Efficient Guided Symbolic Reachability Using Reachability Expressions

Dina Thomas[1], Supratik Chakraborty[1], and Paritosh Pandya[2]

[1] Indian Institute of Technology, Bombay, India
dina@cfdvs.iitb.ac.in, supratik@cse.iitb.ac.in
[2] Tata Institute of Fundamental Research, India
pandya@tifr.res.in

Abstract. Asynchronous systems consist of a set of transitions which are non-deterministically chosen and executed. We present a theory of guiding symbolic reachability in such systems by scheduling clusters of transitions. A theory of reachability expressions which specify the schedules is presented. This theory allows proving equivalence of different schedules which may have radically different performance in BDD-based search. We present experimental evidence to show that optimized reachability expressions give rise to significant performance advantages. The profiling is carried out in the NuSMV framework using examples from discrete timed automata and circuits with delays. A variant tool called NuSMV-DP has been developed for interpreting reachability expressions to carry out the experiments.

1 Introduction

Asynchronous systems consist of a set of processes which execute independently of each other and synchronize occasionally. A standard model of their execution consists of non-deterministically interleaving the actions of individual processes. Activities of such processes can be modeled by a *global transition system* consisting of a set of guarded transitions ($G \mapsto A$). The system starts non-deterministically in one of a set of designated initial states. In any state, one of the enabled transitions is non-deterministically chosen and executed atomically. This causes a state change. This process is then repeated until no new state is reached. Safety verification of such systems typically reduces to exploring whether some undesirable state is reachable by some execution.

Symbolic model checking [7] has emerged as an important technique for program verification and for finding deep logical bugs in reactive systems. Programs are modeled as finite state transition systems. BDD-based [3] symbolic search techniques, e.g. those used in NuSMV [6], can explore very large but finite state spaces efficiently. However, there is wide variability in the computational efficiency of BDD-based searches. It is well-known that the performance of these techniques strongly depends on the size of the BDD representation of transition relations and of intermediate sets of states.

H. Hermanns and J. Palsberg (Eds.): TACAS 2006, LNCS 3920, pp. 120–134, 2006.

Earlier work in this area has addressed this issue by identifying good variable orders for BDDs representing transition relations [1], by partitioning the transition relation conjunctively or disjunctively [8], and by determining good quantification schedules for conjunctive partitioning [4, 5]. Yet another technique is to use guided search where hints [2] are used to direct initial parts of the search.

In this paper, we propose using clusters of individual guarded transitions within the global transition system in ways that generalize conjunctive and disjunctive partitioning. We introduce a notation called *reachability expressions* and investigate its algebraic properties. Each reachability expression denotes a way of computing the set of final states from a set of initial states. Thus, each reachability expression is a predicate transformer. Reachability expressions are rich enough to specify diverse search strategies such as symbolic breadth-first, round-robin etc. They also include familiar operations like sequential composition, union and Kleene closure. Importantly, they allow us to encode more efficient ways of computing the set of reachable states than symbolic breadth-first search.

We have implemented an interpreter for reachability expressions in a tool called NuSMV-DP that works as a wrapper on top of the reachability engine of NuSMV [6]. Using the semantics of reachability expressions, we show that several distinct reachability expressions are equivalent, i.e. they compute the same predicate transformation. However, as our experiments show, the computational effort involved in applying these equivalent predicate transformers to a given predicate using our BDD-based NuSMV-DP tool can vary significantly. We discuss some equivalence transformations which improve the efficiency of evaluating reachability expressions on finite state systems. A Kleene algebra of reachability expressions has also been investigated by Pandya and Raut [9].

We apply our theory to some examples drawn from discrete timed automata. Such automata can be represented as finite state global transition systems, as discussed in [11]. We experimentally evaluate the performance of BDD-based symbolic reachability analysis on these examples using multiple equivalent reachability expressions. A previous technical report [11] gives details of experiments carried out using NuSMV-DP with diverse problems such as the Fischer protocol, the job-shop scheduling problem and some asynchronous circuits with delays. These experiments show a significant improvement in the efficiency of computing reachable states using our technique. For example, we have been able to model check the Fischer protocol with 100 processes, whereas classical techniques such as symbolic search using polyhedra or difference bound matrices can handle instances of this protocol with only up to 20 processes. On other examples, the relative gains are significant but more modest.

The remainder of this paper is organized as follows. In Section 2, we present the basic theory of reachability expressions. Section 3 gives experimental results obtained by applying this theory to improve the efficiency of reachability analysis. The experiments are carried out using the NuSMV-DP tool. Finally, we conclude the paper in Section 4.

2 A Theory of Reachability Expressions

We consider a state transition system as a 4-tuple (V, Q, Q_0, Υ), where V is a finite set of *state variables*, Q is the set of states, $Q_0 \subseteq Q$ is the set of initial states, and Υ is a finite set of *guarded actions*. Each variable $v_i \in V$ has an associated domain $\mathcal{D}_i$. A state $q \in Q$ is an assignment of a value from $\mathcal{D}_i$ to each variable v_i in V. The set of all such assignments constitutes the set of states Q. A *guarded action* is a pair $(G \mapsto A)$, where the guard G denotes a Boolean combination of predicates on the variables in V or a constant in $\{\mathsf{True}, \mathsf{False}\}$. The action A is either a multiple assignment statement denoting simultaneous assignments to a finite set of variables in V, or the special action skip. We leave the concrete syntax of guarded actions unspecified. Note that each state variable $v_i \in V$ may be assigned a value from $\mathcal{D}_i$ at most once in A. The execution semantics of guarded actions is as usual: If A consists of the special action skip, the values of all state variables remain unchanged. If the system is in state s_1, and if the corresponding assignment of values to variables in V satisfies the guard G, we say that the guarded action $(G \mapsto A)$ is *enabled* in state s_1. The new state reached after executing the action A from state s_1 is obtained by simultaneously assigning to all variables that have been assigned in A, their corresponding values. All state variables that have not been assigned in A retain their values from state s_1 in state s_2.

Let $B = (V, Q, Q_0, \Upsilon)$ be a state transition system. We define a *cluster* to be a non-empty set of guarded actions of B. We also define a special singleton set δ consisting of the guarded action $(\mathsf{True} \mapsto \mathsf{skip})$. Thus, the action in δ can be executed from every state, and its execution takes every state to itself. The empty set of guarded actions is denoted by Θ. An *extended cluster* of the state transition system B is either a subset of Υ or δ or Θ. Every extended cluster C defines a relation, R_C, on the set Q of states. We say that $(s_1, s_2) \in R_C$ iff there exists a guarded action $(G \mapsto A) \in C$ such that G evaluates to True in s_1, and action A takes the system from s_1 to s_2. Given an extended cluster T and a set of states $S\,(\subseteq Q)$, the *image* of S under T is $Im_T(S) = \{s \mid \exists s' \in S,\ R_T(s', s)\}$. It is easy to see that $Im_T : 2^Q \to 2^Q$ is a monotone function.

2.1 Syntax and Semantics

Let $\mathcal{T} = \{\tau_1, \ldots \tau_k\}$ be a set of extended clusters of B. Syntactically, a *reachability expression* over $\mathcal{T}$ is a terminal string obtained from the following grammar:

$$\mathsf{E} \ \rightarrow\ \mathsf{E} + \mathsf{E} \ \mid \mathsf{E}\,;\,\mathsf{E} \ \mid \mathsf{E} \circ \mathsf{E} \ \mid (\mathsf{E}) \ \mid *\mathsf{E} \ \mid \mathbf{T_1} \mid \ldots \mid \mathbf{T_k}$$

In the above syntax, we have used $\mathbf{T_1}, \ldots, \mathbf{T_k}$ to denote reachability expressions corresponding to the extended clusters $\tau_1, \ldots, \tau_k$ in B. We will use this notation throughout this paper.

The notion of evaluating reachability expressions can be formalized by defining their semantics. The semantics of a reachability expression is defined with respect to an underlying state transition system, and is naturally described as a mapping from sets of states to sets of states. Let $B = (V, Q, q_o, \Upsilon)$ be a state transition system, and $\mathcal{T}$ be a set of extended clusters of B. Let σ be a reachability expression over $\mathcal{T}$ and let $S \subseteq Q$. The semantics of σ with respect to B, denoted by $[\![\sigma]\!]_B$, is a mapping from 2^Q to 2^Q defined inductively as follows. We shall henceforth omit the subscript B when it is clear from the context.

- $[\![\mathbf{T_i}]\!](S) = Im_{\tau_i}(S)$, for all $\tau_i \in \mathcal{T}$
- $[\![\sigma_1 + \sigma_2]\!](S) = [\![\sigma_1]\!](S) \cup [\![\sigma_2]\!](S)$
- $[\![\sigma_1 \circ \sigma_2]\!](S) = [\![\sigma_2]\!]([\![\sigma_1]\!](S))$
- $[\![\sigma_1 \,;\, \sigma_2]\!](S) = [\![(\sigma_1 + \delta) \circ (\sigma_2 + \delta)]\!](S)$
- $[\![(\sigma)]\!](S) = [\![\sigma]\!](S)$
- $[\![*\sigma]\!](S) = \bigcup_{i=0}^{\infty} [\![(\sigma)^i]\!](S)$

Although "$;$" is seen to be a derived operator, we retain it for notational convenience.

2.2 Properties of Reachability Expressions

Let σ_1 and σ_2 be reachability expressions over a set $\mathcal{T}$ of symbolic extended clusters. We say that σ_1 is *covered by* σ_2 iff $[\![\sigma_1]\!]_B(S) \subseteq [\![\sigma_2]\!]_B(S)$ for every state transition system B, for every subset S of states of B, and for every instantiation of symbolic clusters in $\mathcal{T}$ with extended clusters of B. We denote this by $\sigma_1 \sqsubseteq \sigma_2$. We say that $\sigma_1 = \sigma_2$, iff $\sigma_1 \sqsubseteq \sigma_2$ and $\sigma_2 \sqsubseteq \sigma_1$. For example, it can be shown from the semantics of reachability expressions that $(\sigma_1 \,;\, \sigma_2) = \delta + \sigma_1 + \sigma_2 + (\sigma_1 \circ \sigma_2)$.

Given a set $\mathcal{T}$ of extended clusters, let $\Pi(\mathcal{T})$ denote the set of all reachability expressions over $\mathcal{T}$. It can be shown that $(\Pi(\mathcal{T}), +)$ forms an idempotent, commutative monoid with Θ as the identity element. Similarly, $(\Pi(\mathcal{T}), \circ)$ forms a monoid with δ as the identity element, and $(\Pi(\mathcal{T}), +, \circ)$ forms an idempotent semiring. In particular, "$\circ$" distributes over "$+$" from both left and right. Detailed proofs of these properties are given in the extended version of this paper [10].

Lemma 1. *Let $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ be reachability expressions.*

(a) If $\sigma_1 \sqsubseteq \sigma_2$ and $\sigma_3 \sqsubseteq \sigma_4$, then $(\sigma_1\, op\, \sigma_3) \sqsubseteq (\sigma_2\, op\, \sigma_4)$, where $op \in \{+, \circ, ;\}$.
(b) $(\sigma_1 \,;\, \sigma_2)^i \sqsubseteq (\sigma_1 \,;\, \sigma_2)^{i+1}$ for all $i \geq 0$.
(c) If $\sigma_1 \sqsubseteq \sigma_2$, then $(\sigma_1)^i \sqsubseteq (\sigma_2)^i$ for all $i \geq 0$, and $(*\sigma_1) \sqsubseteq (*\sigma_2)$.
(d) $(*\sigma_1)^i = (*\sigma_1)$ for all $i \geq 1$, and $*(*\sigma_1) = (*\sigma_1)$.

Proof sketch. Part (a) follows from the semantics of reachability expressions. Parts (b) through (d) are proved by induction on i. Detailed proofs are given in the extended version of this paper [10].

Lemma 2. *For all reachability expressions σ_1 and σ_2, $*(\sigma_1 \,;\, \sigma_2) = *(\sigma_1 \,;\, (*\sigma_2)) = *((*\sigma_1) \,;\, \sigma_2) = *((*\sigma_1) \,;\, (*\sigma_2))$.*

Proof sketch. By the semantics of "$*$", we have $\sigma_2 \sqsubseteq (*\sigma_2)$. Therefore, by Lemma 1(a), $(\sigma_1 \,;\, \sigma_2) \sqsubseteq (\sigma_1 \,;\, (*\sigma_2))$, and by Lemma 1(c), $*(\sigma_1 \,;\, \sigma_2) \sqsubseteq *(\sigma_1 \,;\, (*\sigma_2))$.

To show that $*(\sigma_1 \,;\, (*\sigma_2)) \sqsubseteq *(\sigma_1 \,;\, \sigma_2)$, we first prove that $(\sigma_1 \,;\, (*\sigma_2)) \sqsubseteq *(\sigma_1 \,;\, \sigma_2)$. Details of this proof are available in the extended version of our paper [10].

Applying Lemmas 1(c) and (d), we then get $*(\sigma_1 \,;\, (*\sigma_2)) \sqsubseteq *(\sigma_1 \,;\, \sigma_2)$. Since we also have $*(\sigma_1 \,;\, \sigma_2) \sqsubseteq *(\sigma_1 \,;\, (*\sigma_2))$, it follows that $*(\sigma_1 \,;\, \sigma_2) = *(\sigma_1 \,;\, (*\sigma_2))$. The proof of $*(\sigma_1 \,;\, \sigma_2) = *((*\sigma_1) \,;\, \sigma_2)$ is similar with the roles of σ_1 and σ_2 interchanged.

If we substitute $*\sigma_1$ for σ_1 in the result proved above, we get $*((*\sigma_1) \,;\, \sigma_2) = *((*\sigma_1) \,;\, (*\sigma_2))$. However, $*((*\sigma_1) \,;\, \sigma_2) = *(\sigma_1 \,;\, \sigma_2)$, as argued above. Therefore, $*(\sigma_1 \,;\, \sigma_2) = *((*\sigma_1) \,;\, (*\sigma_2))$.

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

Proof sketch. We prove the theorem by induction on k.
Basis $(k = 1)$: The result holds trivially.
Hypothesis: Assume the result holds for all k in 1 through m.
Induction step: Consider a set of $m+1$ reachability expressions, and let $\sigma_Y = (\sigma_2 + \cdots + \sigma_{m+1})$. We first prove that $*(\sigma_1 + \sigma_Y) = *(\sigma_1 \,;\, \sigma_Y)$. Details of this proof are given in the extended version of this paper [10].

By Lemma 2, we know that $*(\sigma_1 \,;\, \sigma_Y) = *(\sigma_1 \,;\, (*\sigma_Y))$. By the induction hypothesis, $(*\sigma_Y) = (*\sigma_Z)$, where $\sigma_Z = (\sigma_2 \,;\, \ldots \,;\, \sigma_{m+1})$. Therefore, $*(\sigma_1 + \sigma_Y) = *(\sigma_1 \,;\, (*\sigma_Z))$. Applying Lemma 2 again, we get $*(\sigma_1 + \sigma_Y) = *(\sigma_1 \,;\, \sigma_Z)$. The proof is completed by noting that "+" and ";" are associative.

Theorem 2. *Let $\{\sigma_1, \ldots, \sigma_k\}$, $k \geq 2$, be a finite set of reachability expressions. Let $\sigma_Y = (\sigma_1 + \cdots + \sigma_k)$ and $\sigma_Z = (\sigma_1 \,;\, \cdots \,;\, \sigma_k)$.*

(a) For all $n \geq 0$, $(\overset{n}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq (\sigma_Z)^n$.

(b) Furthermore, if $(\sigma_p)^2 \sqsubseteq \sigma_p$ for all p, then $(\overset{n}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq (\sigma_Z)^{n-(\lfloor n/k \rfloor - 1)}$.

Proof sketch.

(a) Since $\sigma_Y \sqsubseteq \sigma_Z$, by Lemma 1(c), $(\sigma_Y)^j \sqsubseteq (\sigma_Z)^j$ for all $j \geq 0$. Therefore, $(\overset{n}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq (\overset{n}{\underset{j=0}{+}} (\sigma_Z)^j)$. Applying Lemma 1(b), it can now be shown that $(\overset{n}{\underset{j=0}{+}} (\sigma_Z)^j) = \sigma_Z^n$. Hence, $(\overset{n}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq \sigma_Z^n$.

(b) Let $n = k.i + r$, where $i \geq 0$ and $0 \leq r < k$. Using induction on k, it can be shown that $(\overset{k.i}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq (\sigma_Z)^{(k-1).i+1}$ for $k \geq 2$. The detailed proof is presented in the extended version of this paper [10].

Since $(\overset{k.i+r}{\underset{j=0}{+}} (\sigma_Y)^j) = (\overset{k.i}{\underset{j=0}{+}} (\sigma_Y)^j) \circ (\overset{r}{\underset{j=0}{+}} (\sigma_Y)^j)$ and $(\overset{r}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq \sigma_Z^r$ (by Theorem 2(a)), we can apply Lemma 1(a) to show that $(\overset{k.i+r}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq \sigma_Z^{(k-1).i+1} \circ \sigma_Z^r = \sigma_Z^{(k.i+r-(i-1))}$. The theorem is proved by noting that $k.i + r = n$ and $i = \lfloor n/k \rfloor$.

Let $B = (V, Q, Q_0, \Upsilon)$ be a finite state transition system, and let $\{\tau_1, \ldots \tau_k\}$ be a set of extended clusters, satisfying $\bigcup_{i=1}^{k} \tau_i = \Upsilon$. From the definition of the semantics of reachability expressions, the set of reachable states of B, denoted *reach*(B), is given by $[\![*(\mathbf{T_1} + \ldots + \mathbf{T_k})]\!](Q_0)$. By Theorem 1, this is also given by $[\![*(\mathbf{T_1} ; \ldots ; \mathbf{T_k})]\!](Q_0)$. Furthermore, Theorem 2(a) guarantees that the number of image computation iterations using $*(\mathbf{T_1} ; \ldots ; \mathbf{T_k})$ never exceeds that required with $*(\mathbf{T_1} + \ldots + \mathbf{T_k})$. Therefore, if computing $[\![(\sigma_Y)^{i+1}]\!](S)$ and computing $[\![(\sigma_Z)^{i+1}]\!](S)$ (using the terminology of Theorem 2) are of comparable complexity, it is advantageous to use $*(\mathbf{T_1} ; \ldots ; \mathbf{T_k})$. This advantage is also demonstrated by our experiments, as reported in Section 3. If $(\sigma_p)^2 \sqsubseteq \sigma_p$ for all p, Theorem 2(b) improves the upper bound of Theorem 2(a) even further. Note that we can have $(\sigma_p)^2 \sqsubseteq \sigma_p$ under several circumstances, e.g. if σ_p is of the form $*\sigma_q$.

Theorem 3. *Let $B = (V, Q, Q_0, \Upsilon)$ be a finite state transition system with extended clusters $\{\tau_1, \ldots \tau_k\}$, such that $\bigcup_{i=1}^{k} \tau_i = \Upsilon$ and $\mathbf{T_k} \not\sqsubseteq *(\mathbf{T_1} + \cdots + \mathbf{T_{k-1}})$. Let σ_X denote $(\mathbf{T_1} + \cdots + \mathbf{T_{k-1}})$, and $\widehat{\sigma}$ denote $(*\sigma_X) \circ *(\mathbf{T_k} ; (*\sigma_X))$.*

(a) $[\![\widehat{\sigma}]\!](Q_0) = $ *reach*(B).
(b) Let σ be any reachability expression over $\{\tau_1, \ldots \tau_k\}$ such that $[\![\sigma]\!](Q_0) =$ reach(B). Let $N_k(\sigma, Q_0)$ denote the number of times image under $\mathbf{T_k}$ is computed until the complete set of states reachable from Q_0 is obtained during evaluation of $[\![\sigma]\!](Q_0)$. Then $N_k(\widehat{\sigma}, Q_0) \leq N_k(\sigma, Q_0) + 1$.

Proof sketch.

(a) From Lemma 2 and Theorem 1, $*(\mathbf{T_k} ; (*\sigma_X)) = *(\mathbf{T_k} ; \sigma_X) = *(\mathbf{T_k} + \sigma_X)$. Since $\delta \sqsubseteq *\sigma_X$, by composing both sides with $*(\mathbf{T_k} + \sigma_X)$ (or, equivalently with $*(\mathbf{T_k} ; (*\sigma_X))$) and by applying Lemma 1(a), we get $*(\mathbf{T_k} + \sigma_X) \sqsubseteq (*\sigma_X) \circ *(\mathbf{T_k} ; (*\sigma_X))$. Therefore, $*(\mathbf{T_k} + \sigma_X) \sqsubseteq \widehat{\sigma}$. However, $\widehat{\sigma} \sqsubseteq *(\mathbf{T_k} + \sigma_X)$. Hence, $\widehat{\sigma} = *(\mathbf{T_k} + \sigma_X)$. Since $[\![*(\mathbf{T_k} + \sigma_X)]\!](Q_0)) =$ *reach*(B), it follows that that $[\![\widehat{\sigma}]\!](Q_0) =$ *reach*(B).
(b) Let σ be an arbitrary reachability expression over $\{\tau_1, \ldots, \tau_k\}$ such that $[\![\sigma]\!](Q_0) =$ *reach*(B). The theorem is proved by showing that the computation of the set of reachable states with σ can be mimicked by $\widehat{\sigma}$ with no more than $N_k(\sigma, Q_0) + 1$ iterations of evaluation of $(\mathbf{T_k} ; (*\sigma_X))$. Details of the proof are given in the extended version of this paper [10].

Given a finite-state system, Theorem 3 gives us a reachability expression that guarantees that the number of image computations under τ_k is at worst 1 more than the minimum number needed to compute the reachable state space using any reachability expression. This is particularly useful when we have clusters with disparate image computation costs. For example, when performing reachability analysis of a network of timed automata, the discrete (or non-time-elapse) transitions of individual automata might be represented by τ_1 through τ_{k-1}, while a combined time-elapse transition for all automata might be represented by τ_k. Since clocks of all automata change synchronously, computing the image under τ_k requires synchronizing all the processes and updating the clocks of *all* automata, unlike computing the image under τ_1 through τ_{k-1}. Consequently, image computation under τ_k is expected to be more expensive (in terms of memory usage and CPU time) in general compared to image computation under τ_1 through τ_{k-1}. In such cases, it may be advantageous to minimize the number of expensive image computations by application of Theorem 3(b).

Often the set of extended clusters in a state transition system are related in such a way that starting from an initial set of states S_0, if we compute the image under a cluster σ_j, no *new* states are reached unless the image under another cluster σ_i has already been computed. As an illustration, consider a combinational circuit in which the behaviour of each gate is modeled as a finite state transition system. Suppose the circuit contains a single-input gate g_1 that is fed by another gate g_2. Suppose further that the circuit starts from a stable internal state (i.e., the output of no gate is scheduled to change). The inputs of the circuit are then changed after some delay, leading to a new state. If we compute the image of this new state under a cluster modeling the behaviour of g_1, the set of reachable states cannot change unless the image under clusters corresponding to g_2 has already been computed. The following theorem shows that such dependencies can be exploited to simplify reachability expressions.

Theorem 4. *Let $\{\sigma_1, \ldots \sigma_k\}$ be a set of extended clusters and S be a set of states satisfying the following conditions:*

C1: $(\sigma_i \circ ((*\sigma_{i+1}) \circ \cdots (*\sigma_k)) \circ \sigma_i) \sqsubseteq (\sigma_i \circ ((*\sigma_i) \circ \cdots (*\sigma_k)))$ *for all* $1 \leq i < k$.
C2: There exists m, $1 \leq m \leq k$ *such that*
  C21: $(\sigma_i \circ \sigma_j) \sqsubseteq (\sigma_i + \sigma_j)$ *for* $1 \leq i, j \leq m$ *and* $i \neq j$.
  C22: $[\![*\sigma_i]\!](S) = S$, *for all* $i > m$.

Then $[\![*(\sigma_1 ; \cdots ; \sigma_k)]\!](S) = [\![(*\sigma_1) ; \cdots ; (*\sigma_k)]\!](S)$.

Proof sketch. We first note that $[\![*(\sigma_1 ; \cdots \sigma_k)]\!](S) = [\![*((*\sigma_1) ; \cdots ; (*\sigma_k))]\!](S)$ by Lemma 2. The theorem is then proved by using induction on r to show that $[\![((*\sigma_1) ; \cdots ; (*\sigma_k))^r]\!](S) \sqsubseteq [\![(*\sigma_1) ; \cdots ; (*\sigma_k)]\!](S)$ for all $r \geq 0$. Details of the proof are given in the extended version of this paper [10].

Condition C1 in Theorem 4 formalizes an ordering of dependencies between the σ_i's. Effectively, C1 states that the effect of computing the image under expressions $\{\sigma_{i+1}, \ldots \sigma_k\}$ does not affect the computation of image under σ_i for all $1 \leq i < k$. Condition C21 asserts that the first few expressions in the above ordering do not depend on any other expressions. Hence, computing the image under the composition of two such expressions gives the same result as computing the image under the expressions individually and then taking their union. Condition C22 states an additional ordering requirement: unless the image of S under one of $\{\sigma_1, \ldots \sigma_m\}$ is computed, the reachability expressions $\{\sigma_{m+1}, \ldots \sigma_k\}$ do not result in any new states being reached. If all three conditions are satisfied, Theorem 4 permits a simplification in the computation of reachable states. In particular, it allows us to obtain the entire set of reachable states by computing the reachable states under each σ_i only once.

3 Experimental Results and Their Analysis

In the previous section, we presented theorems on reachability expressions which embody heuristic strategies for improving the efficiency of symbolic search. These include strategies such as replacing symbolic breadth-first search by round-robin search, and minimizing the number of applications of costly transitions. In order to evaluate the effectiveness of these heuristics, we have implemented an interpreter for reachability expressions in a tool called NuSMV-DP. Our tool acts as a wrapper on top of the reachability analysis engine of NuSMV [6]. It takes as inputs: (a) a description of a finite state transition system as a collection of named clusters, (b) a reachability expression, and (c) an initial set of states. Our tool explores the reachable state space according to the reachability expression and reports performance statistics on termination of the search.

Brief overview of example suite: We have used two classes of examples for our experiments – Fischer protocol and gate-level circuits with delays. Our choice of examples is motivated by their popularity in the domain of timed system analysis, and also by the ease of scaling their sizes.

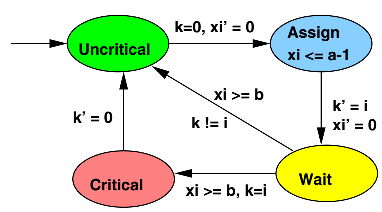

Fig. 1. Fischer's mutual exclusion protocol

Fischer protocol: This is a distributed timed protocol used to ensure mutual exclusion when a number of processes access a shared resource. Each process P_i is modeled as a timed automaton, as shown in Figure 1, where x_i is the clock of P_i and k is a shared variable for communication between processes. In Figure 1, a and b are integer constants that bound the time spent by each process in the "Assign" and "Wait" states. For an n-process Fischer protocol, a network of timed automata is obtained by asynchronous parallel composition of n automata. Details of the model can be found in our technical report [11].

A natural clustering for an n-process Fischer protocol is to have one cluster per process, containing all *discrete* or non-time-elapse guarded actions of the process. Additionally, we must have one cluster containing the guarded action representing the synchronous advancement of time for all processes.

Circuits with inertial and bi-bounded delays: Our second set of examples consists of gate-level circuits. Each gate is modeled as consisting of three parts:

– A *boolean logic block* that gives the boolean value of the output as a function of the boolean values of the inputs.

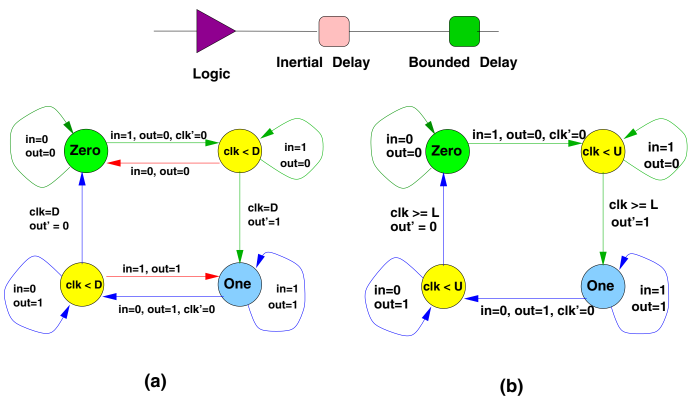

Fig. 2. (a) Inertial delay model (b) Bi-bounded pure delay model

- The output of the logic block is fed to an *inertial delay element* modeled as in Figure 2(a). If the inertial delay is D, the output of this element changes only if a change in its input persists for at least D units of time.
- The output of the inertial delay element is fed to a *bi-bounded pure delay element* which is modeled as shown in Figure 2(b). If the lower and upper bounds associated with this element are l and u respectively, it delays each transition on its input by a non-deterministic delay between l and u units.

Given an interconnection of gates representing a circuit, we compose the state transition behaviours of the logic block, inertial delay element and bi-bounded delay element of each gate to form a network of timed automata. To simplify the model, we assume that D, l and u are identical for all gates. To ensure that every pure delay element causes its output to change once between two consecutive changes at its input, we also assume that $u < D$. When the output of a gate feeds the input of another gate, we ensure during composition that the corresponding output and input transitions occur simultaneously. Time is assumed to flow synchronously for all gates.

A natural clustering for an n-gate circuit modeled as above is to have a cluster for the discrete (non-time-elapse) transitions of each logic function, inertial and bi-bounded delay element, and an additional cluster for the synchronous advancement of time for all clocks. When the output of a gate feeds the input of another gate, we must combine the corresponding guarded actions at the output and input. For our experiments, the circuit inputs are modeled as signals that non-deterministically change their boolean values after a predefined delay Δ_{in}. The exact circuits used in our experiments as well as details about our model can be found in [11]. For both classes of examples, we assume that time is discrete, and model the timed behaviour using bounded-counter automata.

Performance comparisons: For the examples described above, let $\{\tau_1, \ldots, \tau_k\}$ be the set of extended clusters representing non-time-elapse transitions, and let

τ_t be the cluster representing synchronous advancement of time. Let Γ be the monolithic transition relation obtained by combining all transitions into a single cluster. Then, the reachability expression $S_0 = *\mathbf{\Gamma}$ mimics symbolic breadth-first search using this monolithic transition relation, as in the original NuSMV tool. Γ can also be disjunctively partitioned into its component clusters and the image computed using the reachability expression $S_1 = *(\mathbf{T_1} + \cdots + \mathbf{T_k} + \mathbf{T_t})$. While this reduces the effort for each image computation, the number of image computations increases significantly. To control this, we apply Theorem 2, and consider the reachability expression $S_2 = *(\mathbf{T_1} \,;\, \cdots \,;\, \mathbf{T_k} \,;\, \mathbf{T_t})$ instead.

We have experimentally profiled the performance of reachability analysis using the expressions S_0, S_1 and S_2. All our experiments were run on a 3 GHz Intel Pentium 686 processor with 1 GB of main memory, and running Fedora Core Linux 3.4.3-6.fc3.

For the Fischer protocol examples, we computed the set of backward reachable states starting from a set of states in which mutual exclusion is violated. For simplicity, the parameters a and b were set to 1 and 2 respectively, for all processes. The results are shown as bar graphs in Figures 3 and 4. The total number of image computation iterations needed to compute the reachable states using S_0, S_1 and S_2 respectively are shown as triples within parentheses along the abscissa in Figure 3. The missing data corresponds to experiments that did not terminate in 30 minutes. For the circuit examples, we computed the set of forward reachable states starting from a given set of initial states. The results are shown as bar graphs in Figures 5 and 6. In these figures, bar graphs corresponding to experiments on the same circuit but with different values of the parameters l, u, D and Δ_{in} have been grouped together. The total number of image computation iterations needed to compute the reachable states using S_1 and S_2 respectively are shown as comma-separated pairs along the abscissa in Figure 5. For each circuit with r different combinations of l, u, D and Δ_{in} (r ranges from 2 to 4 in our experiments), there are r sets of bar graphs and r lines of comma-separated pairs above the circuit's name. The i^{th} pair from the top and the i^{th} set of bar graphs from the left represent data obtained with the same set of parameters for a given circuit. The number of iterations using S_0 and S_1 were identical for all our circuit experiments. Details of the parameter values used for each circuit are available in a detailed version of this paper [10].

In Figures 3 and 5, "Time (s)" denotes the time in seconds to compute the reachable state space. In Figures 4 and 6, "max BDD" denotes the maximum number of BDD nodes required to store the (partially computed) state space at any time during the state space search.

It can be seen that unguided disjunctive partitioning of the transition relation, as in S_1, results in worse performance than reachability search using a monolithic transition relation. In the absence of guidance, disjunctive partitioning is, therefore, not an effective strategy. Theorem 2 guarantees that S_2 requires no more iterations of image computation than S_1. This is clearly seen in the iteration counts in Figures 3 and 5. A reduction in CPU time is expected from the combined effect of fewer iterations and operations on smaller BDDs. In the

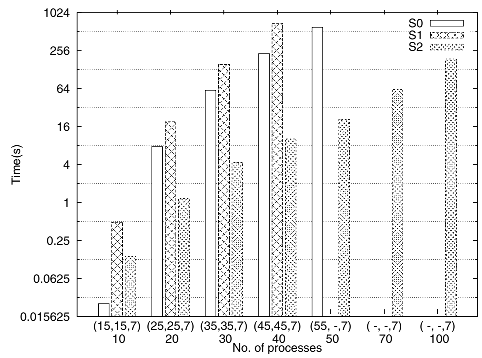

Fig. 3. Analysis of Fischer processes: Time and iteration counts for S_0, S_1, S_2

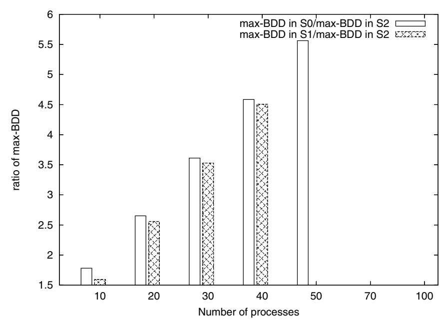

Fig. 4. Analysis of Fischer processes: Ratios of "max BDD"

Fischer protocol examples, S_2 has the best performance. The only exception is the example with 10 processes. Here, the BDDs are small even when using S_0, and no significant gains are obtained by decomposing the transition relation. Instead, iterating through the clusters incurs time overhead when using S_2. For circuits, the ratios in Figure 6 are always greater than 1. Thus, S_2 results in the minimum "max BDD" value. However, as seen in Figure 5, circuits 6, 7 and 8 show better performance using S_0 with respect to time. These circuits were found to have very low "max BDD" values compared to the largest transition cluster size. This is in contrast to circuits 2, 3 and 4 where this ratio was much higher. Therefore, unlike in circuits 2, 3 and 4, BDD sizes of partially computed state sets do not significantly influence the performance of reachability analysis in circuits 6, 7 and 8. Since the largest transition cluster size is large compared to "max BDD", reducing the total number of image computation iterations gives better performance. Thus S_0 performs better than S_2 for circuits 6, 7 and 8.

The BDD representation of the cluster "τ_t" is usually larger than that of other τ_i's since the transitions in τ_t involve clock variables of *all* processes. These large BDDs, in turn, lead to higher costs for computing the image under τ_t vis-a-vis the cost of computing the image under a τ_i. We have seen above that Theorem 3

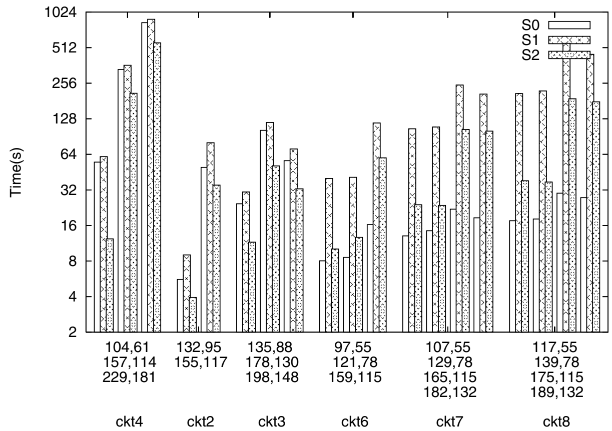

Fig. 5. Analysis of circuits: Time and iteration counts for S_1, S_2. Iteration counts for S_0 are identical to those for S_1.

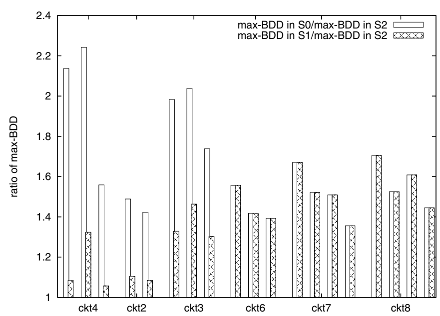

Fig. 6. Analysis of circuits: Ratios of "max BDD"

gives us a way to reduce the number of costly image computations, potentially leading to performance improvements. To validate this experimentally, we measured and recorded the performance of computing the reachable state set using a reachability expression obtained from Theorem 3. In the Fischer protocol examples, the size of the BDD representation of τ_t is comparable to that of the other τ_i clusters. Hence, the effect of minimizing applications of τ_t does not produce a significant performance difference for the Fischer examples, and we report results for only the circuits. Let $\sigma_x = (\mathbf{T_1} \,;\, \cdots \,;\, \mathbf{T_k})$ and let $S_3 = (*\sigma_x) \circ *(\mathbf{T_t} \,;\, (*\sigma_x))$. As seen in Theorem 3, S_3 minimizes the number of image computations under τ_t (up to 1 additional computation). We now compare its performance with an equivalent schedule S_4, which is defined as: $S_4 = *(\mathbf{T_1} \,;\, \cdots \,;\, \mathbf{T_k} \,;\, *\mathbf{T_t})$.

Figure 7 shows the ratios of "max BDD" using S_4 to that using S_3 for the circuit examples. For each circuit, we used different sets of delay parameters, as in the earlier experiments. Within the set of experiments for each circuit, the

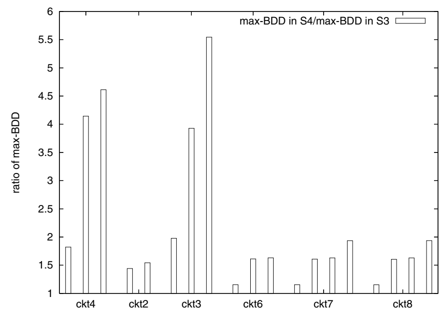

Fig. 7. Analysis of circuits: Ratios of "max BDD" using S_4 to that using S_3

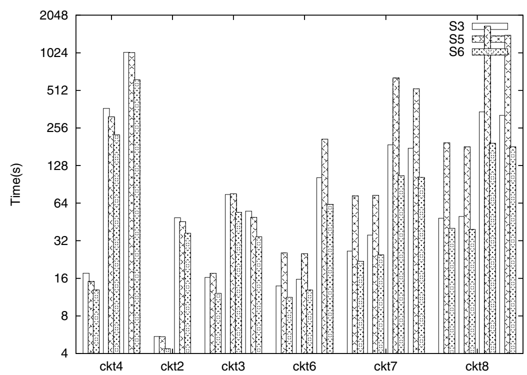

Fig. 8. Analysis of circuits: Time for schedules S_3, S_5 and S_6

number of image computations under τ_t until all reachable states are computed, increases from left to right for both S_4 and S_3. As the number of computations increases, the effect of minimizing applications of τ_t becomes more pronounced, as can be seen from the rising ratio of "max BDD" using S_4 to that using S_3.

Theorem 4 relates to the effect of applying a sequence of clusters consistently with the topological dependencies between them. To evaluate the effectiveness of this Theorem, we performed an additional set of experiments with the circuit examples. In these examples, one can obtain a topological ordering of the gates and circuit elements from the inputs to the outputs. When computing this order, a sub-circuit with a loop must be considered as a single circuit element without exposing the loop. Since the output of each gate/circuit element is fed to a gate or element with a higher topological index, there is an ordering of dependencies between the non-time-elapse clusters τ_i. By choosing the initial state S to be such that all gates are stable (i.e. no gate is scheduled to change its output), we ensure that condition C22 required in Theorem 4 is satisfied.

Let the input-to-output topological ordering of the non-time-elapse clusters be $\tau_1 < \cdots < \tau_k$. Let $\sigma_x = (\mathbf{T_1} ; \cdots ; \mathbf{T_k})$ as before, and let $\sigma_y = (\mathbf{T_k} ; \cdots ; \mathbf{T_1})$

compute images of the clusters in reverse topological order. Moreover, let $\sigma_z = (*\mathbf{T_1}\,;\cdots;*\mathbf{T_k})$. Then, by Theorem 1, we have $*\sigma_x = *\sigma_y$ and by Theorem 4, we have $*\sigma_x = \sigma_z$. Earlier, we have considered the reachability expression $S_3 = (*\sigma_x) \circ *(\mathbf{T_t}\,; (*\sigma_x))$ that minimizes (up to 1 additional computation) the number of image computations under time transition cluster τ_t. We now consider the reachability expressions $S_5 = (*\sigma_y) \circ *(\mathbf{T_t}\,; (*\sigma_y))$ and $S_6 = \sigma_z \circ *(\mathbf{T_t}\,; \sigma_z)$. Using the above mentioned identities, it is easy to see that that $S_3 = S_5 = S_6$, i.e. all three reachability expressions compute the same set of reachable states.

Using S_3, S_5, and S_6 in the circuit examples, "max BDD" was found to be nearly identical. However, the CPU times were different because of additional fix-point checks in S_5 and S_3 compared to those in S_6. In Figure 8, the circuits are arranged from left to right in order of increasing topological depth. The increased number of fix-point checks due to an increase in the number of ordered clusters results in the increased time difference between schedules S_5 and S_3 from circuit 4 to 8 as seen in Figure 8. For circuits with short topological depth, the performance of S_5 and S_3 are similar. However, as the topological depth increases, computing images in topologically sorted order leads to significant improvements compared to computing in reverse topological order. Furthermore, S_6 improves over S_3, albeit marginally, in most cases. This is because all clusters τ_i other than the time transition cluster τ_t are self-disabling; hence computing the image under $\mathbf{T_i}$ and $*\mathbf{T_i}$ require similar computational effort.

4 Discussion and Conclusion

Reachability expressions give the user the ability to specify the heuristics of symbolic state space search. Semantically equivalent reachability expressions can have radically different costs of computation. In this paper, we presented a theory to reason about the equivalence and relative performance of alternative reachability expressions, and validated our predictions with experiments.

Our experimental investigations indicate that when the absolute size of the BDD representation of a monolithic transition relation is small, it is advantageous to perform classical symbolic breadth-first search using the monolithic relation directly. Similarly, when the maximum BDD size encountered in representing (partially) computed state sets is small compared to the BDD size of the monolithic transition relation or the BDD size of the largest transition cluster, it helps to minimize the number of image computation iterations by using classical symbolic breadth-first search. However, if the maximum BDD size for representing (partially) computed state sets is large and there is a dominant cluster represented as a large BDD, it is advantageous to adopt a round-robin scheduling of clusters in a way that minimizes the image computations under the dominant cluster. Further, in round-robin scheduling, ordering the clusters in topological order is effective in examples such as deep circuits, which have considerable forward propagation of events.

The experiments and conclusions reported in this paper are promising, although preliminary. Much more data on a wider set of examples is needed before a comprehensive evaluation of the effectiveness of reachability expressions in improving the performance of state space search can be done. However, we believe that reachability expressions will serve as a useful addition to the existing repository of tools and techniques for making symbolic reachability analyzers more efficient.

Acknowledgments. We thank Varun Kanade for help with the experiments.

References

1. D. Beyer, C. Lewerentz, and A. Noack. Rabbit: A tool for BDD-based verification of real-time systems. In *Proceedings of CAV*, LNCS 2725, pages 122–125, 2003.
2. R. Bloem, K. Ravi, and F. Somenzi. Symbolic guided search for CTL model checking. In *Proceeding of ACM/IEEE DAC*, pages 29–34, 2000.
3. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.
4. P. Chauhan, E. M. Clarke, S. Jha, J. Kukula, T. Shiple, H. Veith, and D. Wang. Non-linear quantification scheduling in image computation. In *Proceedings of ACM/IEEE ICCAD*, pages 293–298, 2001.
5. P. Chauhan, E. M. Clarke, S. Jha, J. Kukula, H. Veith, and D. Wang. Using combinatorial optimization methods for quantification scheduling. In *Proceedings of CHARME*, LNCS 2144, pages 293–309, 2001.
6. A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV version 2: An opensource tool for symbolic model checkin. In *Proceedings of CAV*, LNCS 2404, pages 359–364, 2002.
7. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. In *Proceedings of LICS*, pages 1–33, 1990.
8. A. Narayan, A. J. Isles, J. Jain, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Reachability analysis using partitioned-robdds. In *Proceedings of ACM/IEEE ICCAD*, pages 388–393, 1997.
9. P. K. Pandya and M. Raut. A kleene algebra of reachability expressions and its use in efficient symbolic guided search. Technical Report (in preparation), STCS, Tata Institute of Fundamental Research, 2005.
10. D. Thomas, S. Chakraborty, and P. K. Pandya. Efficient guided symbolic reachability using reachability expressions. Technical Report TR-06-1 (http://www.cfdvs.iitb.ac.in/reports/techrep06.php3), CFDVS, IIT Bombay, January 2006.
11. D. Thomas, P. K. Pandya, and S. Chakraborty. Scheduling clusters in model checking of real time systems. Technical Report TR-04-16 (http://www.cfdvs.iitb.ac.in/reports/techrep04.php3), CFDVS, IIT Bombay, September 2004.