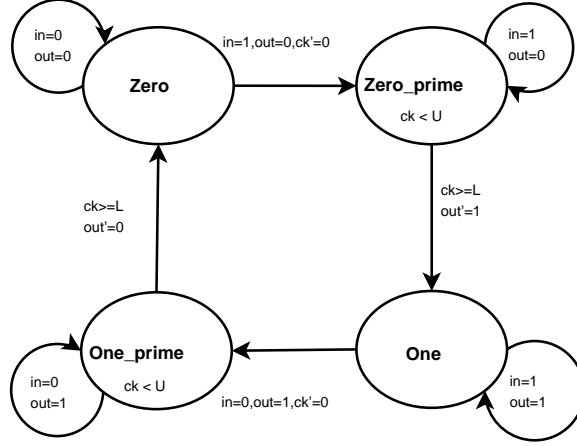


Figure 3.4: Model for bi-bounded pure delay

Experimental Setup

Let the discrete clusters of a circuit be $\{\mathbf{T}_1, \dots, \mathbf{T}_k\}$ and let \mathbf{T}_t be the time transition cluster. As in the case of the Fischer protocol, let Γ denote the monolithic transition cluster. Then the reachability expression

$$S_0 = *\Gamma$$

mimics the symbolic breadth-first search using a monolithic transition relation. Further let $\mathbf{T}_1, \dots, \mathbf{T}_k$ be topologically ordered in the circuit. Let $\sigma_x = (\mathbf{T}_1; \dots; \mathbf{T}_k)$, $\sigma_y = (\mathbf{T}_k; \dots; \mathbf{T}_1)$ and $\sigma_z = (*\mathbf{T}_1; \dots; *\mathbf{T}_k)$. Define reachability expressions as follows

$$\begin{aligned} S_1 &= *(\mathbf{T}_1 + \dots + \mathbf{T}_k + \mathbf{T}_t) \\ S_2 &= *(\mathbf{T}_1; \dots; \mathbf{T}_k; \mathbf{T}_t) \\ S_3 &= (*\sigma_x) \circ *(\mathbf{T}_t; (*\sigma_x)) \\ S_4 &= (*\sigma_y) \circ *(\mathbf{T}_t; (*\sigma_y)) \\ S_5 &= \sigma_z \circ *(\mathbf{T}_t; \sigma_z) \\ S_6 &= *(*\mathbf{T}_1; \dots; *\mathbf{T}_k; \mathbf{T}_t) \end{aligned}$$

We evaluate the relative performance of these reachability expressions on some circuits. These are tabulated in tables 3.3 and 3.4. The experiments in this section were performed on a 3 GHz Intel 686 processor with 1 GB of main memory and running Fedora Core Linux 3.4.3-6.fc3.

In the case of asynchronous circuits, we see that the size of the BDD representing the cluster \mathbf{T}_t is much larger than those representing the discrete transitions (\mathbf{T}_i). This is because the cluster \mathbf{T}_t involves clock variables of all the gates, whereas for the discrete transition clusters \mathbf{T}_i , the cluster involves variables of very few (typically 1 or 2) gates. This suggests that it is important to reduce the number of applications of the cluster \mathbf{T}_t in the reachability analysis. Theorem 2.3 suggests a mechanism to minimize the number of image computations due to cluster \mathbf{T}_t . Clearly S_2 and S_3 outperform S_1 as seen in the table. So also S_4 and S_5 outperform S_1 . We compare then the results between S_2 and S_3 . We observe that the time required using S_3 is comparable and slightly larger than S_2 . The number of applications of the cluster \mathbf{T}_t in S_3 is just one less than in S_2 . The number of applications of other clusters may actually be more in S_3 due to internal fixed points compensating this. Thus we see that theorem 2.3 should be used with caution.

Theorem 2.4 relates to the effect of applying a sequence of clusters consistently with topological dependencies between them. The effect of this is observed in S_5 . We set the initial state of the circuit to be stable, so that the theorem is applicable to the discrete transition clusters. Thus we get using the

Ckt	Δ_{in}	D	(l, u)	Max	$ \mathbf{T}_t $	S_1			S_2			S_3		
						$ T_i $	Time(s)	Itr	BDD	Time(s)	Itr	BDD	Time(s)	Itr
2	10	8	(3,6)	101	40965	6.48	132	30615	2.9	95	27707	4.1	94	27707
2	12	10	(3,8)	101	43413	55.81	155	75819	24.37	117	69919	35.08	116	69919
3	7	6	(3,5)	262	125864	20.98	135	49335	7.99	88	37122	11.25	87	37122
3	10	9	(5,8)	262	139111	80.87	178	159305	35.12	130	108833	51.61	129	108833
3	12	10	(5,8)	262	139927	48.53	198	80135	24.6	148	61526	36.46	147	61526
4	5	4	(2,4)	171	29076	49.14	104	51397	10.87	61	47364	14.09	59	47364
4	7	6	(3,6)	173	30096	274.54	157	200497	153.63	114	151418	265.19	112	151413
4	10	9	(4,8)	175	33664	640.15	229	347057	406.4	181	328447	739.16	179	328261
6	5	4	(2,4)	260	137058	42.82	87	25941	15.86	48	17111	24.85	47	17111
6	7	6	(3,5)	262	140968	54.42	107	30629	25.37	68	22066	42.35	67	22066
6	10	9	(5,8)	262	157132	123.38	140	50207	71.7	101	39351	126.68	99	39351
7	5	4	(2,4)	260	135917	73.18	107	24171	16.87	55	14469	19.13	54	14469
7	7	6	(3,5)	262	139881	75.71	129	23379	16.94	78	15372	25.65	77	15372
7	10	9	(5,8)	262	156-21	175.74	165	51623	73.3	115	34208	131.3	114	34208
7	12	10	(5,8)	262	157213	145.41	182	37294	70.7	132	27514	123.67	131	27514
8	5	4	(2,4)	292	147303	155.07	117	28012	29.03	55	16431	37.02	54	16431
8	7	6	(3,5)	294	151267	154.78	139	26668	26.95	78	17488	36.773	77	17488
8	10	9	(5,8)	306	167407	389.62	175	62502	132.63	115	38856	243.28	114	38856
8	12	10	(5,8)	306	168599	313.95	189	45077	124.74	132	31194	221.45	131	31194

Table 3.3: Performance comparisons for circuit examples

theorem that $*(\sigma_x) = \sigma_z$. We also have that $*\sigma_x = *\sigma_y$. The reachability expression S_4 does computation using reverse topological dependencies. The time required in this case is more than the time required in S_5 as the table shows. Also in most cases S_3 outperforms S_4 as expected. S_5 also performs better than S_3 since the fixed points are taken with individual clusters here rather than σ_x .

The monolithic transition relation does not seem to do much worse than the other schedules in case of circuits. One reason for this may be that the size of the time transition cluster \mathbf{T}_t in all the reachability expressions is comparable to the size of the monolithic transition cluster. In the case of circuits 6, 7 and 8 the monolithic transition relations outperforms all others. This may be due to the special structure of these circuits. These circuits have a sequence of cascaded AND gates with no delay. A partial explanation may be that scheduling these cascaded AND gates actually require some time. By clubbing such AND gates together in a cluster it may be possible to reduce the time required.

3.2.1 Layering the circuit topologically

In this section we investigate creating clusters using layers in a topological ordering. When a DAG is topologically ordered, we can assign a layer to each vertex, such that no vertex with layer number l has a path from any vertex with layer number larger than l leading into it. We create such layers in a circuit, and make clusters such that if two transitions occur in the same layer, they are clubbed together in one layer. This way we can reduce the number of clusters. Theorem 2.2 suggests that reducing the number of clusters may reduce the number of iterations in computing the fixed-points. Also the way we have defined the layers of clusters. The topological dependency of the clusters is also maintained. We performed experiments with the schedules S_2 , S_5 and S_6 . The results are in tables 3.5 and 3.6. These experiments were performed on an HP Notebook with Intel Pentium M 1.60 GHz processor with 512 MB of main memory running Windows XP (with cygwin).

From tables 3.5 and 3.6 we can see that there is not much improvement obtained by clubbing together clusters of transitions in the same layer of a topological order on a circuit together. Using S_2 we see that actually the number of iterations is more than in the previous case. One possible explanation for this is the following. Consider two clusters \mathbf{T}_i and \mathbf{T}_j in the same layer of a topological order. Now consider the reachability expressions $\mathbf{T}_i + \mathbf{T}_j$ and $\mathbf{T}_i; \mathbf{T}_j$. We know that $\mathbf{T}_i + \mathbf{T}_j \subseteq \mathbf{T}_i; \mathbf{T}_j$. Let $\mathbf{T}_1, \dots, \mathbf{T}_k$ be in

Ckt	Δ_{in}	D	(l, u)	S_4			S_5			S_0			
				Time(s)	Itr	BDD	Time(s)	Itr	BDD	$ \Gamma $	Time(s)	Itr	BDD
2	10	8	(3,6)	4.02	94	27707	3.31	94	27707	44068	4.2	132	41255
2	12	10	(3,8)	32.41	116	69919	26.82	116	69919	46514	36.18	155	99501
3	7	6	(3,5)	12.1	87	37181	8.53	87	37122	129948	17.75	135	73603
3	10	9	(5,8)	52.15	129	108833	37.58	129	108833	143217	74.8	178	221807
3	12	10	(5,8)	37.92	147	61526	23.85	147	61526	144033	41.15	198	106946
4	5	4	(2,4)	13.85	59	47342	11.19	60	47364	31265	37.15	104	101201
4	7	6	(3,6)	271.18	112	151525	158.91	113	15148	32671	224.78	157	339490
4	10	9	(4,8)	737.63	179	328261	429.23	180	328447	36218	560.32	229	512018
6	5	4	(2,4)	41.09	47	18060	15.54	47	17111	140886	5.42	97	22405
6	7	6	(3,5)	57.85	67	23195	24.91	67	22066	144874	5.8	121	21676
6	10	9	(5,8)	174.48	99	39848	69.4	100	39351	161031	11.11	159	46544
7	5	4	(2,4)	53.35	54	14555	14.91	54	14469	141961	8.16	107	24171
7	7	6	(3,5)	54.5	77	15464	16.79	77	15372	154949	8.48	129	23379
7	10	9	(5,8)	451.31	114	34080	71.08	114	34208	162106	14.19	165	51623
7	12	10	(5,8)	371.96	131	27668	68.44	131	27514	163302	13.18	182	37294
8	5	4	(2,4)	140.63	54	16543	26.9	54	16431	154815	12.56	117	28012
8	7	6	(3,5)	132.28	77	17606	26.6	77	17488	158803	12.78	139	26688
8	10	9	(5,8)	1163.08	114	38753	128.23	114	38856	174987	21.17	175	62502
8	12	10	(5,8)	981.35	131	31367	120.8	131	31194	176183	19.24	189	45077

Table 3.4: More results on performance comparisons of circuit examples

Ckt	Δ_{in}	D	(l, u)	$ \mathbf{T}_t $	Layered Clusters (S_2)				Single Clusters (S_2)			
					Max $ \mathbf{T}_i $	Time(s)	Itr	BDD	Max $ \mathbf{T}_i $	Time(s)	Itr	BDD
2	4	3	(1,2)	6147	521	0.54	56	2140	138	0.6	39	2143
2	10	8	(3,6)	7895	536	4.54	128	11517	141	2.07	104	8380
2	12	10	(3,8)	8155	536	11.93	150	25206	141	4.99	127	16193
3	4	3	(1,2)	9447	732	0.62	44	2697	251	0.59	31	2526
3	7	6	(3,5)	10972	751	4.96	105	12300	262	3.25	81	10721
3	10	9	(5,8)	12078	751	14.93	145	26481	262	10.31	120	23671
3	12	10	(5,8)	12226	751	6.94	130	20787	262	4.69	111	18672

Table 3.5: Performance comparisons of circuit examples with layered clusters

topological order, such that $\{\mathbf{T}_1, \dots, \mathbf{T}_{k_1}\}, \dots, \{\mathbf{T}_{k_{l-1}+1}, \dots, \mathbf{T}_k\}$ are clusters in layers 1 to l . Now we see that $((\mathbf{T}_1 + \dots + \mathbf{T}_{k_1}); \dots; (\mathbf{T}_{k_{l-1}+1} + \dots + \mathbf{T}_k); \mathbf{T}_t)^n \sqsubseteq (\mathbf{T}_1; \dots; \mathbf{T}_k; \mathbf{T}_t)^n$ holds for all n . This explains the facts that there may be actually more number of iterations if we use S_2 with clubbing clusters in the same layer together. When S_5 or S_6 is used, then the fixed-points with respect to individual clusters make sure that the number of iterations are exactly the same in both the cases. We can show that $(*(\mathbf{T}_1 + \dots + \mathbf{T}_{k_1}); \dots; *(\mathbf{T}_{k_{l-1}+1} + \dots + \mathbf{T}_k)) = (*\mathbf{T}_1; \dots; *\mathbf{T}_k)$. Thus the number of iterations has to be the same in both the cases. The trade off between reducing the number of clusters and the increased size of the individual clusters seem to cancel off and we see that S_5 and S_6 give almost identical results in both the cases.

3.2.2 Analyzing the Timed Transition Relation

We see that the timed transition relation \mathbf{T}_t is quite large for even moderate sized circuits of about 20 gates. This proves to be a potential block in getting the verification through for somewhat larger circuits. Our experiments show that for a 74181 adder, if we define the timed transition \mathbf{T}_t as a single cluster synchronously advancing all the clocks, NuSMVDP freezes. It is thus necessary to split the timed transition itself into smaller transitions which are of moderate size. The synchronization action is ensured

Ckt	Δ_{in}	D	(l, u)	Layered Clusters (S_6)			Single Clusters (S_6)			Layered Clusters (S_7)			Single Clusters (S_7)		
				Time(s)	Itr	BDD	Time(s)	Itr	BDD	Time(s)	Itr	BDD	Time(s)	Itr	BDD
2	4	3	(1,2)	0.6	38	2143	0.56	38	2143	0.59	39	2143	0.56	39	2143
2	10	8	(3,6)	2.95	103	8380	2.47	103	8380	2.17	104	8380	2.52	104	8380
2	12	10	(3,8)	6.8	126	16193	6.46	126	16193	6.59	127	16193	6.5	127	16193
3	4	3	(1,2)	0.65	30	2526	0.67	30	2526	0.6	31	2526	0.67	31	2526
3	7	6	(3,5)	3.75	80	10721	3.89	80	10721	3.68	81	10721	3.84	81	10721
3	10	9	(5,8)	12.28	119	23671	11.97	119	23671	12.23	120	23671	12.1	120	23671
3	12	10	(5,8)	5.56	110	18672	5.55	110	18672	5.66	111	18672	5.66	111	18672

Table 3.6: More performance comparisons of circuit examples with layered clusters

Benchmark	Time(s)
c432	11.15
c499	18.24
c880	74.75
c1355	175.17
c1908	∞

Table 3.7: Time required to generate models for ISCAS85 benchmarks

We use partitioned a time transition relation as opposed to a single time transition cluster to generate these models. The larger models not shown in the table did not get generated in reasonable time

by making a sequential composition of the timed transitions for all the gates.

From figure 3.2 we see that a gate is modeled by feeding the output of an inertial delay model to the input of a bi-bounded pure delay. By examining the two models (fig. 3.3, 3.4) we see that time transition only affects the clock variables, state variables and the shared variable between the automata, i.e. the output of the inertial delay and input of the bi-bounded delay. The time transition of any gate thus can be represented as a cluster with very few variables. However we require that all clocks should advance simultaneously. For this we perform a sequential composition across all the gates.

Let $\mathbf{Rt}_1, \dots, \mathbf{Rt}_k$ be the time transition clusters of the gates in the circuit. Then let \mathbf{T}_t be defined as

$$\mathbf{T}_t = \mathbf{Rt}_1 \circ \dots \circ \mathbf{Rt}_k$$

Also consider the relation

$$\mathbf{T}_t = \mathbf{Rt}_1 \wedge \dots \wedge \mathbf{Rt}_k$$

We can see that these will be the same, since the each of the clusters \mathbf{Rt}_i have disjoint sets of variables. We see that if the global transition is not applicable at some time, it will be because at least one of the gates is waiting for an urgent discrete transition. In that case the sequential composition of the clusters at that gate will return a null set. Otherwise, both the expressions will advance time by a single unit. This allows for much smaller time-transition clusters. For example the model for 74181 is easily generated on NuSMV using this kind of partitioning of a time cluster. Table 3.2.2 shows the time required to generate models for the ISCAS85 benchmarks. The experiments were performed on an IBM ThinkPad with Intel Pentium M 1.86 GHz processor with 512 MB of main memory running Windows XP (with cygwin).

Despite partitioning the time transition relation we were not able to make much progress with the 74181 adder. Although it did begin executing after building the model, the frontier sets grew too large making it impossible for NuSMVDP to perform reachability computation. We thus need mechanisms to reduce the size of the BDD representing the reachable states. The remaining part of the report deals with techniques to help bringing this about.

Chapter 4

Reachability Matrices

In this chapter we introduce the concept of reachability matrices. Reachability matrices allow using vectors of sets of states rather than a single set of states. We may then represent reachable states in a partitioned manner similar to the transition relations. We introduce the semantic interpretation of reachability matrices and also discuss some of the algebraic properties. The properties of the reachability matrices are very similar to most of the properties of reachability expressions discussed in chapter 2. We begin by making a few preliminary definitions.

4.1 Some Definitions

Definition 4.1 State Vector Let $\bar{S} = (S_1, \dots, S_k)$ be a vector of length k , where each of S_i is a set of states. Then we call \bar{S} a state vector.

Definition 4.2 Norm If $\bar{S} = (S_1, \dots, S_k)$ is a state vector, we define the norm of the vector \bar{S} , represented by $\|\bar{S}\|$ to be $S_1 \cup \dots \cup S_k$. We will usually denote $\|\bar{S}\|$ by S .

We also allow operations such as union (\cup) and intersection (\cap) on state vectors, where the operations are carried out element-wise.

In chapter 2 we studied the concept of reachability expressions as a predicate transformer, i.e. takes a set of states and returns a set of states. We can define appropriate reachability expressions to obtain the fixed point under the transitions of the system, which gives the set of reachable states. A natural extension to this idea is that of a matrix of reachability expressions, which we will call a *reachability matrix*, to act upon a state vector and return another state vector. We would then want to compute fixed points of state vectors with respect to certain reachability matrices to obtain the state vector representing the set of reachable states, i.e. its norm is the set of reachable states.

Definition 4.3 Reachability Matrix A reachability matrix \mathbf{R} of size $k \times k$ is a matrix with k^2 elements, each of which is a reachability expression.

We have seen in chapter 2 that the set of reachability expressions under the operators $+$, \circ form a semiring. These can be used to define operators on reachability matrices. The next few sections will discuss algebraic properties of reachability matrices.

We now define how a reachability matrix acts upon a state vector. We apply the reachability expressions in the matrix column-wise to the state vector and aggregate each of these to get an element of the new vector. The formal definition is given below. Let \mathbf{R} be a reachability matrix and \bar{S} and \bar{S}' the initial and final state vectors respectively. We will denote the application of reachability matrix on a

state vector by $\llbracket \mathbf{R} \rrbracket(\bar{S})$.¹

$$\left[\left(\begin{array}{ccc} r_{11} & \cdots & r_{1k} \\ \vdots & \ddots & \vdots \\ r_{k1} & \cdots & r_{kk} \end{array} \right) \right] (S_1, \dots, S_k) = (S'_1, \dots, S'_k) \quad (4.1)$$

The following is the relation between \bar{S}' and \bar{S} .

$$S'_i = \llbracket r_{1i} \rrbracket(S_1) \cup \cdots \cup \llbracket r_{ki} \rrbracket(S_k)$$

We now discuss the algebra of reachability matrices.

4.2 Properties of Reachability Matrices

We define some operators on reachability matrices and their semantics. The discussion below assumes that all reachability matrices are of the size $k \times k$.

- Define Φ to be the additive identity matrix where each element $\Phi_{ij} = \Theta$.
- Define Δ to be the multiplicative identity matrix where each element $\Delta_{ij} = \delta$, if $i = j$ and $\Delta_{ij} = \Theta$ otherwise.
- If \mathbf{R}, \mathbf{Q} are two reachability matrices, then define $\mathbf{P} = \mathbf{R} + \mathbf{Q}$ to be the matrix, where $\mathbf{P}_{ij} = \mathbf{R}_{ij} + \mathbf{Q}_{ij}$
- If \mathbf{R}, \mathbf{Q} are two reachability matrices, then define $\mathbf{P} = \mathbf{R} \circ \mathbf{Q}$ to be the matrix where $\mathbf{P}_{ij} = +_{l=1}^k \mathbf{R}_{il} \circ \mathbf{Q}_{lj}$.
- If \mathbf{R} is a reachability matrix, then \mathbf{R}^0 is defined to be Δ . $\mathbf{R}^i = \mathbf{R}^{i-1} \circ \mathbf{R}$, $i \geq 0$.
- If \mathbf{R}, \mathbf{Q} are two reachability matrices, then define $\mathbf{P} = \mathbf{R}; \mathbf{Q}$ to be $(\mathbf{R} + \Delta) \circ (\mathbf{Q} + \Delta) = \Delta + \mathbf{R} + \mathbf{Q} + \mathbf{R} \circ \mathbf{Q}$ ²
- If \mathbf{R} is a reachability matrix, then define $*\mathbf{R}$ as $+_{i=0}^{\infty} \mathbf{R}^i$.

The above expressions define what we call is a *generalized reachability expression*. They are just like reachability expressions except that they are defined on matrices. It is important to realize that all these expressions are also expressible as a single matrix with more complicated reachability expressions as its elements. However if we are given a set of matrices $\mathcal{R} = \{\mathbf{R}_1, \dots, \mathbf{R}_k\}$ and if we want to combine these using the above operations we will prefer to deal with it as a *generalized reachability expression* over this set \mathcal{R} .

In the following discussion we assume a base transition system $\mathcal{S} = (V, Q, Q_0, \gamma)$ and a set of transition clusters $\mathcal{T} = \{\mathbf{T}_1, \dots, \mathbf{T}_k\}$. We consider a set of all $k \times k$ matrices of reachability expressions formed by these transition clusters. We will denote the set of all reachability matrices of the transition system \mathcal{S} with respect to the clusters \mathcal{T} as $\mathcal{M}_{\mathcal{S}}(\mathcal{T})$, i.e. set of all matrices whose entries are reachability expressions formed using the clusters in \mathcal{T} . If the underlying transition system \mathcal{S} is obvious we will simply denote it by $\mathcal{M}(\mathcal{T})$.

Lemma 4.1 $(\mathcal{M}(\mathcal{T}), +)$ forms commutative idempotent monoid.

¹The form of the semantics of reachability matrix in equation(4.1) is quite unnatural. If we write the vector \bar{S} to the left of the matrix, then it looks very much like multiplication of a matrix to a vector. However we will use this notation because it is consistent with the notations used for reachability expressions in chapter 2.

²We show later that distributive laws hold on reachability matrices, thus the expansion is valid.

Proof :

We observe that the following hold for reachability matrices $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$. These follow from definition. Φ is the matrix defined above which acts as an additive identity.

$$\begin{aligned}\mathbf{R}_1 + (\mathbf{R}_2 + \mathbf{R}_3) &= (\mathbf{R}_1 + \mathbf{R}_2) + \mathbf{R}_3 - \text{Associativity} \\ \mathbf{R}_1 + \mathbf{R}_2 &= \mathbf{R}_2 + \mathbf{R}_1 - \text{Commutativity} \\ \mathbf{R}_1 + \Phi &= \Phi + \mathbf{R}_1 = \mathbf{R}_1 - \text{Additive Identity} \\ \mathbf{R}_1 + \mathbf{R}_1 &= \mathbf{R}_1 - \text{Idempotent}\end{aligned}$$

Thus proved.

Lemma 4.2 $(\mathcal{M}(T), \circ)$ forms a monoid

Proof :

We observe that the following hold for reachability matrices $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$. These follow from definition. Δ is the matrix defined above which acts as multiplicative identity. Note that since the reachability expressions form a semiring(ref. chapter 2) under $+, \circ$ we have that matrix multiplication is associative.

$$\begin{aligned}\mathbf{R}_1 \circ (\mathbf{R}_2 \circ \mathbf{R}_3) &= (\mathbf{R}_1 \circ \mathbf{R}_2) \circ \mathbf{R}_3 - \text{Associativity} \\ \mathbf{R}_1 \circ \Delta &= \Delta \circ \mathbf{R}_1 = \mathbf{R}_1 - \text{Multiplicative Identity}\end{aligned}$$

Thus proved.

Lemma 4.3 $(\mathcal{M}(T), +, \circ)$ forms a semiring

Proof :

We have already shown that this set forms a commutative semigroup(Lemma 4.1)³ under $+$ and a semigroup(Lemma 4.2) under \circ . We now show the distributive laws.

$$\begin{aligned}(\mathbf{R}_1 \circ (\mathbf{R}_2 + \mathbf{R}_3))_{ij} &= +_{l=1}^k (\mathbf{R}_1)_{il} \circ ((\mathbf{R}_2)_{lj} + (\mathbf{R}_3)_{lj}) \\ &= +_{l=1}^k (\mathbf{R}_1)_{il} \circ (\mathbf{R}_2)_{lj} + (\mathbf{R}_1)_{il} \circ (\mathbf{R}_3)_{lj} \\ &= (+_{l=1}^k (\mathbf{R}_1)_{il} \circ (\mathbf{R}_2)_{lj}) + (+_{l=1}^k (\mathbf{R}_1)_{il} \circ (\mathbf{R}_3)_{lj}) \\ &= (\mathbf{R}_1 \circ \mathbf{R}_2)_{ij} + (\mathbf{R}_1 \circ \mathbf{R}_3)_{ij}\end{aligned}$$

. Similarly we can show that the right distributive law holds. Thus we have

$$\begin{aligned}\mathbf{R}_1 \circ (\mathbf{R}_2 + \mathbf{R}_3) &= \mathbf{R}_1 \circ \mathbf{R}_2 + \mathbf{R}_1 \circ \mathbf{R}_3 - \text{Left Distributive} \\ (\mathbf{R}_1 + \mathbf{R}_2) \circ \mathbf{R}_3 &= \mathbf{R}_1 \circ \mathbf{R}_3 + \mathbf{R}_2 \circ \mathbf{R}_3 - \text{Right Distributive}\end{aligned}$$

Thus the set of reachability matrices form a semiring under $+, \circ$.

4.3 Some Results

In this section we show several results which demonstrate the relation between reachability matrices and the underlying reachability expressions.

Theorem 4.1 If \mathbf{R}, \mathbf{Q} are reachability matrices, then $\forall \bar{S}, \llbracket \mathbf{R} + \mathbf{Q} \rrbracket(\bar{S}) = \llbracket \mathbf{R} \rrbracket(\bar{S}) \cup \llbracket \mathbf{Q} \rrbracket(\bar{S})$.

³Monoid is a semigroup with identity

Proof :

Consider the i^{th} element of $\llbracket \mathbf{R} + \mathbf{Q} \rrbracket(\bar{S})$.

$$\begin{aligned}
\llbracket \mathbf{R} + \mathbf{Q} \rrbracket(\bar{S})_i &= \bigcup_{j=1}^k \llbracket \mathbf{R}_{ji} + \mathbf{Q}_{ji} \rrbracket(S_j) \\
&= \bigcup_{j=1}^k \llbracket \mathbf{R}_{ji} \rrbracket(S_j) \cup \llbracket \mathbf{Q}_{ji} \rrbracket(S_j) \\
&= \left(\bigcup_{j=1}^k \llbracket \mathbf{R}_{ji} \rrbracket(S_j) \right) \cup \left(\bigcup_{j=1}^k \llbracket \mathbf{Q}_{ji} \rrbracket(S_j) \right) \\
&= \llbracket \mathbf{R} \rrbracket(\bar{S})_i \cup \llbracket \mathbf{Q} \rrbracket(\bar{S})_i
\end{aligned}$$

Hence proved.

Theorem 4.2 *If \mathbf{R}, \mathbf{Q} are reachability matrices, then $\forall \bar{S}, \llbracket \mathbf{R} \circ \mathbf{Q} \rrbracket(\bar{S}) = \llbracket \mathbf{Q} \rrbracket(\llbracket \mathbf{R} \rrbracket(\bar{S}))$.*

Proof :

First consider the i^{th} element of $\llbracket \mathbf{R} \rrbracket(\bar{S})$.

$$\begin{aligned}
\llbracket \mathbf{R} \rrbracket(\bar{S})_i &= \bigcup_{j=1}^k \llbracket \mathbf{R}_{ji} \rrbracket(S_j) \\
\llbracket \mathbf{Q} \rrbracket(\llbracket \mathbf{R} \rrbracket(\bar{S}))_i &= \bigcup_{l=1}^k \llbracket \mathbf{Q}_{li} \rrbracket(\llbracket \mathbf{R} \rrbracket(\bar{S})_l) \\
&= \bigcup_{l=1}^k \llbracket \mathbf{Q}_{li} \rrbracket \left(\bigcup_{j=1}^k \llbracket \mathbf{R}_{jl} \rrbracket(S_j) \right) \\
&= \bigcup_{l=1}^k \bigcup_{j=1}^k \llbracket \mathbf{Q}_{li} \rrbracket(\llbracket \mathbf{R}_{jl} \rrbracket(S_j)) \\
&= \bigcup_{j=1}^k \bigcup_{l=1}^k \llbracket \mathbf{R}_{jl} \circ \mathbf{Q}_{li} \rrbracket(S_j) \\
&= \bigcup_{j=1}^k \llbracket \bigoplus_{l=1}^k \mathbf{R}_{jl} \circ \mathbf{Q}_{li} \rrbracket(S_j) \\
&= \llbracket \mathbf{R} \circ \mathbf{Q} \rrbracket(\bar{S})_i
\end{aligned}$$

Hence proved.

Next we want to define notion of *cover* among reachability matrices. We say that $\bar{S} = (S_1, \dots, S_k) \subseteq \bar{S}' = (S'_1, \dots, S'_k)$ iff $\forall i, 1 \leq i \leq k, S_i \subseteq S'_i$. We say that a reachability matrix \mathbf{R} is *covered by* a matrix \mathbf{Q} , iff $\forall \bar{S}, \llbracket \mathbf{R} \rrbracket(\bar{S}) \subseteq \llbracket \mathbf{Q} \rrbracket(\bar{S})$. We denote this by $\mathbf{R} \sqsubseteq \mathbf{Q}$. The next result shows the relation between reachability matrices and reachability expressions involved in the matrix with respect to the relation \sqsubseteq . In particular it refines the meaning of $\mathbf{R} \sqsubseteq \mathbf{Q}$ in the case of reachability matrices. We note here that using the fact that $\llbracket \sigma \rrbracket$ is monotonic for σ reachability expression, it trivially follows that $\llbracket \mathbf{R} \rrbracket$ is monotonic for \mathbf{R} reachability matrix.

Theorem 4.3 *Let \mathbf{R}, \mathbf{Q} be reachability matrices. Then $\mathbf{R} \sqsubseteq \mathbf{Q}$ iff $\forall i, j, \mathbf{R}_{ij} \sqsubseteq \mathbf{Q}_{ij}$.*

Proof : We will first prove the if part, i.e. $\forall i, j, \mathbf{R}_{ij} \sqsubseteq \mathbf{Q}_{ij} \implies \mathbf{R} \sqsubseteq \mathbf{Q}$. Consider an arbitrary state vector $\bar{S} = (S_1, \dots, S_k)$. Then we have

$$\begin{aligned} \llbracket \mathbf{R} \rrbracket(\bar{S})_i &= \bigcup_{j=1}^k \llbracket \mathbf{R}_{ji} \rrbracket(S_j) \\ &\subseteq \bigcup_{j=1}^k \llbracket \mathbf{Q}_{ji} \rrbracket(S_j) \\ &= \llbracket \mathbf{Q} \rrbracket(\bar{S})_i \end{aligned}$$

Since this holds for all i by the definition above we have that $\mathbf{R} \sqsubseteq \mathbf{Q}$.

Now to prove the converse. Consider $\bar{S} = (S_1, \dots, S_k)$, where $S_i = A$ for an arbitrary set A , and $\forall j \neq i, S_j = \emptyset$.

$$\begin{aligned} \llbracket \mathbf{R} \rrbracket(\bar{S})_j &= \bigcup_{l=1}^k \llbracket \mathbf{R}_{lj} \rrbracket(S_l) \\ &= \llbracket \mathbf{R}_{ij} \rrbracket(A) \\ \llbracket \mathbf{Q} \rrbracket(\bar{S})_j &= \llbracket \mathbf{Q}_{ij} \rrbracket(A) \end{aligned}$$

Since $\mathbf{R} \sqsubseteq \mathbf{Q}$, $\llbracket \mathbf{R} \rrbracket(\bar{S}) \subseteq \llbracket \mathbf{Q} \rrbracket(\bar{S})$. Hence $\llbracket \mathbf{R}_{ij} \rrbracket(A) \subseteq \llbracket \mathbf{Q}_{ij} \rrbracket(A)$. Since A, i, j are arbitrary, we have that $\forall i, j, \mathbf{R}_{ij} \sqsubseteq \mathbf{Q}_{ij}$. Thus we get the result.

We now state some more basic properties for reachability matrices which will follow immediately from the above results. The proofs are very similar to the case of reachability expressions. These are given in [8]. Here only the aspects of the proofs which are different are mentioned. In most cases just replacing S by \bar{S} and a reachability expression by a reachability matrix suffices.

Property 4.1 If $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2$ and $\mathbf{R}_3 \sqsubseteq \mathbf{R}_4$, then $\mathbf{R}_1 \text{op} \mathbf{R}_3 \sqsubseteq \mathbf{R}_2 \text{op} \mathbf{R}_4$, where $\text{op} \in \{+, \circ, ;\}$.

Property 4.2 $(\mathbf{R}_1; \mathbf{R}_2)^i \sqsubseteq (\mathbf{R}_1; \mathbf{R}_2)^{i+1}$ for all $i \geq 0$.

Property 4.3 If $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2$, then $\mathbf{R}_1^i \sqsubseteq \mathbf{R}_2^i$ for all $i \geq 0$, and $(*\mathbf{R}_1) \sqsubseteq (*\mathbf{R}_2)$.

Property 4.4 $(*\mathbf{R})^i = (*\mathbf{R})$ for all $i \geq 1$, and $*(*\mathbf{R}) = (*\mathbf{R})$.

Proof : In the proof of this property in [8] we use a state vector \bar{S} instead of S . We then consider an arbitrary element $q_k \in S_k$.

Property 4.5 Let τ be a reachability matrix each element of which is $*(\mathbf{T}_1 + \dots + \mathbf{T}_n)$. Then for any arbitrary reachability matrix \mathbf{R} , $\mathbf{R} \sqsubseteq \tau$.

Proof : In the case of reachability expressions we know that $\forall \sigma, \sigma \sqsubseteq *(\mathbf{T}_1 + \dots + \mathbf{T}_n)$. Thus by theorem 4.3 we get that $\mathbf{R} \sqsubseteq \tau$.

Property 4.6 For all reachability matrices \mathbf{R}_1 and \mathbf{R}_2 , $*(\mathbf{R}_1; \mathbf{R}_2) = *(\mathbf{R}_1; *\mathbf{R}_2) = *(*\mathbf{R}_1; \mathbf{R}_2) = *(*\mathbf{R}_1; *\mathbf{R}_2)$.

Lemma 4.4 Let $\mathcal{R} = \{\mathbf{R}_1, \dots, \mathbf{R}_k\}$ be a set of reachability matrices. Let \mathbf{R} be an arbitrary generalized reachability expression based on reachability matrices in \mathcal{R} . Then $\mathbf{R} \sqsubseteq *(\mathbf{R}_1 + \dots + \mathbf{R}_k)$.

Proof :

This can be proved using structural induction which is similar to the proof of lemma 6 given in [8].

We now generalize some of the theorems of chapter 2 for the case of reachability matrices. The first theorem is a generalization of theorem 2.1. Note that the proof is purely algebraic in the case of reachability expressions. Since the algebra is the same for reachability matrices an identical proof is valid and we omit it here.

Theorem 4.4 Let $\{\mathbf{R}_1, \dots, \mathbf{R}_k\}, k \geq 1$, be a finite set of reachability matrices. Then $*(\mathbf{R}_1 + \dots + \mathbf{R}_k) = *(\mathbf{R}_1; \dots; \mathbf{R}_k)$.

The next theorem generalizes theorem 2.2. Again the proof is almost identical. We just give the patch for the proof to handle the case of state vectors.

Theorem 4.5 Let $\{\mathbf{R}_1, \dots, \mathbf{R}_k\}, k \geq 2$, be a finite set of reachability matrices. Let $\mathbf{R}_Y = (\mathbf{R}_1 + \dots + \mathbf{R}_k)$ and $\mathbf{R}_Z = (\mathbf{R}_1; \dots; \mathbf{R}_k)$. Then

1. $\forall n \geq 0, (+_{i=0}^n (\mathbf{R}_Y)^i) \sqsubseteq (\mathbf{R}_Z)^n$
2. If $(\mathbf{R}_p)^2 \sqsubseteq \mathbf{R}_p$ for all p , then $(+_{i=0}^n (\mathbf{R}_Y)^i) \sqsubseteq (\mathbf{R}_Z)^{n - (\lfloor \frac{n}{k} \rfloor - 1)}$.

Proof :

Refer to proof of theorem 2 from [8]. The proof for part 1 holds identically in this theorem. For the proof of part 2, we need to replace S by \bar{S} and $q \in S$ by $q_k \in S_k$. The rest of the proof holds owing to the identical algebraic properties of reachability matrices.⁴

Next we provide a generalization of theorem 2.3. We will not use the idea of reachable states since we do not have a way of providing a good condition about when reachability matrices do give reachable states. We will discuss some sufficient conditions in chapter 5.

Theorem 4.6 If $\{\mathbf{R}_1, \dots, \mathbf{R}_k\}$ are a set of reachability matrices, such that $\mathbf{R}_k \not\sqsubseteq *(\mathbf{R}_1 + \dots + \mathbf{R}_{k-1})$. Let $\mathbf{R}_X = (\mathbf{R}_1 + \dots + \mathbf{R}_{k-1})$ and $\widehat{\mathbf{R}} = *\mathbf{R}_X \circ *(\mathbf{R}_k; *\mathbf{R}_X)$. Then the following hold.

1. $\widehat{\mathbf{R}} = *(\mathbf{R}_1 + \dots + \mathbf{R}_k)$
2. For any \mathbf{R} satisfying $\mathbf{R} = \widehat{\mathbf{R}}$ and for any state vector \bar{S} , we have $N_k(\widehat{\mathbf{R}}, \bar{S}) \leq N_k(\mathbf{R}, \bar{S}) + 1$ where $N_k(\mathbf{R}, \bar{S})$ denotes the number of time image computation is done using the matrix \mathbf{R} until a fixed point is reached starting with \bar{S} .

Proof : We give an intuition for the proof here. The formal proof follows much on the same lines as the proof of theorem 3 given in [8]. We have $*(\mathbf{R}_k; \mathbf{R}_X) = *(\mathbf{R}_k + \mathbf{R}_X) = *(\mathbf{R}_1 + \dots + \mathbf{R}_k)$ (from theorem 4.4). It is easy to see then that $*(\mathbf{R}_1 + \dots + \mathbf{R}_k) = *(\mathbf{R}_k; \mathbf{R}_X) \sqsubseteq \widehat{\mathbf{R}}$. We have $*\mathbf{R}_X \sqsubseteq *(\mathbf{R}_k; *\mathbf{R}_X)$. Hence $*\mathbf{R}_X \circ *(\mathbf{R}_k; *\mathbf{R}_X) \sqsubseteq (*(\mathbf{R}_k; *\mathbf{R}_X))^2 = *(\mathbf{R}_k; *\mathbf{R}_X) = *(\mathbf{R}_1 + \dots + \mathbf{R}_k)$.

We then note that any generalized reachability expression can be thought of as a sequence of matrix operations. Suppose for a particular generalized reachability expression \mathbf{R} we perform M image computations under the matrix \mathbf{R}_k before hitting a fixed point. Then each expression in between the applications of \mathbf{R}_k is expressible by $*\mathbf{R}_X$. Thus we have that the generalized reachability expression $*\mathbf{R}_X; \mathbf{R}_k; *\mathbf{R}_X; \dots; \mathbf{R}_k; *\mathbf{R}_X$ where \mathbf{R}_k is applied M times covers the same computation. We know that the fixed points reached are same. We may need one more application of \mathbf{R}_k to verify the fixed point. Hence we have the required result.

The next result generalizes theorem 2.4. The definition of ordering is more stringent here. Again we will only present a sketch of a proof here. The formal proof is not hard to write.

Theorem 4.7 Let $\{\mathbf{R}_1, \dots, \mathbf{R}_k\}$ be a set of reachability matrices satisfying the condition.

- $*\mathbf{R}_i \circ *\mathbf{R}_{i+1} \circ \dots \circ *\mathbf{R}_k \circ *\mathbf{R}_i \sqsubseteq *\mathbf{R}_i \circ *\mathbf{R}_{i+1} \circ \dots \circ *\mathbf{R}_k$ for all $1 \leq i < k$.

Then $*(\mathbf{R}_1; \dots; \mathbf{R}_k) = (*\mathbf{R}_1); \dots; (*\mathbf{R}_k)$.

Proof :

The condition in plain words says that once a transition matrix \mathbf{R}_i is applied, then applying the matrices indexed greater than it are not going to give new possibilities for applying the matrix \mathbf{R}_i again. We first note that $*\mathbf{R}_1; *\mathbf{R}_2 = *\mathbf{R}_1 \circ *\mathbf{R}_2$. Let $\sigma = (*\mathbf{R}_1) \circ \dots \circ (*\mathbf{R}_k)$. Suppose we show that $\forall i, \sigma^i = \sigma$. Then

⁴It is easy to show that in general matrices over semirings also form a semiring.

we will get $*\sigma = \sigma$. Thus $*(\mathbf{R}_1; \dots; \mathbf{R}_k) = *(*\mathbf{R}_1; \dots; *\mathbf{R}_k) = *(\sigma) = \sigma = *\mathbf{R}_1; \dots; *\mathbf{R}_k$. Thus we will get the required result.

We will show that $\sigma^2 = \sigma$ and then the rest will follow from induction. It is obvious that $\sigma \sqsubseteq \sigma^2$. So we only need to show that $\sigma^2 \sqsubseteq \sigma$. We need to apply the condition repeatedly as shown below.

$$\begin{aligned}
\sigma^2 &= *\mathbf{R}_1 \circ \dots \circ *\mathbf{R}_k \circ *\mathbf{R}_1 \circ *\mathbf{R}_2 \circ \dots \circ *\mathbf{R}_k \\
&\sqsubseteq *\mathbf{R}_1 \circ *\mathbf{R}_2 \circ \dots \circ *\mathbf{R}_k \circ *\mathbf{R}_2 \circ *\mathbf{R}_3 \circ \dots \circ *\mathbf{R}_k \\
&\sqsubseteq \dots \\
&\sqsubseteq *\mathbf{R}_1 \circ \dots \circ *\mathbf{R}_{k-1} \circ *\mathbf{R}_k \circ *\mathbf{R}_{k-1} \circ *\mathbf{R}_k \\
&\sqsubseteq *\mathbf{R}_1 \circ \dots \circ *\mathbf{R}_{k-1} \circ *\mathbf{R}_k \circ *\mathbf{R}_k \\
&\sqsubseteq \sigma
\end{aligned}$$

Thus we get the result.

In chapter 2 we saw that a reachability expression defines a method to make computations on the set of states. The experiments in chapter 3 show that the time and space required for the computations vary significantly based on the schedule used. In a similar fashion the generalized reachability expressions over reachability matrices themselves represent a method to compute reachable state vectors. Thus it needs to be investigated what expressions of reachability matrices allow efficient computation of reachable state vectors. There are still other results which might be generalized in the case of reachability matrices. It is not yet clear what expressions over reachability matrices lead to computation of reachable states. We will see some sufficient conditions in chapter 5 under which reachability matrices compute reachable states. However we do not as yet know a sufficient and necessary condition.

Chapter 5

Partitioning the Reachable States

The state space of a model is the set of all possible valuations of the boolean variables in the model. As the number of boolean variables increases, the size of the state-space blows up exponentially. This problem is handled to some extent by symbolic model checking, as it stores the set of states as BDDs, where the size of the set represented by the BDD may be exponential in the size of the BDD. However, this is not always the case. It is possible that the size of the BDD itself may get very large. In this case it may be possible to represent the set of states as a union of two or more sets of states, which may or may not be disjoint, and represent each of these sets by a BDD. Thus we will store a set of states as a vector of BDDs. We use the theory of reachability matrices developed in chapter 4 to define transitions on state vectors. We try to find heuristics for partitioning sets of states where the size of the individual BDDs may be much smaller than the size of the reachable states.

We will restrict our discussion to the case when $k = 2$. Most results are identical in the case for any k but for the sake of simplicity we will deal with vectors of length 2 and matrices of size 2×2 . We look at some general properties that the state vectors and reachability matrices must satisfy in order to help compute the fixed points efficiently. We look at two possible ways to split the set of states. There might be many more but these seem to be a natural choice. We will then give the corresponding reachability matrices and prove that they can be helpful in computing fixed points which give the set of reachable states.

5.1 Resizing State Vectors

We have seen how a reachability matrix \mathbf{R} acts on a state vector. It is clear from the semantics that there may be several state vectors which give the same resultant image state vector under the reachability matrix \mathbf{R} . It thus might make sense to resize state vectors to those which give the same image under the reachability matrix, but makes the image computation more efficient.

Definition 5.1 Resize Operator *A resize operator ρ , is a function from the set of state vectors to itself.*

Thus if \bar{S} is a state vector, $\rho(\bar{S})$ is also a state vector. We say that a particular resize operator ρ is *consistent* with respect to a reachability matrix \mathbf{R} and a state vector \bar{S} , if $\llbracket \mathbf{R} \rrbracket(\bar{S}) = \llbracket \mathbf{R} \rrbracket(\rho(\bar{S}))$ ¹. The definition of consistency is stringent. The next result is quite straightforward, but nevertheless very important for our discussion. Let $\rho \circ \mathbf{R}$ denote applying the reachability matrix \mathbf{R} to a state vector after resizing it by ρ . We let $*(\rho \circ \mathbf{R})$ take the obvious meaning of applying $(\rho \circ \mathbf{R})$ to a state vector 0, 1, ... times and taking unions at each level.

Lemma 5.1 *Let \mathbf{R} be a reachability matrix. Let \bar{I} be the initial state vector. Let ρ be a resize operator satisfying the following conditions*

¹In this case equality is defined component wise. The case when equality is defined in terms of norm requires further investigation

1. ρ is consistent with respect to \mathbf{R} and \bar{I} , i.e. $\llbracket \mathbf{R} \rrbracket(\bar{I}) = \llbracket \mathbf{R} \rrbracket(\rho(\bar{I}))$.

2. If ρ is consistent with respect to \mathbf{R} and \bar{S} , it is consistent with respect to \mathbf{R} and $\llbracket \mathbf{R} \rrbracket(\bar{S})$.

Then we have that $\llbracket *(\rho \circ \mathbf{R}) \rrbracket(\bar{I}) = \llbracket \mathbf{R} \rrbracket(\bar{I})$.

Proof :

We observe that both the sides are iterative methods to compute fixed points. Thus we prove by induction that they compute the same reachable state vector and in fact in the same number of iterations.

We will prove this by induction on the number of iterations. For $i = 0$, we have the same initial start vector and also we are given that ρ is consistent with respect to \mathbf{R} and \bar{I} . We assume the induction hypothesis, that iteratively the same vectors (call this $\bar{S}_{(k)}$) are computed for up to the k^{th} iteration and that ρ is consistent with \mathbf{R} and $\bar{S}_{(k)}$. We will prove the same holds for $i = k + 1$. Clearly $\bar{S}_{(k+1)} = \llbracket \mathbf{R} \rrbracket(\bar{S}_{(k)}) = \llbracket \mathbf{R} \rrbracket(\rho(\bar{S}_{(k)}))$ is the same state vector for both the methods after the $k + 1^{st}$ iteration and ρ is consistent with respect to \mathbf{R} and $\bar{S}_{(k+1)}$ by the conditions of the lemma. Thus by induction we have that the fixed points computed will be same and in the same number of iterations in either case.

We make it a point here to emphasize the importance of the resize operator ρ while computing fixed points using repeated matrix multiplications. While initially we may start off with a good vector it may very well be the case that the intermediate state vectors have large individual components. In the previous chapter we studied the algebra of reachability matrices. This was important in proving the correct semantics, i.e. that computation performed will actually give the the reachable states. In this chapter we also try to investigate techniques that will actually make efficient computations and keep the components of the state vectors small. The resize operator should be chosen to be such that it preserves the semantics of the reachability matrix, but at the same time improves on the efficiency of computation. We now look at two ways to partition the set of states and discuss some issues related to the same.

5.2 Formulation 1

Consider the current reachable states as a set S . Suppose we could write the transition relation as just two clusters² \mathbf{T}_1 and \mathbf{T}_2 , i.e. $\mathbf{T}_1 \cup \mathbf{T}_2 = \gamma$. In fact we just require that $*(\mathbf{T}_1 + \mathbf{T}_2)$ gives the set of reachable states. We want to maintain a state vector $\bar{S} = (S_1, S_2)$ as the state vector representing the current set of reachable states. Of course if S is the current set of reachable states, then we have that $\|\bar{S}\| = S_1 \cup S_2 = S$. Let $\mathbf{T} = \mathbf{T}_1 + \mathbf{T}_2$. We also want that $\llbracket \mathbf{T}_1 \rrbracket(S_1) = \llbracket \mathbf{T}_1 \rrbracket(S)$ and $\llbracket \mathbf{T}_2 \rrbracket(S_2) = \llbracket \mathbf{T}_2 \rrbracket(S)$. Thus all states which have transitions in \mathbf{T}_1 enabled are in S_1 and all the states which have transitions in \mathbf{T}_2 enabled are in S_2 . The partitioning is based on the idea that the BDD size of the states in which a particular cluster of transitions are active will be small because of the similarities in boolean valuations. We define the reachability matrix in this case as follows:

$$\mathbf{R}_1 = \begin{pmatrix} \mathbf{T}_1 & \mathbf{T}_1 \\ \mathbf{T}_2 & \mathbf{T}_2 \end{pmatrix}$$

We will show that this matrix \mathbf{R}_1 can be used to compute the set of reachable states.

5.2.1 Some Results

The next theorem shows relation between applying the reachability matrix \mathbf{R}_1 to a state vector, and application of the global transition relation \mathbf{T} on the norm of the same state vector. It shows that the norm of the new state vector is just the set of states obtained by applying \mathbf{T} to the norm of the original state vector.

Theorem 5.1 *Let $\bar{S} = (S_1, S_2)$ be a state vector, and let \mathbf{R}_1 be the reachability matrix defined above. Further let $\mathbf{T} = \mathbf{T}_1 + \mathbf{T}_2$. $\|\bar{S}\| = S_1 \cup S_2 = S$. Suppose the conditions listed below hold.*

²This can always be done by clubbing clusters together. In fact we may even use reachability expressions for \mathbf{T}_1 and \mathbf{T}_2 satisfying $*(\mathbf{T}_1 + \mathbf{T}_2)(I) = \text{reachablestates}$.

$$1. \llbracket \mathbf{T}_1 \rrbracket(S) = \llbracket \mathbf{T}_1 \rrbracket(S_1)$$

$$2. \llbracket \mathbf{T}_2 \rrbracket(S) = \llbracket \mathbf{T}_2 \rrbracket(S_2)$$

Then we have that $\llbracket \mathbf{R}_1 \rrbracket(\bar{S}) \models \llbracket \mathbf{T} \rrbracket(S)$. Furthermore let $\llbracket \mathbf{R}_1 \rrbracket(\bar{S}) = \bar{S}' = (S'_1, S'_2)$ (also $\llbracket \bar{S}' \rrbracket = S'$) and we have that

$$1. \llbracket \mathbf{T}_1 \rrbracket(S') = \llbracket \mathbf{T}_1 \rrbracket(S'_1)$$

$$2. \llbracket \mathbf{T}_2 \rrbracket(S') = \llbracket \mathbf{T}_2 \rrbracket(S'_2)$$

Proof : Let $\bar{S}' = (S'_1, S'_2) = \llbracket \mathbf{R}_1 \rrbracket(\bar{S})$. Then by definition we have that

$$\begin{aligned} S'_1 &= \llbracket \mathbf{T}_1 \rrbracket(S_1) \cup \llbracket \mathbf{T}_2 \rrbracket(S_2) \\ S'_2 &= \llbracket \mathbf{T}_1 \rrbracket(S_1) \cup \llbracket \mathbf{T}_2 \rrbracket(S_2) \\ S' &= S'_1 \cup S'_2 \\ &= \llbracket \mathbf{T}_1 \rrbracket(S_1) \cup \llbracket \mathbf{T}_2 \rrbracket(S_2) \\ &= \llbracket \mathbf{T}_1 \rrbracket(S) \cup \llbracket \mathbf{T}_2 \rrbracket(S) \\ &= \llbracket \mathbf{T}_1 + \mathbf{T}_2 \rrbracket(S) \\ &= \llbracket \mathbf{T} \rrbracket(S) \end{aligned}$$

Also note that in this case we always have that $S'_1 = S'_2 = \llbracket \bar{S}' \rrbracket = S'$. Hence it is obvious that the following hold

$$\begin{aligned} \llbracket \mathbf{T}_1 \rrbracket(S'_1) &= \llbracket \mathbf{T}_1 \rrbracket(S') \\ \llbracket \mathbf{T}_2 \rrbracket(S'_2) &= \llbracket \mathbf{T}_2 \rrbracket(S') \end{aligned}$$

Hence proved.

The form of \bar{S}' above may look disheartening, since all the components are identical. This will not avoid any blow-up and in fact result in redundant storage. It is interesting to look at the form of \mathbf{R}_1^2 or in general \mathbf{R}_1^n . From the definition of \mathbf{R}_1 and the definition of multiplication of matrices.

$$\begin{aligned} \mathbf{R}_1^2 &= \begin{pmatrix} \mathbf{T}_1 \circ \mathbf{T}_1 + \mathbf{T}_1 \circ \mathbf{T}_2 & \mathbf{T}_1 \circ \mathbf{T}_1 + \mathbf{T}_1 \circ \mathbf{T}_2 \\ \mathbf{T}_2 \circ \mathbf{T}_1 + \mathbf{T}_2 \circ \mathbf{T}_2 & \mathbf{T}_2 \circ \mathbf{T}_1 + \mathbf{T}_2 \circ \mathbf{T}_2 \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{T}_1 \circ \mathbf{T} & \mathbf{T}_1 \circ \mathbf{T} \\ \mathbf{T}_2 \circ \mathbf{T} & \mathbf{T}_2 \circ \mathbf{T} \end{pmatrix} \\ \mathbf{R}_1^n &= \begin{pmatrix} \mathbf{T}_1 \circ \mathbf{T}^{n-1} & \mathbf{T}_1 \circ \mathbf{T}^{n-1} \\ \mathbf{T}_2 \circ \mathbf{T}^{n-1} & \mathbf{T}_2 \circ \mathbf{T}^{n-1} \end{pmatrix} \end{aligned}$$

We have seen in theorem 5.1 that whatever the start state vector, after one application of the matrix \mathbf{R}_1 we will have all components of the state vector equal, i.e. if $\bar{S}' = \llbracket \mathbf{R}_1 \rrbracket(\bar{S})$ then $S'_1 = S'_2$. Also we see \mathbf{T}^{n-1} show up in each entry of the reachability matrix \mathbf{R}_1^n . Thus we can see that we will not have much use of a plain partitioning of the set of states in this manner. Before we look at ways to avoid this problem, we will see how the computation of fixed points goes in this case. Since we are dealing with finite state-spaces, we can guarantee that an iterative approach will lead to fixed points. The semantics of $\llbracket *\mathbf{R}_1 \rrbracket(\bar{S})$ are

$$\llbracket *\mathbf{R}_1 \rrbracket(\bar{S}) = \bar{S} \cup \llbracket \mathbf{R}_1 \rrbracket(\bar{S}) \cup \llbracket \mathbf{R}_1^2 \rrbracket(\bar{S}) \cup \dots$$

where union of state vectors is defined by component-wise union.

Suppose that the initial vector \bar{I} is such that $\llbracket \bar{I} \rrbracket$ is the set of initial states. Suppose also that it satisfies the hypothesis of theorem 5.1. Then from theorem 5.1 it is easy to see that if $\bar{R} = \llbracket *\mathbf{R}_1 \rrbracket(\bar{I})$, then $\llbracket \bar{R} \rrbracket$ is the set of reachable states. It can be shown that $\llbracket *\mathbf{R}_1 \rrbracket(\bar{I}) \models \llbracket *\mathbf{T} \rrbracket(\llbracket \bar{I} \rrbracket)$.

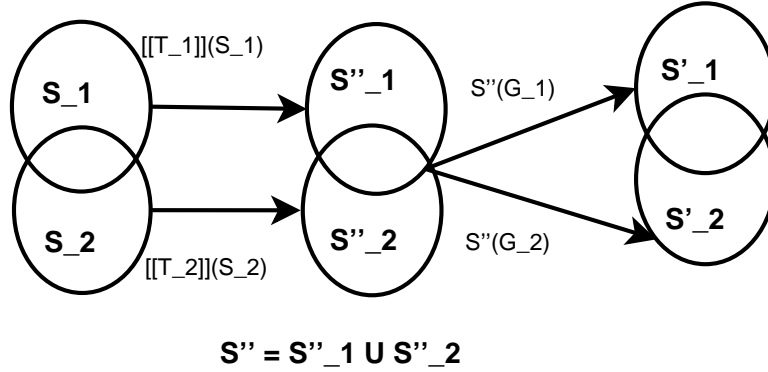


Figure 5.1: Diagram demonstrating application of formulation 1

5.2.2 Avoiding the Blow-up

We now look at ways to get over the hurdle of blow-up of components of the reachable states vector in this case. This is where the resize operator ρ plays an important role. One way of ensuring that the components S_1 and S_2 of the current reachable state vector don't get too large is to minimize their intersection. The idea behind separating these states was that all states from which there are transitions in \mathbf{T}_1 enabled should be in S_1 and those which have transitions in \mathbf{T}_2 enabled in S_2 . On obtaining S_1, S_2 we can reduce them so that S_1 contains only those states which have transitions in \mathbf{T}_1 enabled. Similarly S_2 is a set which contains only those states which have transitions in \mathbf{T}_2 enabled. The way to go about doing this is intersection with the guards. The transitions in the clusters $\mathbf{T}_1, \mathbf{T}_2$ are of the form $(G \mapsto A)$ where G is a guard and A is an action. Let $\mathbf{G}_1 = \bigvee_{(G \mapsto A) \in \mathbf{T}_1} G$, and $\mathbf{G}_2 = \bigvee_{(G \mapsto A) \in \mathbf{T}_2} G$. Note that $\mathbf{G}_1, \mathbf{G}_2$ represent the set of all states in the state space in which some guard of the clusters $\mathbf{T}_1, \mathbf{T}_2$ respectively are enabled. Suppose that \bar{S} was a start state vector, and on applying the reachability matrix \mathbf{R}_1 we get $\bar{S}' = (S'_1, S'_2)$. Then we resize \bar{S}' as

$$\begin{aligned} \rho(S'_1) &= S'_1 \wedge \mathbf{G}_1 \\ \rho(S'_2) &= S'_2 \wedge \mathbf{G}_2 \end{aligned}$$

This is the ρ operator that we have defined to resize the state vectors every now and then. We can easily show it to be consistent as required by lemma 5.1. By making this resize after every iteration or every few iterations during the computation of fixed point, the result of theorem 5.1 that

1. $\llbracket \mathbf{T}_1 \rrbracket(S') = \llbracket \mathbf{T}_1 \rrbracket(S'_1)$
2. $\llbracket \mathbf{T}_2 \rrbracket(S') = \llbracket \mathbf{T}_2 \rrbracket(S'_2)$

are still satisfied. Thus the fixed point computed by this method will indeed give the set of reachable states. And now we see that the intersection of S'_1 and S'_2 tends to be small, that is just the set of states from which there are transitions in both \mathbf{T}_1 and \mathbf{T}_2 enabled. This is a general method proposed, and depending on the nature of the problem, we must take care to choose the clusters \mathbf{T}_1 and \mathbf{T}_2 correctly, so that the components of the state vector will be evenly distributed. A possible way to do this could be to have discrete transitions clubbed under \mathbf{T}_1 and the timed transitions under \mathbf{T}_2 . It is also possible to choose more than two transition clusters and a longer state vector, since all of this analysis will go through identically in that case.

5.3 Formulation 2

In the previous case we had partitioned the set of states based upon which transitions are active in the state. Now we will partition the set of states based upon which transition can get us into the state. Given

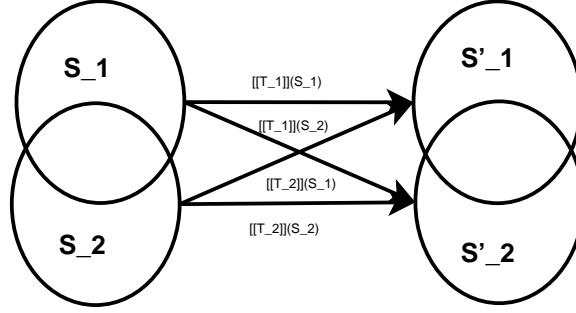


Figure 5.2: Diagram demonstrating application of formulation 2

the state vector $\bar{S} = (S_1, S_2)$ based upon the transition relations \mathbf{T}_1 and \mathbf{T}_2 , we want that each state in S_1 is reachable from some state by a transition in \mathbf{T}_1 and each state in S_2 is reachable from some state by a transition in \mathbf{T}_2 . We use the following reachability matrix in this case

$$\mathbf{R}_2 = \begin{pmatrix} \mathbf{T}_1 & \mathbf{T}_2 \\ \mathbf{T}_1 & \mathbf{T}_2 \end{pmatrix}$$

We again show a theorem similar to theorem 5.1.

5.3.1 Some Results

Theorem 5.2 *Let $\bar{S} = (S_1, S_2)$ be the start state vector and \mathbf{R}_2 is the reachability matrix defined above. Let $S' = \llbracket \mathbf{R}_2 \rrbracket(\bar{S})$. Let $\mathbf{T} = \mathbf{T}_1 + \mathbf{T}_2$. Then we have that $\llbracket \bar{S}' \rrbracket = \llbracket \mathbf{T} \rrbracket(\llbracket \bar{S} \rrbracket)$.*

Proof :

Consider \bar{S}'

$$\begin{aligned} S'_1 &= \llbracket \mathbf{T}_1 \rrbracket(S_1) \cup \llbracket \mathbf{T}_1 \rrbracket(S_2) \\ &= \llbracket \mathbf{T}_1 \rrbracket(S_1 \cup S_2) \\ &= \llbracket \mathbf{T}_1 \rrbracket(\llbracket \bar{S} \rrbracket) \\ S'_2 &= \llbracket \mathbf{T}_2 \rrbracket(\llbracket \bar{S} \rrbracket) \\ \llbracket \bar{S}' \rrbracket &= S'_1 \cup S'_2 \\ &= \llbracket \mathbf{T}_1 \rrbracket(\llbracket \bar{S} \rrbracket) \cup \llbracket \mathbf{T}_2 \rrbracket(\llbracket \bar{S} \rrbracket) \\ &= \llbracket \mathbf{T}_1 + \mathbf{T}_2 \rrbracket(\llbracket \bar{S} \rrbracket) \\ &= \llbracket \mathbf{T} \rrbracket(\llbracket \bar{S} \rrbracket) \end{aligned}$$

Thus proved.

Again it is worth investigation what form \mathbf{R}_2^n takes. The forms for \mathbf{R}_2^2 and \mathbf{R}_2^n are shown below.

$$\begin{aligned} \mathbf{R}_2^2 &= \begin{pmatrix} \mathbf{T}_1 \circ \mathbf{T}_1 + \mathbf{T}_2 \circ \mathbf{T}_1 & \mathbf{T}_1 \circ \mathbf{T}_2 + \mathbf{T}_2 \circ \mathbf{T}_2 \\ \mathbf{T}_1 \circ \mathbf{T}_1 + \mathbf{T}_2 \circ \mathbf{T}_1 & \mathbf{T}_1 \circ \mathbf{T}_2 + \mathbf{T}_2 \circ \mathbf{T}_2 \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{T} \circ \mathbf{T}_1 & \mathbf{T} \circ \mathbf{T}_2 \\ \mathbf{T} \circ \mathbf{T}_1 & \mathbf{T} \circ \mathbf{T}_2 \end{pmatrix} \\ \mathbf{R}_2^n &= \begin{pmatrix} \mathbf{T}^{n-1} \circ \mathbf{T}_1 & \mathbf{T}^{n-1} \circ \mathbf{T}_2 \\ \mathbf{T}^{n-1} \circ \mathbf{T}_1 & \mathbf{T}^{n-1} \circ \mathbf{T}_2 \end{pmatrix} \end{aligned}$$

Again observe the expression for \mathbf{R}_2^n . In this case we see that \mathbf{T}^{n-1} is composed by a \mathbf{T}_1 which can ensure that the size of the set is limited. In this case, in fact the new state vector obtained has

components which are exact, i.e. S'_1 is exactly the set of states which are obtained by a transition of \mathbf{T}_1 from some earlier state and similarly S'_2 . Thus there is no natural resize operator evident here. It is likely that the overlap between the two components in this case is also large. Again from theorem 5.2 it is easy to show that the fixed point reached in this case will be exactly the set of reachable states. If \bar{I} is the initial vector ($\|\bar{I}\|$ is the set of initial states), then we have that if $\bar{R} = \llbracket *R_2 \rrbracket(\bar{I})$, then $\|\bar{R}\|$ is the set of reachable states. We present some experiments in the case of circuits using this formulation in the next chapter.

Chapter 6

Experiments with Reachability Matrices

In this chapter we present some experiments based on the theory developed in chapters 4 and 5. The experiments were performed on asynchronous circuits using the gate level model as described in chapter 3. We describe the setup of the experiment and make some comments on the results.

6.1 Algorithm for Model Checking

We split the transition relation into two parts \mathbf{T}_d and \mathbf{T}_t which represent the discrete and timed transitions respectively. Then we use the reachability matrix

$$\mathbf{R} = \begin{pmatrix} \mathbf{T}_d & \mathbf{T}_t \\ \mathbf{T}_d & \mathbf{T}_t \end{pmatrix}$$

This is the matrix defined in the formulation 2 in chapter 5. Below is the pseudo-code for the computation of fixed points. Here the sets of states are represented as BDDs.

```
Sd := I; // I is set of initial states
St := I;
vd := 1; vt := 1;
while(vd + vt) {
  S'd :=  $\llbracket \mathbf{T}_d \rrbracket(S_d) \vee \llbracket \mathbf{T}_d \rrbracket(S_t)$ ;
  S't :=  $\llbracket \mathbf{T}_t \rrbracket(S_d) \vee \llbracket \mathbf{T}_t \rrbracket(S_t)$ ;
  vd := S'd ∧ ¬Sd;
  vt := S't ∧ ¬St;
  Sd := Sd ∨ S'd; St := St ∨ S't;
  vd := zero?(Sd); vt := zero?(St);
}
//Reachable states are in Sd ∪ St.
```

Figure 6.1: Algorithm for computing reachable state vector

Suppose that the gate level discrete transition relations are $\mathbf{T}_1, \dots, \mathbf{T}_k$ and the time transition relations are $\mathbf{Rt}_1, \dots, \mathbf{Rt}_k$, then we use $\mathbf{T}_d = \mathbf{T}_1; \dots; \mathbf{T}_k$ and $\mathbf{T}_t = \mathbf{Rt}_1 \circ \dots \circ \mathbf{Rt}_k$. We know that $\ast(\mathbf{T}_1 + \mathbf{T}_2)$ gives the set of reachable states and thus using theorem 5.2 we get that the fixed point obtained by using \mathbf{R} will indeed give the set of reachable states as the norm of the final state vector.

Ckt	Δ_{in}	D	(l, u)	σ_1			σ_2			\mathbf{R}		
				Time(s)	Itr	BDD	Time(s)	Itr	BDD	Time(s)	Itr	BDD
2	4	3	(1,2)	1.25	39	3249	2.44	57	3905	1.53	57	2205
2	12	10	(6,8)	6.18	123	8131	19.18	148	26917	14.43	149	18107
3	4	3	(1,2)	1.31	31	4683	2.74	46	6041	1.59	46	3082
3	12	10	(6,8)	8.22	103	11680	92.08	127	162438	105.46	127	123972
10	4	3	(1,2)	83.82	28	108010	126.26	38	115705	29.5	38	35045
10	12	10	(6,8)	280.08	86	110696	533.64	104	245986	219.45	104	132572
17	4	3	(1,2)	0.96	27	3659	1.3	37	3699	0.61	38	1027
17	12	10	(6,8)	4.08	83	7480	5.1	94	7495	1.33	94	1729

Table 6.1: Performance comparison between partitioned and unpartitioned state space

We use the reachability expressions $\sigma_1 = *(\mathbf{T}_d; \mathbf{T}_t)$ and $\sigma_2 = *(\mathbf{T}_d + \mathbf{T}_t)$ without partitioning the state space to compare the effects of partitioning. Note that it is more appropriate to compare the performance of the reachability matrix \mathbf{R} with σ_2 rather than σ_1 given the way the algorithm given in figure 6.1 operates. In the case of \mathbf{R} and σ_2 the algorithms for model checking are almost identical except for the partitioning, so it will serve as a good comparison.

A look at table 6.1 shows that \mathbf{R} outperforms σ_2 in almost all the cases. In several cases it also outperforms σ_1 . In particular it is much better than others in ckt 10, which is a 4 bit - bitwise XOR implemented using AND, OR and NOT gates. This may be due to the large number of parallel transitions possible in this circuit. All the experiments were performed on an IBM ThinkPad with Intel Pentium M 1.86 GHz processor with 512 MB of main memory running Windows XP (with cygwin).

The experiments in this chapter are very preliminary and the code is just a hack into the existing NuSMVDP. But even these primitive experiments show encouraging results. In particular we have used the second formulation which we conjectured will not work very well because it does not allow for any scope of resizing the state vector. Much further work needs to be done on finding good reachability matrices and resize operators.

Chapter 7

Conclusions and Future Work

We have seen that in model checking systems with large state spaces partitioning of transition relations play a major role. Reachability expressions provide a formal framework for analyzing the schedules used to compute the set of reachable states. They are also helpful in proving semantics for the same.

Analogous to partitioning the transition relations, we believe that partitioning the reachable state space may also give benefits. In this case we need to introduce the idea of state vectors rather than a single set of states. Reachability matrices provide a method to manipulate state vectors much the same way as reachability expressions transform sets of states. We have provided a fairly general theory of reachability matrices on the lines of the theory of reachability expressions.

The *generalized reachability expressions* over a set of reachability matrices define a method to compute the reachable state vectors. Although a generalized reachability expression may be collapsed to a single reachability matrix there are advantages for not doing so. For example the matrices in an expression may be sparse whereas the resultant matrix need not be sparse at all. We might also want to intertwine resize operators with reachability matrices.

We now need to begin a more comprehensive set of experiments to judge the benefits of partitioning the state space. The preliminary results do look positive, but the gains are not substantial. There will be trade-offs between smaller sizes and larger number operations as we make more partitions and increase the size of the matrices we deal with. This needs to be investigated.

As a final note we must keep in mind that pushing through larger circuits is important. For example, we have not seen any substantial benefits of partitioning sets of states in the case of 74181 adder. While it is certainly of advantage to reduce the time required for model checking smaller circuits, significant challenge lies in pushing the bounds of symbolic model checking to these larger circuits.

References

- [1] Rajeev Alur. Timed automata. In *Computer Aided Verification*, pages 8–22, 1999.
- [2] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.
- [3] Alessandro Cimatti, Edmund M. Clarke, Fausto Giunchiglia, and Marco Roveri. NUSMV: A new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.
- [4] J.R. Burch, E.M. Clarke, and D.E. Long. Symbolic model checking with partitioned transition relations. In A. Halaas and P.B. Denyer, editors, *International Conference on Very Large Scale Integration*, pages 49–58, Edinburgh, Scotland, 1991. North-Holland.
- [5] J.R. Burch, E.M. Clarke, D.E. Long, K.L. MacMillan, and D.L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(4):401–424, 1994.
- [6] Dina Thomas, Supratik Chakraborty, and Paritosh Pandya. NuSMV-DP : *User’s Manual*. CFDVS, IIT Bombay, <http://www.cse.iitb.ac.in/~dina/research/NuSMVDP>.
- [7] Dina Thomas, Supratik Chakraborty, and Paritosh Pandya. Efficient guided symbolic reachability using reachability expressions. In *TACAS*, 2006.
- [8] Dina Thomas, Supratik Chakraborty, and Paritosh Pandya. Efficient guided symbolic reachability using reachability expressions. Technical Report TR-06-19, CFDVS, IIT Bombay, 2006.