Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

Efficient Guided Symbolic Reachability using Reachability Expressions

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

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

Received: date / Revised version: date

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 NuSMVDP 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 [12] has emerged as an important technique for program verification and for finding deep logical bugs in reactive systems. In this technique, programs are modeled as finite state transition systems and the state space of the model is searched for undesirable behaviour. BDD-based [3] symbolic search techniques, e.g. those used in NuSMV [11], 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.

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 [13], and by determining good quantification schedules for conjunctive partitioning [6,7]. Yet other techniques include guided search where hints [2] are used to direct initial parts of the search, iterative squaring [4,5] where the number of image computation steps is reduced exponentially at the cost of transition relation squaring, chaining techniques [14] and saturation-based methods [9] that guide the search to use certain clusters of "local" transitions before others, so that peak memory requirements and fixpoint computation iterations are reduced.

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, chaining etc. They also include familiar operations like sequential composition, union and Kleene closure. Starting from a set of initial states, different reachability expressions compute different sets of reachable states, in general. However, when multiple expressions encode the same predicate transformer, although they compute the same set of states, the computations may vary widely in performance. Importantly, reachability expressions allow us to encode more efficient ways of computing the complete set of reachable states than symbolic breadth-first search. They also provide a theoretical framework to understand why techniques like saturation and chaining, among others, work well in practice.

We have implemented an interpreter for reachability expressions in a tool called NuSMVDP[1] that works as a wrapper on top of the reachability engine of NuSMV [11]. 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 state predicate using our BDD-based NuSMVDP tool can vary significantly. This highlights the need to study properties of reachability expressions, and to correlate these properties with experimental data. Towards this end, we discuss some equivalence transformations that improve the efficiency of evaluating reachability expressions on finite state systems, and report experiments using unoptimized and multiple variants of optimized reachability expressions.

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 [16]. We experimentally evaluate the performance of BDD-based symbolic reachability analysis on these examples using multiple equivalent reachability expressions. A previous technical report [16] gives details of experiments carried out using NuSMVDP with diverse problems such as analyzing 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 techniques vis-a-vis classical breadth-first search. For example, we have been able to model check the Fischer protocol with 100 processes by guiding the search with specific reachability expressions discussed in Section 3, 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 NuSMVDP tool. Finally, we conclude the paper in Section 4.

[1] DP stands for "Disjunctive Partitioning" for historical reasons, although the tool can interpret arbitrary reachability expressions.

2 A Theory of Reachability Expressions

We consider a (possibly infinite) 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 subset 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 the next state.

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 τ defines a relation, R_τ, on the set Q of states. We say that $(s_1, s_2) \in R_\tau$ iff there exists a guarded action $(G \mapsto A) \in \tau$ 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 τ and a set of states $S\,(\subseteq Q)$, the *image* of S under τ is defined as $Im_\tau(S) = \{s \mid \exists s' \in S,\, R_\tau(s', s)\}$. It is easy to see that $Im_\tau : 2^Q \rightarrow 2^Q$ is a monotone function with $Im_\tau(\emptyset) = \emptyset$.

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 extended clusters $\tau_1, \ldots, \tau_k$ in B. This is done to distinguish between extended clusters and reachability expressions, in the same spirit that letters in an alphabet are distinguished from regular expressions over the alphabet. We will use the above 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. For clarity of notation, we will henceforth omit the subscript B when it is clear from the context.

- $[\![\,\mathbf{T_i}\,]\!](S) = Im_{\tau_i}(S)$, for all $\tau_i \in \mathcal{T}$
- $[\![\,\sigma_1 + \sigma_2\,]\!](S) = [\![\,\sigma_1\,]\!](S) \cup [\![\,\sigma_2\,]\!](S)$
- $[\![\,\sigma_1 \circ \sigma_2\,]\!](S) = [\![\,\sigma_2\,]\!]([\![\,\sigma_1\,]\!](S))$
- $[\![\,\sigma_1 \,;\, \sigma_2\,]\!](S) = [\![\,(\sigma_1 + \delta) \circ (\sigma_2 + \delta)\,]\!](S)$
- $[\![\,(\sigma)\,]\!](S) = [\![\,\sigma\,]\!](S)$
- $[\![\,\sigma^0\,]\!](S) = S$ and $[\![\,\sigma^{i+1}\,]\!](S) = [\![\,\sigma \circ \sigma^i\,]\!](S)$ for all $i \geq 0$.
- $[\![\,*\sigma\,]\!](S) = \bigcup_{i=0}^{\infty} [\![\,\sigma^i\,]\!](S)$

Although "$;$" is seen to be a derived operator, we retain it for notational convenience. Also note from the above definitions that $[\![\,\sigma^0\,]\!](S) = [\![\,\delta\,]\!](S)$ for all S.

2.2 Reachability expressions and regular expressions

A reachability expression over $\mathcal{T} = \{\tau_1, \ldots \tau_k\}$ can also be viewed as a regular expression over the alphabet $\{\mathbf{T_1}, \ldots, \mathbf{T_k}\}$, with $+$ as the union operator, $\circ$ as the concatenation operator and $*$ as the Kleene star operator. This analogy is best understood if one considers a reachability expression as specifying a set of finite image computation sequences. An image computation sequence is a word over the alphabet $\{\mathbf{T_1}, \ldots, \mathbf{T_k}\}$. Each such finite sequence $\mathbf{T_i} \circ \cdots \circ \mathbf{T_j}$ defines a predicate transformer that takes a state predicate S and returns $Im_{\tau_j}(\cdots Im_{\tau_i}(S)) = [\![\,\mathbf{T_i} \circ \cdots \circ \mathbf{T_j}\,]\!](S)$. A reachability expression, viewed as a regular expression, specifies a regular set of such finite sequences or words. The effect of applying a reachability expression σ to a set of states S, or $[\![\,\sigma\,]\!](S)$, is then obtained by transforming S by each image computation sequence in the regular set represented by σ, and accumulating them. It can be easily shown that the semantics of reachability expressions as defined in Section 2.1 coincides with the above interpretation. As we will see later, Θ is the identity for the $+$ operator and annihilator for the $\circ$ operator, while δ is the identity for the $\circ$ operator. Viewed as regular expressions, Θ therefore plays the role of the empty language, and δ plays the role of the language consisting of only the empty word.

Lemma 1. *Let σ_1 and σ_2 be reachability expressions. σ_1 is semantically equivalent to σ_2 iff the regular expressions σ_1 and σ_2 represent the same language.*

Proof. Let σ_1 and σ_2 be reachability expressions. If the corresponding regular expressions represent the same language, then every image computation sequence used to define $[\![\,\sigma_1\,]\!]$ is also used to define $[\![\,\sigma_2\,]\!]$ and vice versa. Therefore, the reachability expressions are semantically equivalent.

Now consider the case when the regular expressions do not represent the same language. Without loss of generality, let α be a word in the language represented by σ_1 but not in the language represented by σ_2. Let $\alpha = \beta_0 \circ \beta_1 \circ \cdots \circ \beta_n$, where $\beta_i \in \{\mathbf{T_1}, \cdots, \mathbf{T_k}\}$ for $0 \leq i \leq n$. To show that $[\![\,\sigma_1\,]\!]$ differs from $[\![\,\sigma_2\,]\!]$, we construct a state transition system B with $m = \lceil \log(n+2) \rceil$ binary valued variables $b_1.b_2 \ldots b_m$ and 2^m states such that $[\![\,\sigma_1\,]\!]_B \neq [\![\,\sigma_2\,]\!]_B$. Let $\{S_0, S_1, \cdots, S_{2^m-1}\}$ be the set of states of B such that B is in state S_i if the binary values of $b_1.b_2 \ldots b_m$ encode the integer i. Using the structure of the word α, we now define guarded actions such that the only way to reach S_{n+1} from S_0 is by using the unique sequence of image computations corresponding to the word α.

Let A denote the multiple assignment statement: $b'_1 = \neg b_1$ and $b'_i = b_i \oplus (b_1 \wedge b_2 \wedge \cdots \wedge b_{i-1})$ for $2 \leq i \leq m$. Then action A takes state S_i to state $S_{(i+1) \bmod 2^m}$ for $0 \leq i \leq (2^m - 1)$. Let E_i be a boolean expression on $b_1, \cdots, b_m$ that evaluates to True iff the values of $b_1, \cdots, b_m$ encode the integer i. Thus, E_i precisely describes state S_i for $0 \leq i \leq 2^m - 1$. Suppose $\mathbf{T_1}$ occurs at positions $i_1, i_2, \cdots, i_p$ in the word α, i.e., $\beta_{i_1} = \beta_{i_2} = \cdots = \beta_{i_p} = \mathbf{T_1}$. We define a guard $G_1 = E_{i_1} \vee E_{i_2} \vee \cdots \vee E_{i_p}$. Similarly, we define guards G_j for every $\mathbf{T_j}$ that appears in α. For all $\mathbf{T_j}$'s that do not appear in α, we define $G_j = \mathsf{False}$. Finally, the transition clusters for B are defined as $\tau_i = \{(G_i \mapsto A)\}$ for all i in 1 through k.

It is not hard to see that with the above definition of τ_i's, the only sequence of image computations that can take B from S_0 to S_{n+1} is the one represented by the word α. Since α is in the language represented by σ_1 but not in the language represented by σ_2, we have $S_{n+1} \in [\![\,\sigma_1\,]\!]_B(S_0)$ and $S_{n+1} \notin [\![\,\sigma_2\,]\!]_B(S_0)$. Hence σ_1 is not semantically equivalent to σ_2. This shows that reachability expressions σ_1 and σ_2 are equivalent iff the corresponding regular expressions represent the same language.

It follows from Lemma 1 that semantic equivalence checking of reachability expressions is no harder than language equivalence checking of the corresponding regular expressions. Using the standard construction of nondeterministic finite automata (NFA) from regular expressions, determinization of NFAs, and language equivalence checking of deterministic finite automata (DFA), one can effectively check equivalence of two reachability expressions. Since determinizing an NFA involves an exponential blowup in the worst case, checking the equivalence of reachability expressions σ_1 and σ_2 can be done in $O(2^{O(|\sigma_1|+|\sigma_2|)})$ time.

Though reachability expressions are syntactically similar to regular expressions, semantically they define how sets of states are transformed in an underlying state transition system. Thus, there are two ways of reasoning about reachability expressions: by viewing them simply as languages without reference to an underlying state transition system, or by reasoning about them as state predicate transformers in an underlying state transition system. While some results on reachability expressions discussed in this paper can also be proved by treating reachability expressions as regular expressions, we present arguments that use the notion of states and transitions wherever possible, to give a better picture of how image computation steps are applied during reachability analysis. Note that the complementation operation for regular languages does not give anything meaningful when applied to the language of image computation sequences represented by a reachability expression σ. Indeed, if α is an image computation sequence *not* originially present in the language represented by a reachability expression σ, applying α on a set of states S can, in general, lead to states both outside $[\![\,\sigma\,]\!]\,(S)$ as well in $[\![\,\sigma\,]\!]\,(S)$.

2.3 Properties of Reachability Expressions

We now discuss some interesting properties of reachability expressions. These properties eventually help us to understand why some expressions lead to more efficient search strategies than others in practice.

Lemma 2. *Let σ be an arbitrary reachability expression and $B = (V, Q, Q_0, \Upsilon)$ be a state transition system. Then (i) $[\![\,\sigma\,]\!]\,(\emptyset) = \emptyset$, and (ii) $[\![\,\sigma\,]\!]\,(S_1 \cup S_2) = [\![\,\sigma\,]\!]\,(S_1) \cup [\![\,\sigma\,]\!]\,(S_2)$ for all $S_1, S_2 \subseteq Q$.*

Proof. We use induction on the structure of σ.
Basis: If $\sigma = \mathbf{T_i}$, both parts of the lemma hold from the definition of Im_{T_i}. If $\sigma = \delta$, $[\![\,\delta\,]\!]\,(\emptyset) = \emptyset$ by definition of δ, and $[\![\,\delta\,]\!]\,(S_1 \cup S_2) = S_1 \cup S_2 = [\![\,\delta\,]\!]\,(S_1) \cup [\![\,\delta\,]\!]\,(S_2)$. If $\sigma = \varTheta$, $[\![\,\sigma\,]\!]\,(S) = \emptyset$ for all $S \subseteq Q$; hence both parts of the lemma hold.
Hypothesis: Let σ_1 and σ_2 be reachability expressions satisfying both parts of the lemma.
Induction step: We consider the following cases:

1. $\sigma = \sigma_1 + \sigma_2$:
 (i) Emptiness of $[\![\,\sigma_1 + \sigma_2\,]\!]\,(\emptyset)$ follows from the definition of the semantics and from the induction hypothesis.
 (ii) By definition, $[\![\,\sigma_1 + \sigma_2\,]\!]\,(S_1 \cup S_2) = [\![\,\sigma_1\,]\!]\,(S_1 \cup S_2) \cup [\![\,\sigma_2\,]\!]\,(S_1 \cup S_2)$. By induction hypothesis, the latter is equal to $[\![\,\sigma_1\,]\!]\,(S_1) \cup [\![\,\sigma_1\,]\!]\,(S_2) \cup [\![\,\sigma_2\,]\!]\,(S_1) \cup [\![\,\sigma_2\,]\!]\,(S_2)$. By associativity and commutativity of set union, the above is equal to $(\,[\![\,\sigma_1\,]\!]\,(S_1) \cup [\![\,\sigma_2\,]\!]\,(S_1)) \cup (\,[\![\,\sigma_1\,]\!]\,(S_2) \cup [\![\,\sigma_2\,]\!]\,(S_2))$. By the semantics defined above, this is equal to $[\![\,\sigma_1 + \sigma_2\,]\!]\,(S_1) \cup [\![\,\sigma_1 + \sigma_2\,]\!]\,(S_2)$.
2. $\sigma = \sigma_1 \circ \sigma_2$:
 (i) By definition, $[\![\,\sigma\,]\!]\,(\emptyset) = [\![\,\sigma_2\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(\emptyset))$. It therefore follows from the induction hypothesis that this set is empty.
 (ii) To evaluate $[\![\,\sigma_1 \circ \sigma_2\,]\!]\,(S_1 \cup S_2)$, we notice from the definition of the semantics that this is $[\![\,\sigma_2\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S_1 \cup S_2))$. By the induction hypothesis for σ_1, this is equal to $[\![\,\sigma_2\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S_1) \cup [\![\,\sigma_1\,]\!]\,(S_2))$. By the hypothesis for σ_2, this can be further written as $[\![\,\sigma_2\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S_1)) \cup [\![\,\sigma_2\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S_2))$. By the definition of the semantics, this equals $[\![\,\sigma_1 \circ \sigma_2\,]\!]\,(S_1) \cup [\![\,\sigma_1 \circ \sigma_2\,]\!]\,(S_2)$.
3. $\sigma = \sigma_1\,;\,\sigma_2$: Since $[\![\,\sigma\,]\!] = [\![\,(\sigma_1 + \delta) \circ (\sigma_2 + \delta)\,]\!]$, both parts of the lemma are easily proven from the induction hypothesis and from the two cases proved above.
4. $\sigma = (\sigma_1)$: The proof follows from the induction hypothesis and from the definition of $[\![\,(\sigma_1)\,]\!]$.
5. $\sigma = *\sigma_1$: We first show by induction on i that $[\![\,\sigma_1^i\,]\!]\,(S_1 \cup S_2) = [\![\,\sigma_1^i\,]\!]\,(S_1) \cup [\![\,\sigma_1^i\,]\!]\,(S_2)$ and $[\![\,\sigma_1^i\,]\!]\,(\emptyset)$ is empty for all $i \geq 0$.
   - *Basis:* If $i = 0$, $\sigma_1^i = \delta$ by definition, and we have already argued above that δ satisfies both parts of the lemma.
   - *Hypothesis:* Assume that $[\![\,\sigma_1^i\,]\!]$ satisfies both parts of the lemma for all i such that $0 \leq i \leq k$.
   - *Induction step:* By definition, $\sigma_1^{k+1} = \sigma_1 \circ \sigma_1^k$. By induction hypotheses, both $[\![\,\sigma_1\,]\!]$ and $[\![\,\sigma_1^k\,]\!]$ satisfy the two parts of the lemma. Therefore, by case 2 proved above, $[\![\,\sigma_1^{k+1}\,]\!]$ also satisfies both parts of the lemma. Hence, $[\![\,\sigma_1^i\,]\!]\,(S_1 \cup S_2) = [\![\,\sigma_1^i\,]\!]\,(S_1) \cup [\![\,\sigma_1^i\,]\!]\,(S_2)$, and $[\![\,\sigma_1^i\,]\!]\,(\emptyset) = \emptyset$ for all $i \geq 0$.

 We now prove the two parts of the lemma for $\sigma = *\sigma_1$.
 (i) Since $[\![\,\sigma_1^i\,]\!]\,(\emptyset) = \emptyset$ for all $i \geq 0$, it follows from the definition of the semantics that $[\![\,*\sigma_1\,]\!]\,(\emptyset) = \bigcup_{i=0}^{\infty} [\![\,\sigma_1^i\,]\!]\,(\emptyset) = \emptyset$.
 (ii) Let us assume that $q \in [\![\,*\sigma_1\,]\!]\,(S_1 \cup S_2)$. By definition of the semantics, there exists an integer $r\,(\geq 0)$ such that $q \in [\![\,\sigma_1^r\,]\!]\,(S_1 \cup S_2)$. We have already seen above that $[\![\,\sigma_1^i\,]\!]\,(S_1 \cup S_2) = [\![\,\sigma_1^i\,]\!]\,(S_1) \cup [\![\,\sigma_1^i\,]\!]\,(S_2)$ for all $i \geq 0$. Therefore, $q \in [\![\,\sigma_1^r\,]\!]\,(S_1) \cup [\![\,\sigma_1^r\,]\!]\,(S_2)$, and hence $q \in [\![\,*\sigma_1\,]\!]\,(S_1) \cup [\![\,*\sigma_1\,]\!]\,(S_2)$. This shows that $[\![\,*\sigma_1\,]\!]\,(S_1 \cup S_2) \subseteq [\![\,*\sigma_1\,]\!]\,(S_1) \cup [\![\,*\sigma_1\,]\!]\,(S_2)$. Similarly, if $q \in [\![\,*\sigma_1\,]\!]\,(S_1) \cup [\![\,*\sigma_1\,]\!]\,(S_2)$, there exists an integer $p\,(\geq 0)$ such that either $q \in [\![\,\sigma_1^p\,]\!]\,(S_1)$ or $q \in [\![\,\sigma_1^p\,]\!]\,(S_2)$. Since we have shown above that $[\![\,\sigma_1^i\,]\!]\,(S_1 \cup S_2) = [\![\,\sigma_1^i\,]\!]\,(S_1) \cup [\![\,\sigma_1^i\,]\!]\,(S_2)$ for all $i \geq 0$, this implies that $q \in [\![\,\sigma_1^p\,]\!]\,(S_1 \cup S_2)$ and hence $q \in [\![\,*\sigma_1\,]\!]\,(S_1 \cup S_2)$. It therefore follows that $[\![\,*\sigma_1\,]\!]\,(S_1) \cup [\![\,*\sigma_1\,]\!]\,(S_2) \subseteq [\![\,*\sigma_1\,]\!]\,(S_1 \cup S_2)$, which along with

our earlier result implies $[\![\, *\sigma_1 \,]\!]\,(S_1 \cup S_2) = [\![\, *\sigma_1 \,]\!]\,(S_1) \cup [\![\, *\sigma_1 \,]\!]\,(S_2)$.

Corollary 1. *For all reachability expressions σ, $[\![\,\sigma\,]\!] : 2^Q \rightarrow 2^Q$ is a monotone function.*

Proof. Let $S_1 \subseteq S_2 \subseteq Q$ and let $S_3 = S_2 \setminus S_1$. Then $S_2 = S_1 \cup S_3$. By Lemma 2, $[\![\,\sigma\,]\!]\,(S_2) = [\![\,\sigma\,]\!]\,(S_1) \cup [\![\,\sigma\,]\!]\,(S_3)$. Hence, $[\![\,\sigma\,]\!]\,(S_1) \subseteq [\![\,\sigma\,]\!]\,(S_2)$.

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 $\sigma_1 \sqsubseteq \sigma_2$. It is easy to see that $\sqsubseteq$ is a reflexive, antisymmetric and transitive relation on reachability expressions. We say that $\sigma_1 = \sigma_2$, iff $\sigma_1 \sqsubseteq \sigma_2$ and $\sigma_2 \sqsubseteq \sigma_1$. Clearly, $\sigma_1 = \sigma_2$ iff they are semantically equivalent.
The following properties of reachability expressions follow from the definition of their semantics and from basic set theory. In the following, σ, σ_1, σ_2, σ_3 refer to reachability expressions over a set of extended clusters, and S refers to a set of states ($S \subseteq Q$).

P1: $(\sigma + \sigma) = \sigma$.
By definition, $[\![\,(\sigma+\sigma)\,]\!]\,(S) = [\![\,\sigma\,]\!]\,(S) \cup [\![\,\sigma\,]\!]\,(S) = [\![\,\sigma\,]\!]\,(S)$.

P2: $(\sigma_1 + \sigma_2) = (\sigma_2 + \sigma_1)$.
This follows from the definition of the semantics and commutativity of set union.

P3: $(\sigma_1 + (\sigma_2 + \sigma_3)) = ((\sigma_1 + \sigma_2) + \sigma_3)$.
This follows from the definition of the semantics and associativity of set union.

P4: $(\sigma + \Theta) = \sigma$.
By definition, $[\![\,(\sigma + \Theta)\,]\!]\,(S) = [\![\,\sigma\,]\!]\,(S) \cup [\![\,\Theta\,]\!]\,(S) = [\![\,\sigma\,]\!]\,(S) \cup \emptyset = [\![\,\sigma\,]\!]\,(S)$.

P5: $(\sigma_1 \circ (\sigma_2 \circ \sigma_3)) = ((\sigma_1 \circ \sigma_2) \circ \sigma_3)$.
By definition, $[\![\,(\sigma_1 \circ (\sigma_2 \circ \sigma_3))\,]\!]\,(S) = [\![\,(\sigma_2 \circ \sigma_3)\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S)) = [\![\,\sigma_3\,]\!]\,(\,[\![\,\sigma_2\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S)))$. Similarly, $[\![\,((\sigma_1 \circ \sigma_2) \circ \sigma_3)\,]\!]\,(S) = [\![\,\sigma_3\,]\!]\,(\,[\![\,(\sigma_1 \circ \sigma_2)\,]\!]\,(S)) = [\![\,\sigma_3\,]\!]\,(\,[\![\,\sigma_2\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S)))$.

P6: $(\sigma \circ \delta) = (\delta \circ \sigma) = \sigma$.
By definition, $[\![\,(\sigma \circ \delta)\,]\!]\,(S) = [\![\,\delta\,]\!]\,(\,[\![\,\sigma\,]\!]\,(S))$. From the definition of δ, $[\![\,\delta\,]\!]\,(\,[\![\,\sigma\,]\!]\,(S)) = [\![\,\sigma\,]\!]\,(S)$. Similarly, $[\![\,(\delta \circ \sigma)\,]\!]\,(S) = [\![\,\sigma\,]\!]\,(\,[\![\,\delta\,]\!]\,(S)) = [\![\,\sigma\,]\!]\,(S)$.

P7: $(\sigma \circ \Theta) = (\Theta \circ \sigma) = \Theta$.
By definition of Θ, $[\![\,\Theta\,]\!]\,(S) = \emptyset$ for all S. Therefore $[\![\,(\sigma \circ \Theta)\,]\!]\,(S) = [\![\,\Theta\,]\!]\,(\,[\![\,\sigma\,]\!]\,(S)) = \emptyset$. Similarly, $[\![\,(\Theta \circ \sigma)\,]\!]\,(S) = [\![\,\sigma\,]\!]\,(\,[\![\,\Theta\,]\!]\,(S)) = [\![\,\sigma\,]\!]\,(\emptyset)$. By Lemma 2, $[\![\,\sigma\,]\!]\,(\emptyset) = \emptyset$, which is equal to $[\![\,\Theta\,]\!]\,(S)$.

P8: $(\sigma_1 \circ (\sigma_2 + \sigma_3)) = ((\sigma_1 \circ \sigma_2) + (\sigma_1 \circ \sigma_3))$.
By definition of the semantics, $[\![\,(\sigma_1 \circ (\sigma_2 + \sigma_3))\,]\!]\,(S) = [\![\,(\sigma_2 + \sigma_3)\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S)) = [\![\,\sigma_2\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S)) \cup [\![\,\sigma_3\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S)) = [\![\,(\sigma_1 \circ \sigma_2)\,]\!]\,(S) \cup [\![\,(\sigma_1 \circ \sigma_3)\,]\!]\,(S)$. By definition of the semantics, the last term is $[\![\,(\sigma_1 \circ \sigma_2) + (\sigma_1 \circ \sigma_3)\,]\!]\,(S)$. Thus $\circ$ distributes over $+$ from the left.

P9: $((\sigma_1 + \sigma_2) \circ \sigma_3) = ((\sigma_1 \circ \sigma_3) + (\sigma_2 \circ \sigma_3))$.
By definition of the semantics, $[\![\,((\sigma_1 + \sigma_2) \circ \sigma_3)\,]\!]\,(S) = [\![\,\sigma_3\,]\!]\,(\,[\![\,(\sigma_1 + \sigma_2)\,]\!]\,(S)) = [\![\,\sigma_3\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S) \cup [\![\,\sigma_2\,]\!]\,(S))$. By Lemma 2, the latter is equal to $[\![\,\sigma_3\,]\!]\,(\,[\![\,\sigma_1\,]\!]\,(S)) \cup [\![\,\sigma_3\,]\!]\,(\,[\![\,\sigma_2\,]\!]\,(S))$. By definition of the semantics, this is $[\![\,(\sigma_1 \circ \sigma_3)\,]\!]\,(S) \cup [\![\,(\sigma_2 \circ \sigma_3)\,]\!]\,(S)$, or equivalently, $[\![\,(\sigma_1 \circ \sigma_3) + (\sigma_2 \circ \sigma_3)\,]\!]\,(S)$. Thus, $\circ$ distributes over $+$ from the right as well.

P10: $(\sigma_1\,;\,\sigma_2) = \delta + \sigma_1 + \sigma_2 + (\sigma_1 \circ \sigma_2)$.
Follows from the definition of the semantics, property P6 and the distributivity properties P8 and P9 proved above.

P11: $(\sigma_1\,;\,(\sigma_2\,;\,\sigma_3)) = ((\sigma_1\,;\,\sigma_2)\,;\,\sigma_3)$.
By definition, $(\sigma_1\,;\,(\sigma_2\,;\,\sigma_3)) = (\sigma_1 + \delta) \circ ((\sigma_2\,;\,\sigma_3) + \delta) = (\sigma_1 + \delta) \circ ((\sigma_2 + \delta) \circ (\sigma_3 + \delta) + \delta)$. From property P10, $\delta \sqsubseteq ((\sigma_2 + \delta) \circ (\sigma_3 + \delta))$. Furthermore, since $+$ is idempotent (Property P1 above), the above simplifies to $(\sigma_1 + \delta) \circ ((\sigma_2 + \delta) \circ (\sigma_3 + \delta))$. From property P5, we get $(\sigma_1 + \delta) \circ ((\sigma_2 + \delta) \circ (\sigma_3 + \delta)) = ((\sigma_1 + \delta) \circ (\sigma_2 + \delta)) \circ (\sigma_3 + \delta)$. Once again, since $\delta \sqsubseteq ((\sigma_1 + \delta) \circ (\sigma_2 + \delta))$ and $+$ is idempotent, the above expression is equal to $((\sigma_1 + \delta) \circ (\sigma_2 + \delta) + \delta) \circ (\sigma_3 + \delta) = ((\sigma_1\,;\,\sigma_2)\,;\,\sigma_3)$. This proves that $;$ is associative.

P12: $(\sigma\,;\,\Theta) = (\Theta\,;\,\sigma) = (\sigma + \delta)$.
Using the definition of $;$ and applying property P4 we get $(\sigma\,;\,\Theta) = (\sigma + \delta) \circ (\Theta + \delta) = (\sigma + \delta) \circ \delta$. Applying property P6 we get $(\sigma + \delta) \circ \delta = (\sigma + \delta)$. Similarly $(\Theta\,;\,\sigma) = (\Theta + \delta) \circ (\sigma + \delta) = \delta \circ (\sigma + \delta) = (\sigma + \delta)$.

Not surprisingly, properties P1 through P9 correspond to algebraic properties of regular expressions. Since the $+$, $;$ and $\circ$ operators are associative (properties P3, P5, and P11), we omit parentheses for grouping their operands, whenever convenient.

Given a set $\mathcal{T}$ of extended clusters, let $\Pi(\mathcal{T})$ denote the set of all reachability expressions over $\mathcal{T}$. From properties P1, P2, P3 and P4 above, $(\Pi(\mathcal{T}), +)$ forms an idempotent, commutative monoid with Θ as the identity element. Similarly, from properties P5 and P6 $(\Pi(\mathcal{T}), \circ)$ forms a monoid with δ as the identity element. It follows that $(\Pi(\mathcal{T}), +, \circ)$ forms an idempotent semiring. Note that $(\Pi(\mathcal{T}), ;)$ does not have an identity element, and therefore simply forms a semigroup.

Given a state transition system, we now discuss how the set of states reachable from the initial states can be computed using reachability expressions.

Lemma 3. *Let $\{\tau_i \mid 1 \leq i \leq k\}$ be a set of extended clusters, and $\tau = \bigcup_{i=1}^{k} \tau_i$. Then $Im_\tau(S) = [\![\,\mathbf{T_1} + \ldots + \mathbf{T_k}\,]\!]\,(S)$.*

Proof. Follows directly from the definition of τ and from the definition of $[\![\,\mathbf{T_1} + \ldots + \mathbf{T_k}\,]\!]$.

Lemma 4. *Let $B = (V, Q, Q_0, \Upsilon)$ be a state transition system, σ be a reachability expression over a set $\mathcal{T}$ of* *extended clusters of B, and $S_0 \subseteq Q$. Let S_j denote $[\![\overset{j}{\underset{i=0}{+}} (\sigma)^i]\!]_B(S_0)$ for all $j \geq 0$. If $S_t = S_{t+1}$ for some positive integer t, then $[\![*\sigma]\!]_B(S_0) = S_t$.*

Proof. Since $S_j = [\![\overset{j}{\underset{i=0}{+}} (\sigma)^i]\!]_B(S_0)$, it follows from the definition of the semantics that $S_j \subseteq S_{j+1}$ for all $j \geq 0$. We first show using induction on r that if $S_t = S_{t+1}$, then $S_r = S_t$ for all $r \geq t+1$.
Basis: Since $S_{t+1} = S_t$, the result holds for $r = t+1$.
Hypothesis: Let $S_r = S_t$ for $t+1 \leq r \leq n$.
Induction step: Using the definition of the semantics we get $[\![(\sigma)^{n+1}]\!]_B(S_0) = [\![\sigma \circ (\sigma)^n]\!]_B(S_0)$. Using the associativity of $\circ$, $[\![\sigma \circ (\sigma)^n]\!]_B(S_0) = [\![(\sigma)^n \circ \sigma]\!]_B(S_0)$. By the induction hypothesis, $[\![\overset{n}{\underset{i=0}{+}} (\sigma)^i]\!]_B(S_0) = [\![\overset{t}{\underset{i=0}{+}} (\sigma)^i]\!]_B(S_0)$. This implies that $[\![(\sigma)^n]\!]_B(S_0) \subseteq [\![\overset{t}{\underset{i=0}{+}} (\sigma)^i]\!]_B(S_0)$. Using Corollary 1 and the definition of the semantics, $[\![(\sigma)^n \circ \sigma]\!]_B(S_0) \subseteq [\![(\overset{t}{\underset{i=0}{+}} (\sigma)^i) \circ \sigma]\!]_B(S_0)$. Therefore, by the distributivity property P9, $[\![(\sigma)^{n+1}]\!]_B(S_0) \subseteq [\![\overset{t+1}{\underset{i=1}{+}} (\sigma)^i]\!]_B(S_0)$. Since $[\![\overset{t+1}{\underset{i=1}{+}} (\sigma)^i]\!]_B(S_0) \subseteq [\![\overset{t+1}{\underset{i=0}{+}} (\sigma)^i]\!]_B(S_0) = S_{t+1} = S_t$, by transitivity of set inclusion, $[\![\sigma^{n+1}]\!]_B(S_0) \subseteq S_t$. Now recall from the induction hypothesis that $S_n = S_t$. Hence, $S_{n+1} = S_n \cup [\![(\sigma)^{n+1}]\!]_B(S_0) = S_t$ as well. This proves that $S_r = S_t$ for all $r \geq t+1$.
To show that $[\![*\sigma]\!]_B(S_0) = S_t$, consider a state $q \in [\![*\sigma]\!]_B(S_0)$. By the definition of the semantics, there exists an $r\,(\geq 0)$ such that $q \in [\![(\sigma)^r]\!]_B(S_0)$. In other words, $q \in S_r$. If $r < t$, we have $q \in S_t$ since $S_j \subseteq S_{j+1}$ for all $j \geq 0$. If $r \geq t$, we again have $q \in S_t$ since $S_j = S_t$ for all $j \geq t$. Hence, $[\![*\sigma]\!]_B(S_0) \sqsubseteq S_t$. Also, from the definition of $[\![*\sigma]\!]_B(S_0)$, we have $S_t \sqsubseteq [\![*\sigma]\!]_B(S_0)$. Therefore, $[\![*\sigma]\!]_B(S_0) = S_t$.

Theorem 1. *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 of B such that $\bigcup_{i=1}^{k} \tau_i = \Upsilon$. The set of reachable states of B, denoted as reach(B), is given by $[\![*(\mathbf{T_1} + \ldots + \mathbf{T_k})]\!](Q_0)$.*

Proof. Let $\mathbf{T}$ be the reachability expression corresponding to the cluster Υ. By the definition of the semantics, $[\![(\mathbf{T})^i]\!](Q_0)$ gives the set of states reachable from some state in Q_0 in exactly i steps. Therefore *reach*$(B) = \bigcup_{i=0}^{\infty} [\![(\mathbf{T})^i]\!](Q_0)$. By definition, the right-hand side of the above equality is $[\![*\mathbf{T}]\!](Q_0)$. The lemma now follows by noting that $\mathbf{T} = (\mathbf{T_1} + \ldots + \mathbf{T_k})$ by Lemma 3.

Like regular expressions, reachability expressions enjoy several useful inclusion properties. These are discussed below.

Lemma 5. *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. (a) Since $\sigma_1 \sqsubseteq \sigma_2$ and $\sigma_3 \sqsubseteq \sigma_4$, for every set S of states, $[\![\sigma_1]\!](S) \subseteq [\![\sigma_2]\!](S)$ and $[\![\sigma_3]\!](S) \subseteq [\![\sigma_4]\!](S)$. Therefore, $[\![\sigma_1]\!](S) \cup [\![\sigma_3]\!](S) \subseteq [\![\sigma_2]\!](S) \cup [\![\sigma_4]\!](S)$. In other words, $(\sigma_1 + \sigma_3) \sqsubseteq (\sigma_2 + \sigma_4)$. Since $\sigma_1 \sqsubseteq \sigma_2$, therefore $[\![\sigma_1]\!](S) \subseteq [\![\sigma_2]\!](S)$. Applying Corollary 1, we get $[\![\sigma_3]\!]([\![\sigma_1]\!](S)) \subseteq [\![\sigma_3]\!]([\![\sigma_2]\!](S))$. However, $\sigma_3 \sqsubseteq \sigma_4$ implies that $[\![\sigma_3]\!]([\![\sigma_2]\!](S)) \subseteq [\![\sigma_4]\!]([\![\sigma_2]\!](S))$. From the transitivity of set inclusion, we then get $[\![\sigma_3]\!]([\![\sigma_1]\!](S)) \subseteq [\![\sigma_4]\!]([\![\sigma_2]\!](S))$. Hence, $(\sigma_1 \circ \sigma_3) \sqsubseteq (\sigma_2 \circ \sigma_4)$. Since $(\sigma_i ; \sigma_j) = (\sigma_i + \delta) \circ (\sigma_j + \delta)$, it follows from the results above (for $op \in \{+, \circ\}$) that $(\sigma_1 ; \sigma_3) \sqsubseteq (\sigma_2 ; \sigma_4)$.

(b) By definition, $(\sigma_1 ; \sigma_2)^{i+1} = (\sigma_1 ; \sigma_2) \circ (\sigma_1 ; \sigma_2)^i$ for all $i \geq 0$. From property P10 listed above, $\delta \sqsubseteq (\sigma_1 ; \sigma_2)$. Applying Lemma 5(a), we get $\delta \circ (\sigma_1 ; \sigma_2)^i \sqsubseteq (\sigma_1 ; \sigma_2)^{i+1}$. By property P6, we therefore have $(\sigma_1 ; \sigma_2)^i \sqsubseteq (\sigma_1 ; \sigma_2)^{i+1}$.

(c) We show by induction on i that if $\sigma_1 \sqsubseteq \sigma_2$, then $(\sigma_1)^i \sqsubseteq (\sigma_2)^i$ for all $i \geq 0$.
Basis: If $i = 0$, $(\sigma_1)^i = (\sigma_2)^i = \delta$ by definition. Therefore, $(\sigma_1)^i \sqsubseteq (\sigma_2)^i$.
Hypothesis: Let $(\sigma_1)^i \sqsubseteq (\sigma_2)^i$ for $0 \leq i \leq k$.
Induction step: Since $\sigma_1 \sqsubseteq \sigma_2$ and $(\sigma_1)^k \sqsubseteq (\sigma_2)^k$ (from induction hypothesis), by applying Lemma 5(a), we get $(\sigma_1 \circ (\sigma_1)^k) \sqsubseteq (\sigma_2 \circ (\sigma_2)^k)$. In other words, $(\sigma_1)^{k+1} \sqsubseteq (\sigma_2)^{k+1}$. Therefore, $(\sigma_1)^i \sqsubseteq (\sigma_2)^i$ for all $i \geq 0$.
Using the above result, we get $\bigcup_{i=0}^{\infty} [\![(\sigma_1)^i]\!](S) \subseteq \bigcup_{i=0}^{\infty} [\![(\sigma_2)^i]\!](S)$. Hence, $[\![*\sigma_1]\!](S) \subseteq [\![*\sigma_2]\!](S)$ for all sets S of states. In other words, $(*\sigma_1) \sqsubseteq (*\sigma_2)$.

(d) We first show that $(*\sigma_1)^2 = (*\sigma_1)$. Let S be a set of states, and let $S_1 = [\![*\sigma_1]\!](S)$ and $S_2 = [\![(*\sigma_1)^2]\!](S)$. By definition, $S_2 = [\![*\sigma_1]\!](S_1)$. Thus, $S_2 = \bigcup_{i=0}^{\infty} [\![(\sigma_1)^i]\!](S_1)$ and hence, $S_1 \subseteq S_2$. Recalling the definitions of S_1 and S_2, we get $(*\sigma_1) \sqsubseteq (*\sigma_1)^2$.
To show that $(*\sigma_1)^2 \sqsubseteq (*\sigma_1)$, we consider an arbitrary state $q \in [\![(*\sigma_1)^2]\!](S)$. Since $[\![(*\sigma_1)^2]\!](S) = [\![*\sigma_1]\!](S_1)$, there exists a state $q' \in S_1$ and an integer $r \geq 0$ such that $q \in [\![(\sigma_1)^r]\!](\{q'\})$. Since $q' \in S_1$ and $S_1 = [\![*\sigma_1]\!](S)$, there exists an integer $t \geq 0$ such that $q' \in [\![(\sigma_1)^t]\!](S)$. Thus, $q \in [\![(\sigma_1)^r]\!]([\![(\sigma_1)^t]\!](S))$. In other words, $q \in [\![(\sigma_1)^t \circ (\sigma_1)^r]\!](S)$. Since $\circ$ is associative, we can express the above as $q \in [\![(\sigma_1)^{t+r}]\!](S)$. It follows that $q \in [\![*\sigma_1]\!](S)$. Therefore, $[\![(*\sigma_1)^2]\!](S) \subseteq [\![*\sigma_1]\!](S)$ and hence, $(*\sigma_1)^2 \sqsubseteq (*\sigma_1)$. Since we have also shown above that $(*\sigma_1) \sqsubseteq (*\sigma_1)^2$, we get $(*\sigma_1)^2 = (*\sigma_1)$.
We now show by induction on i that $(*\sigma_1)^i = (*\sigma_1)$ for all $i \geq 1$.

Basis: For $i = 1$, the result holds trivially.
Hypothesis: Let us assume that $(*\sigma_1)^i = (*\sigma_1)$ for $1 \leq i \leq k$.
Induction step: By definition, $(*\sigma_1)^{k+1} = (*\sigma_1) \circ (*\sigma_1)^k$. By the induction hypothesis, $(*\sigma_1)^k = (*\sigma_1)$. Therefore, $(*\sigma_1)^{k+1} = (*\sigma_1) \circ (*\sigma_1) = (*\sigma_1)^2$. However, we have shown above that $(*\sigma_1)^2 = (*\sigma_1)$. Therefore, $(*\sigma_1)^{k+1} = (*\sigma_1)$. We have thus shown by induction that $(*\sigma_1)^i = (*\sigma_1)$ for all $i \geq 1$.
To show that $*(*\sigma_1) = (*\sigma_1)$, we note that $[\![\, *(*\sigma_1) \,]\!](S) = \bigcup_{j=0}^{\infty} [\![\, (*\sigma_1)^j \,]\!](S)$. However, we have seen above that $(*\sigma_1)^j = (*\sigma_1)$ for all $j \geq 1$. Hence, $\bigcup_{j=0}^{\infty} [\![\, (*\sigma_1)^j \,]\!](S) = S \cup \bigcup_{j=1}^{\infty} [\![\, *\sigma_1 \,]\!](S)$. Since $S \subseteq [\![\, *\sigma_1 \,]\!](S)$, the right hand side of the above equality reduces to $[\![\, *\sigma_1 \,]\!](S)$. Therefore, $*(*\sigma_1) = (*\sigma_1)$.

Lemma 6. *Let $\mathcal{T} = \{\tau_1, \ldots, \tau_k\}$ be a set of extended clusters. Then $*(\mathbf{T_1} + \cdots + \mathbf{T_k})$ is maximal with respect to $\sqsubseteq$ over the set of all reachability expressions over $\mathcal{T}$.*

Proof. Let σ be an arbitrary reachability expression over $\mathcal{T}$. We use induction on the structure of σ to prove $\sigma \sqsubseteq *(\mathbf{T_1} + \cdots + \mathbf{T_k})$.
Basis: If $\sigma = \mathbf{T_i}$ for $i \in \{1, \ldots, k\}$, or $\sigma = \delta$ or $\sigma = \Theta$, the $\sqsubseteq$ relation holds trivially.
Hypothesis: Let σ_1 and σ_2 be reachability expressions over $\{\tau_1, \ldots, \tau_k\}$, such that $\sigma_1 \sqsubseteq *(\mathbf{T_1} + \cdots + \mathbf{T_k})$ and $\sigma_2 \sqsubseteq *(\mathbf{T_1} + \cdots + \mathbf{T_k})$.
Induction step: We consider the following cases:

1. $\sigma = \sigma_1 + \sigma_2$: The result follows from the induction hypothesis, Lemma 5(a) and the idempotency of the $+$ operator.
2. $\sigma = \sigma_1 \circ \sigma_2$: The result follows from the induction hypothesis, Lemma 5(a) and Lemma 5(d).
3. $\sigma = \sigma_1 \,;\, \sigma_2$: The result follows from the induction hypothesis, Lemma 5(a), the definition of the semantics of the $;$ operator, and Lemma 5(d).
4. $\sigma = (\sigma_1)$: The result follows trivially from the induction hypothesis and the definition of $[\![\, (\sigma_1) \,]\!]$.
5. $\sigma = *\sigma_1$: The result follows from the induction hypothesis, Lemma 5(c) and Lemma 5(d).

This completes the induction, proving that all reachability expressions over $\{\tau_1, \ldots, \tau_k\}$ are covered by $*(\mathbf{T_1} + \cdots + \mathbf{T_k})$. In other words, $*(\mathbf{T_1} + \cdots + \mathbf{T_k})$ is maximal with respect to the $\sqsubseteq$ relation.

Semantic equivalence relations between reachability expressions within the scope of the $*$ operator are particularly useful for identifying alternative search strategies that give the same end result but with potentially different performance. We discuss these equivalence relations below.

Lemma 7. *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. By definition of the semantics, $\sigma_2 \sqsubseteq (*\sigma_2)$. Therefore, by Lemma 5(a), $(\sigma_1 \,;\, \sigma_2) \sqsubseteq (\sigma_1 \,;\, (*\sigma_2))$, and by Lemma 5(c), $*(\sigma_1 \,;\, \sigma_2) \sqsubseteq *(\sigma_1 \,;\, (*\sigma_2))$.
To show that $*(\sigma_1 \,;\, (*\sigma_2)) \sqsubseteq *(\sigma_1 \,;\, \sigma_2)$, consider a set S of states, and let $q \in [\![\, \sigma_1 \,;\, (*\sigma_2) \,]\!](S)$. Since $(\sigma_1 \,;\, (*\sigma_2)) = (\delta + \sigma_1 + (*\sigma_2) + (\sigma_1 \circ (*\sigma_2)))$, we have the following cases:

- $q \in [\![\, \delta + \sigma_1 \,]\!](S)$: Since $(\delta + \sigma_1) \sqsubseteq (\sigma_1 \,;\, \sigma_2)$, $q \in [\![\, *(\sigma_1 \,;\, \sigma_2) \,]\!](S)$.
- $q \in [\![\, (*\sigma_2) \,]\!](S)$: There exists an integer $r \geq 0$ such that $q \in [\![\, (\sigma_2)^r \,]\!](S)$. Since $\sigma_2 \sqsubseteq (\sigma_1 \,;\, \sigma_2)$, by Lemma 5(c), $(\sigma_2)^r \sqsubseteq (\sigma_1 \,;\, \sigma_2)^r$. Hence, $q \in [\![\, *(\sigma_1 \,;\, \sigma_2) \,]\!](S)$.
- $q \in [\![\, \sigma_1 \circ (*\sigma_2) \,]\!](S)$: There exists an integer $r \geq 0$ such that $q \in [\![\, \sigma_1 \circ (\sigma_2)^r \,]\!](S)$. Since $(\sigma_2)^r \sqsubseteq (\sigma_1 \,;\, \sigma_2)^r$ and $\sigma_1 \sqsubseteq (\sigma_1 \,;\, \sigma_2)$, by Lemma 5(a), $(\sigma_1 \circ (\sigma_2)^r) \sqsubseteq (\sigma_1 \,;\, \sigma_2)^{r+1}$. Therefore we get, $q \in [\![\, (\sigma_1 \,;\, \sigma_2)^{r+1} \,]\!](S)$ and hence, $q \in [\![\, *(\sigma_1 \,;\, \sigma_2) \,]\!](S)$.

We have thus shown above that $(\sigma_1 \,;\, (*\sigma_2)) \sqsubseteq *(\sigma_1 \,;\, \sigma_2)$. Applying Lemmas 5(c) and (d), we 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 2. *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. 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 show that $*(\sigma_1 + \sigma_Y) = *(\sigma_1 \,;\, \sigma_Y)$.
Since $(\sigma_1 + \sigma_Y) \sqsubseteq (\sigma_1 \,;\, \sigma_Y)$, by Lemma 5(c), $*(\sigma_1 + \sigma_Y) \sqsubseteq *(\sigma_1 \,;\, \sigma_Y)$. However, $(\sigma_1 \,;\, \sigma_Y) = \delta + \sigma_1 + \sigma_Y + (\sigma_1 \circ \sigma_Y) \sqsubseteq *(\sigma_1 + \sigma_Y)$. Therefore, by Lemmas 5(c) and (d), $*(\sigma_1 \,;\, \sigma_Y) \sqsubseteq *(\sigma_1 + \sigma_Y)$. It follows 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, applying Lemma 7 we get $*(\sigma_1 + \sigma_Y) = *(\sigma_1 \,;\, (*\sigma_Y)) = *(\sigma_1 \,;\, (*\sigma_Z)) = *(\sigma_1 \,;\, \sigma_Z)$. Since $+$ and $;$ are associative, the proof is complete.

Corollary 2. *The $;$ operator is commutative within the scope of $*$.*

Proof. Follows from Theorem 2 and the commutativity of $+$.

Corollary 3. *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 of B such that $\bigcup_{i=1}^{k} \tau_i = \Upsilon$. The set of* *reachable states of B is given by* $reach(B) = [\![*(\mathbf{T_1} ; \cdots ; \mathbf{T_k})]\!] (Q_0)$.

Proof. Follows from Theorem 1 and Theorem 2.

We now discuss some results that help in explaining why some reachability expressions lead to search strategies with better performance than those described by other expressions.

Theorem 3. *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. (a) Since $\sigma_Y \sqsubseteq \sigma_Z$, by Lemma 5(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 5(b), it can 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 $n = k \cdot i + r$, where $i \geq 0$ and $0 \leq r < k$. We first show by induction on k that $(\overset{k \cdot i}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq (\sigma_Z)^{(k-1) \cdot i+1}$ for $k \geq 2$.

Basis ($k = 2$) : We prove that $(\overset{2i}{\underset{j=0}{+}} (\sigma_1 + \sigma_2)^j) \sqsubseteq (\sigma_1 ; \sigma_2)^{i+1}$, for $i \geq 0$.

Consider a set S of states, and let $q \in [\![\overset{2i}{\underset{j=0}{+}} (\sigma_1 + \sigma_2)^j]\!] (S)$. Then, there exists a non-negative integer N, no greater than $2i$, such that $q \in [\![(\sigma_1 + \sigma_2)^N]\!] (S)$. If $N = 0$, $q \in [\![\delta]\!] (S)$. However, since $\delta \sqsubseteq (\sigma_1 ; \sigma_2)^{i+1}$, $q \in [\![(\sigma_1 ; \sigma_2)^{i+1}]\!] (S)$ as well. If $N > 0$, there exists a sequence of indices, $(r_1, \ldots, r_N)$, such that each $r_j \in \{1, 2\}$ and $q \in [\![(\sigma_{r_1} \circ \cdots \circ \sigma_{r_N})]\!] (S)$. We show below that $(\sigma_{r_1} \circ \cdots \circ \sigma_{r_N}) \sqsubseteq (\sigma_1 ; \sigma_2)^{i+1}$. Since $(\sigma_p)^2 \sqsubseteq \sigma_p$ for all $p \in \{1, 2\}$, it suffices to consider sequences $(r_1, \ldots, r_M)$ in which all consecutive indices are distinct, and $1 \leq M \leq N$. However, since each $r_j \in \{1, 2\}$, indices 1 and 2 must alternate in $(r_1, \ldots, r_M)$. We have the following cases:

- $M = 2t$, $t > 0$, $r_1 = 1$: $(\sigma_{r_1} \circ \cdots \circ \sigma_{r_M}) = (\sigma_1 \circ \sigma_2)^t \sqsubseteq (\sigma_1 ; \sigma_2)^t$.
- $M = 2t$, $t > 0$, $r_1 = 2$: $(\sigma_{r_1} \circ \cdots \circ \sigma_{r_M}) = (\sigma_2 \circ (\sigma_1 \circ \sigma_2)^{t-1} \circ \sigma_1) \sqsubseteq (\sigma_1 ; \sigma_2)^{t+1}$.
- $M = 2t + 1$, $t > 0$, $r_1 = 1$: $(\sigma_{r_1} \circ \cdots \circ \sigma_{r_M}) = ((\sigma_1 \circ \sigma_2)^t \circ \sigma_1) \sqsubseteq (\sigma_1 ; \sigma_2)^{t+1}$.
- $M = 2t + 1$, $t > 0$, $r_1 = 2$: $(\sigma_{r_1} \circ \cdots \circ \sigma_{r_M}) = (\sigma_2 \circ (\sigma_1 \circ \sigma_2)^t) \sqsubseteq (\sigma_1 ; \sigma_2)^{t+1}$.

Thus, in all cases, $(\sigma_{r_1} \circ \cdots \circ \sigma_{r_M}) \sqsubseteq (\sigma_1 ; \sigma_2)^{\lfloor M/2 \rfloor + 1}$. Since M is bounded above by $2i$, $\lfloor M/2 \rfloor + 1 \leq i + 1$. By Lemma 5(b) , it then follows that $(\sigma_{r_1} \circ \cdots \circ \sigma_{r_M}) \sqsubseteq (\sigma_1 ; \sigma_2)^{i+1}$.

Hypothesis:
Let $(\overset{k \cdot i}{\underset{j=0}{+}} (\sigma_1 + \cdots + \sigma_k)^j) \sqsubseteq (\sigma_1 ; \cdots ; \sigma_k)^{(k-1) \cdot i+1}$ for $2 \leq k \leq m$, $i \geq 0$ and for all $\sigma_1, \ldots, \sigma_k$.

Induction step: Let $\sigma_W = \sigma_m + \sigma_{m+1}$. Then, σ_Y or $(\sigma_1 + \cdots + \sigma_{m+1})$ can be rewritten as $(\sigma_1 + \cdots + \sigma_{m-1} + \sigma_W)$. Note that the last expression is obtained by applying the $+$ operator to m reachability expressions. Since $+$ is idempotent and $\circ$ distributes over $+$, $(\overset{(m+1) \cdot i}{\underset{j=0}{+}} (\sigma_Y)^j) = (\overset{m \cdot i}{\underset{j=0}{+}} (\sigma_Y)^j) \circ (\overset{i}{\underset{l=0}{+}} (\sigma_Y)^l)$. By the induction hypothesis, $(\overset{m \cdot i}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq (\sigma_1 ; \cdots ; \sigma_{m-1} ; \sigma_W)^{(m-1) \cdot i+1}$. Since $\sigma_W = (\sigma_m + \sigma_{m+1}) \sqsubseteq (\sigma_m ; \sigma_{m+1})$, by Lemmas 5(a) and 5(c), we get $(\sigma_1 ; \cdots ; \sigma_{m-1} ; \sigma_W)^{(m-1) \cdot i+1} \sqsubseteq (\sigma_1 ; \cdots ; \sigma_{m-1} ; \sigma_m ; \sigma_{m+1})^{(m-1) \cdot i+1} = \sigma_Z^{(m-1) \cdot i+1}$, where $\sigma_Z = (\sigma_1 ; \cdots \sigma_{m+1})$. Hence, by transitivity, $(\overset{m \cdot i}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq \sigma_Z^{(m-1) \cdot i+1}$.

We have also shown in Theorem 3(a) that $(\overset{i}{\underset{l=0}{+}} (\sigma_Y)^l) \sqsubseteq (\sigma_Z)^i$. By Lemma 5(a), we then get $(\overset{mi}{\underset{j=0}{+}} (\sigma_Y)^j) \circ (\overset{i}{\underset{l=0}{+}} (\sigma_Y)^l) \sqsubseteq (\sigma_Z^{(m-1) \cdot i+1} \circ \sigma_Z^i)$. In other words, $(\overset{(m+1) \cdot i}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq \sigma_Z^{((m+1)-1) \cdot i+1}$. We have therefore shown that if $\sigma_Y = (\sigma_1 + \cdots + \sigma_k)$ and $\sigma_Z = (\sigma_1 ; \cdots ; \sigma_k)$, then $(\overset{k \cdot i}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq \sigma_Z^{(k-1) \cdot i+1}$ for all $k \geq 2$.

Since $(\overset{r}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq \sigma_Z^r$ (by Theorem 3(a)), applying Lemma 5(a), we get $(\overset{k \cdot i+r}{\underset{j=0}{+}} (\sigma_Y)^j) = (\overset{k \cdot i}{\underset{j=0}{+}} (\sigma_Y)^j) \circ (\overset{r}{\underset{j=0}{+}} (\sigma_Y)^j) \sqsubseteq \sigma_Z^{(k-1) \cdot i+1} \circ \sigma_Z^r = \sigma_Z^{(k \cdot i+r-(i-1))}$. The theorem is now proved by noting that $k \cdot i + r = n$ and $i = \lfloor n/k \rfloor$.

Let $B = (V, Q, Q_0, \Upsilon)$ be a finite state transition system. Let $\{\tau_1, \ldots, \tau_k\}$ be a set of extended clusters, satisfying $\bigcup_{i=1}^k \tau_i = \Upsilon$. From Theorem 1 and Corollary 3, the set of reachable states of B, denoted *reach*(B), is given by $[\![*(\mathbf{T_1} + \ldots + \mathbf{T_k})]\!] (Q_0)$ and also alternatively by $[\![*(\mathbf{T_1} ; \ldots ; \mathbf{T_k})]\!] (Q_0)$. However, Theorem 3(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 $[\![(\mathbf{T_1} + \ldots + \mathbf{T_k})^i]\!] (S)$ is comparable in complexity to computing $[\![(\mathbf{T_1} ; \ldots ; \mathbf{T_k})^i]\!] (S)$, it is advantageous to use $*(\mathbf{T_1} ; \ldots ; \mathbf{T_k})$. This advantage is also demonstrated in our experiments, as reported in Section 3. If $(\sigma_p)^2 \sqsubseteq \sigma_p$ for all p, Theorem 3(b) improves the upper bound of Theorem 3(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 $*\sigma_q$.

Sometimes the computational effort involved in computing the image of a set of states under a certain transition cluster may be much higher than that required to compute images under other transition clusters. The following theorem gives a reachability expression that can be useful in such situations.

Theorem 4. *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 $(*\sigma_X) \circ *(\mathbf{T_k}\,;\,(*\sigma_X))$. Then the following hold.*

(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 $N_k(\sigma)$ denote the number of times image under τ_k is computed until the complete set of states reachable from Q_0 is obtained during evaluation of $[\![\, \sigma \,]\!]\,(Q_0)$. Then $N_k(\widehat{\sigma}) \leq N_k(\sigma) + 1$.

Proof. (a) From Lemma 7 and Theorem 2, $*(\mathbf{T_k}\,;\,(*\sigma_X)) = *(\mathbf{T_k}\,;\,\sigma_X) = *(\mathbf{T_k} + \sigma_X)$. Since $\delta \sqsubseteq *\sigma_X$, composing both sides with $*(\mathbf{T_k} + \sigma_X)$ (or, equivalently with $*(\mathbf{T_k}\,;\,(*\sigma_X))$) and by applying Lemma 5(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, $\widehat{\sigma} \sqsubseteq *(\mathbf{T_k} + \sigma_X)$ by Lemma 6. Hence, $\widehat{\sigma} = *(\mathbf{T_k} + \sigma_X)$. We know from Theorem 1 that $[\![\, *(\mathbf{T_k} + \sigma_X) \,]\!]\,(Q_0)) = reach(B)$. It therefore follows that $[\![\, \widehat{\sigma} \,]\!]\,(Q_0) = reach(B)$.

(b) Since $\widehat{\sigma} = (*\sigma_X)\,;\,*(\mathbf{T_k}\,;\,(*\sigma_X))$, evaluation of $[\![\, \widehat{\sigma} \,]\!]\,(Q_0)$ involves execution of the following steps:
(i) A variable W (of type "set of states") is initialized to $[\![\, *\sigma_X \,]\!]\,(Q_0)$.
(ii) If no new states have been added to W in the previous step, we terminate and return W as the value of $[\![\, \widehat{\sigma} \,]\!]\,(Q_0)$.
(iii) Else, W is updated to the union of W and $[\![\, \mathbf{T_k}\,;\,(*\sigma_X) \,]\!]\,(W)$, and we go back to step(ii).

Since $\mathbf{T_k}$ is evaluated once during each execution of step(iii) above, $N_k(\widehat{\sigma})$ equals the number of executions of step(iii).

Let σ be an arbitrary reachability expression over $\{\tau_1, \ldots, \tau_k\}$ such that $[\![\, \sigma \,]\!]\,(Q_0) = reach(B)$. From Lemma 6, we know that all reachability expressions not involving $\mathbf{T_k}$, i.e. over the extended clusters $\{\tau_1, \ldots, \tau_{k-1}\}$, are covered by $*\sigma_X$. However, since $\mathbf{T_k}$ is not covered by $*(\mathbf{T_1} + \cdots + \mathbf{T_{k-1}})$, $[\![\, \sigma \,]\!]\,(Q_0) \not\sqsubseteq [\![\, *\sigma_X \,]\!]\,(Q_0)$ in general. Instead, if M denotes the value of $N_k(\sigma)$, we have $[\![\, \sigma \,]\!]\,(Q_0) \sqsubseteq [\![\, *\sigma_X \,]\!]\,(Q_0) \cup [\![\, *\sigma_X \circ \mathbf{T_k} \circ *\sigma_X \,]\!]\,(Q_0) \cup [\![\, *\sigma_X \circ (\mathbf{T_k} \circ *\sigma_X)^2 \,]\!]\,(Q_0) \cup \cdots \cup [\![\, *\sigma_X \circ (\mathbf{T_k} \circ *\sigma_X)^M \,]\!]\,(Q_0)$. Since $reach(B) = [\![\, \sigma \,]\!]\,(Q_0)$ (this is our premise), it follows that $reach(B) \sqsubseteq [\![\, (*\sigma_X) \circ (\overset{M}{\underset{j=0}{+}} (\mathbf{T_k} \circ (*\sigma_X))^j) \,]\!]\,(Q_0)$. Since $(\sigma_i \circ \sigma_j) \sqsubseteq (\sigma_i\,;\,\sigma_j)$ for all σ_i and σ_j, we have $reach(B) \sqsubseteq [\![\, (*\sigma_X) \circ (\overset{M}{\underset{j=0}{+}} (\mathbf{T_k}\,;\,(*\sigma_X))^j) \,]\!]\,(Q_0)$. However, from Theorem 4(a), we know that $[\![\, (*\sigma_X) \circ (\overset{M}{\underset{j=0}{+}} (\mathbf{T_k}\,;\,(*\sigma_X))^j) \,]\!]\,(Q_0) \sqsubseteq reach(B)$. It therefore follows that $[\![\, (*\sigma_X) \circ (\overset{M}{\underset{j=0}{+}} (\mathbf{T_k}\,;\,(*\sigma_X))^j) \,]\!]\,(Q_0) = reach(B)$. Since $\widehat{\sigma} = (*\sigma_X) \circ *(\mathbf{T_k}\,;\,(*\sigma_X))$, the above equality for $reach(B)$ implies that when computing $[\![\, \widehat{\sigma} \,]\!]\,(Q_0)$, the total number of iterations of step(iii) of the procedure outlined earlier is no larger than $M + 1$. Hence, $N_k(\widehat{\sigma}) \leq N_k(\sigma) + 1$.

Given a finite-state system, Theorem 4 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 automata and updating *all* their clocks simultaneously, 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 4(b). Note, however, that application of Theorem 4(b) may increase the number of times image under transition clusters other than τ_k are computed. Since the overall performance of the search depends on the combined cost of computing image under all transition clusters, application of Theorem 4(b) is not guaranteed to improve the search performance. Thus, one must be careful in estimating how the benefit of minimizing image compuations under τ_k compares with the cost of potentially increasing image computations under the other τ_i's, before applying Theorem 4(b).

Interestingly, the motivation and reasoning behind Theorem 4 is similar to the argument used to justify the saturation algorithm in the tool SMART [9]. In SMART, the transition relation is partitioned into clusters and ordered into levels. Suppose $\tau_1, \ldots, \tau_k$ are ordered into levels 1 to k respectively. The ordering is done such that the support set for level $i + 1$ is not contained in the union of support sets for levels 1 to i. Then starting at level 1 and increasing the level number, we compute the fixpoint with respect to all lower levels. This can be viewed as a search strategy similar to that described by the reachability expression $(*\mathbf{T_1}) \circ (*(\mathbf{T_1} + \mathbf{T_2})) \circ \cdots \circ (*(\mathbf{T_1} + \ldots + \mathbf{T_k}))$. Note that in the above ex-

pression, there are many more image computations using τ_1 than using τ_k. Also τ_1 being lower in the level has a smaller support set, while τ_k possibly has a larger support set. Thus the saturation-based schedule also tries to reduce the number of image computations with respect to (computationally) "difficult" transition clusters. If $\mathbf{T_1}$ and $\mathbf{T_2}$ can be represented using much smaller BDDs than $\mathbf{T_1} + \mathbf{T_2}$, then it may be beneficial to consider the expression $(*\mathbf{T_1}) \circ (*(*\mathbf{T_1} ; *\mathbf{T_2}))$ instead of $(*\mathbf{T_1}) \circ (*(\mathbf{T_1} + \mathbf{T_2}))$. However, this may increase the total number of fixpoint iterations and fixpoint checks required, and one has to carefully weigh the advantages against this cost.
As seen above, often there exists a tradeoff between keeping the BDD sizes small and increasing the number of fixpoint iterations. If the intermediate BDD sizes consume too much memory and become a bottleneck for the computation, then expressions that potentially increase the number of iterations but keep BDD sizes small can be expected to perform better. On the other hand, if the BDD sizes do not blow up, then reachability expressions that reduce the number of iterations should be preferred.

In several applications, 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. Note that a similar observation is exploited in techniques like chaining [14] and saturation [9] as well.

Theorem 5. *Let $\{\sigma_1, \ldots, \sigma_k\}$ be an ordered set of reachability expressions satisfying the condition $(*\sigma_i) \circ (*\sigma_j) \circ \sigma_i = (*\sigma_i) \circ (*\sigma_j)$ for all i, j satisfying $1 \le i < j \le k$. Then $*(\sigma_1 ; \cdots ; \sigma_k) = (*\sigma_1) ; \cdots ; (*\sigma_k)$.*

Proof. We first show by induction on r that $(*\sigma_i) \circ (*\sigma_j) \circ (\sigma_i)^r = (*\sigma_i) \circ (*\sigma_j)$ for all $r \ge 0$ and $1 \le i < j \le k$.
Basis (r=0): Follows from Property P6 and the fact that $\sigma_i^0 = \delta$.
Hypothesis: Let $(*\sigma_i) \circ (*\sigma_j) \circ (\sigma_i)^r = (*\sigma_i) \circ (*\sigma_j)$ for $0 \le r \le n$.
Induction step: By associativity of $\circ$, $(*\sigma_i) \circ (*\sigma_j) \circ (\sigma_i)^{n+1} = ((*\sigma_i) \circ (*\sigma_j) \circ (\sigma_i)^n) \circ \sigma_i$. By the induction hypothesis, the above expression is equal to $((*\sigma_i) \circ (*\sigma_j)) \circ \sigma_i$. This, in turn, is equal to $((*\sigma_i) \circ (*\sigma_j))$ by the condition of the theorem (our premise). Hence, $(*\sigma_i) \circ (*\sigma_j) \circ (\sigma_i)^r = (*\sigma_i) \circ (*\sigma_j)$ for all $r \ge 0$.

Let $q \in [\![(*\sigma_i) \circ (*\sigma_j) \circ (*\sigma_i)]\!] (S)$ for an arbitrary set of states S. Then there exists an integer $r\,(\ge 0)$ such that $q \in [\![(*\sigma_i) \circ (*\sigma_j) \circ \sigma_i{}^r]\!] (S)$. Therefore, by the result proved above, $q \in [\![(*\sigma_i) \circ (*\sigma_j)]\!] (S)$. Hence, $(*\sigma_i) \circ (*\sigma_j) \circ (*\sigma_i) \sqsubseteq (*\sigma_i) \circ (*\sigma_j)$. Also, since $\delta \sqsubseteq *\sigma_i$, we have $(*\sigma_i) \circ (*\sigma_j) \sqsubseteq (*\sigma_i) \circ (*\sigma_j) \circ (*\sigma_i)$. Therefore, $(*\sigma_i) \circ (*\sigma_j) \circ (*\sigma_i) = (*\sigma_i) \circ (*\sigma_j)$ for all i, j satisfying $1 \le i < j \le k$.

Next, we observe that by repeated application of Lemma 7, $*(\sigma_1 ; \cdots ; \sigma_k) = *((*\sigma_1) ; \cdots ; (*\sigma_k))$. From the definition of the semantics, $*((*\sigma_1) ; \cdots ; (*\sigma_k)) = *((*\sigma_1 + \delta) \circ \cdots \circ (*\sigma_k + \delta))$. Since $\delta \sqsubseteq *\sigma_i$ for all i, the above expression simplifies to $*((*\sigma_1) \circ \cdots \circ (*\sigma_k))$. Let $\alpha = (*\sigma_1) \circ \cdots \circ (*\sigma_k)$. The theorem is now proved by showing that $*\alpha = \alpha$.

We first show by induction on r that $\alpha^r \sqsubseteq \alpha$ for all $r \ge 0$.
Basis (r=0): Follows from the definition of α and the fact that $\alpha^0 = \delta$.
Hypothesis: Let $\alpha^r \sqsubseteq \alpha$ for $0 \le r \le m$.
Induction step: By definition, $\alpha^{m+1} = \alpha \circ \alpha^m$. By the induction hypothesis, $\alpha^m \sqsubseteq \alpha$. Therefore, by Lemma 5(a), $\alpha^{m+1} \sqsubseteq \alpha^2$. To complete the induction, we prove below that $\alpha^2 = \alpha$.
By definition, $\alpha = (*\sigma_1) \circ (*\sigma_2) \circ \cdots \circ (*\sigma_k)$. We have already seen above that $(*\sigma_1) \circ (*\sigma_i) = (*\sigma_1) \circ (*\sigma_i) \circ (*\sigma_1)$ for all $i > 1$. Hence, $\alpha = ((*\sigma_1) \circ (*\sigma_2) \circ (*\sigma_1)) \circ \cdots \circ (*\sigma_k)$. Continuing similarly, we get $\alpha = ((*\sigma_1) \circ (*\sigma_2) \circ (*\sigma_1) \circ (*\sigma_3) \circ (*\sigma_1) \circ \cdots \circ (*\sigma_1) \circ (*\sigma_k))$. Now recalling that α was originally defined as $(*\sigma_1) \circ \cdots \circ (*\sigma_k)$, we have $\alpha^2 = ((*\sigma_1) \circ (*\sigma_2) \circ (*\sigma_1) \circ (*\sigma_3) \circ (*\sigma_1) \circ \cdots \circ (*\sigma_1) \circ (*\sigma_k)) \circ ((*\sigma_1) \circ \cdots \circ (*\sigma_k))$. However, $(*\sigma_1) \circ (*\sigma_k) \circ (*\sigma_1) = (*\sigma_1) \circ (*\sigma_k)$ as shown above. Therefore, $\alpha^2 = ((*\sigma_1) \circ (*\sigma_2) \circ (*\sigma_1) \circ \cdots \circ (*\sigma_{k-1}) \circ (*\sigma_1) \circ (*\sigma_k)) \circ (*\sigma_2) \circ \cdots \circ (*\sigma_k)$. Since $(*\sigma_1) \circ (*\sigma_{k-1}) \circ (*\sigma_1) = (*\sigma_1) \circ (*\sigma_{k-1})$, α^2 further simplifies to $((*\sigma_1) \circ (*\sigma_2) \circ (*\sigma_1) \circ \cdots \circ (*\sigma_{k-1}) \circ (*\sigma_k)) \circ (*\sigma_2) \circ \cdots \circ (*\sigma_k)$. Continuing similarly, we get $\alpha^2 = ((*\sigma_1) \circ (*\sigma_2) \circ (*\sigma_3) \circ \cdots \circ (*\sigma_k)) \circ (*\sigma_2) \circ \cdots \circ (*\sigma_k)$. Note that we have effectively removed $*\sigma_1$ from the second copy of α in α^2. We can continue in a similar manner to remove each $*\sigma_i$, $1 \le i \le k-1$, from the second copy of α, without changing α^2. This eventually gives us $\alpha^2 = ((*\sigma_1) \circ \cdots \circ (*\sigma_k)) \circ (*\sigma_k)$. By Lemma 5(d), $(*\sigma_k)^2 = *\sigma_k$. Therefore, $\alpha^2 = \alpha$, completing the inductive proof that $\alpha^r \sqsubseteq \alpha$ for all $r \ge 0$.

Finally, we show that $*\alpha = \alpha$. By definition of the semantics, $\alpha \sqsubseteq *\alpha$. To show that $*\alpha \sqsubseteq \alpha$ as well, consider $q \in [\![*\alpha]\!] (S)$ for an arbitrary set of states S. Then there exists an $r\,(\ge 0)$ such that $q \in [\![\alpha^r]\!] (S)$. We have shown above that $\alpha^r \sqsubseteq \alpha$ for all $r \ge 0$. Hence, $q \in [\![\alpha]\!] (S)$.

This implies that $*\alpha \sqsubseteq \alpha$. Since we have already shown that $\alpha \sqsubseteq *\alpha$, it follows that $\alpha = *\alpha$.

Recalling the definition of α, we conclude that $(*\sigma_1) \circ \cdots \circ (*\sigma_k) = *((*\sigma_1) \circ \cdots \circ (*\sigma_k))$. The proof of the theorem is completed by noting that $(*\sigma_1) \circ \cdots \circ (*\sigma_k) = (*\sigma_1) \; ; \; \cdots \; ; \; (*\sigma_k)$.

The condition $((*\sigma_i) \circ (*\sigma_j) \circ \sigma_i = (*\sigma_i) \circ (*\sigma_j))$ in Theorem 5 formalizes an ordering of dependencies between the σ_i's. Effectively, it states that once the image under $*\sigma_i$ is computed, the image under $*\sigma_j$ does not affect the computation of image under σ_i for all $1 \leq i < j \leq k$. If this condition is satisfied, Theorem 5 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.

"Chaining" [14,8] refers to a technique that has been effectively used by Roig, Cortadella and Pastor and others to significantly reduce the number of fixpoint iterations in reachability analysis of asynchronous circuits and Petri-net based models. In the context of circuits, each gate has one or more inputs and an output signal. When a transition on these inputs and/or outputs is fired, the system moves to a new state. Chaining reduces the number of fixpoint iterations by firing enabled transitions on the input and output signals of gates in a specific order, such that the effect of firing an input is taken into account when the state of the output is updated in the same fixpoint iteration. The improvements obtained by chaining can be explained using the result of Theorems 3 and 5. Suppose each transition on each signal is in a separate transition cluster. If the expression $*(\mathbf{T_1} + \ldots + \mathbf{T_k})$ is used for reachability analysis, each time a transition on the input signal of a gate fires, a new set of states is reached. However it is not until the next fixpoint iteration that the change propagates through the gate to its output. If, instead, the expression $*(\mathbf{T_1} \,;\, \ldots \,;\, \mathbf{T_k})$ or $*(*\mathbf{T_1} \,;\, \ldots \,;\, *\mathbf{T_k})$ is used with the input transition indexed lower than the output transition, then the change in the inputs is propagated to the outputs in the same fixpoint iteration. This has been shown [14] to reduce the number of iterations needed for computing the reachable state space significantly. The expression $*(\mathbf{T_1} \,;\, \ldots \,;\, \mathbf{T_k})$ or $*(*\mathbf{T_1} \,;\, \ldots \,;\, *\mathbf{T_k})$ is effective if the ordering of transition clusters captures the input to output order in which transitions may be triggered by signals. This order adds new states to the reachable state set eagerly, resulting in a reduced number of fixpoint iterations. Note, however, that techniques that reduce the number of fixpoint iterations are not guaranteed to always improve the search performance. Indeed, the cost of each iteration of the outermost fixpoint computation in $*(\mathbf{T_1} + \ldots + \mathbf{T_k})$, $*(\mathbf{T_1} \,;\, \ldots \,;\, \mathbf{T_k})$ and $*(*\mathbf{T_1} \,;\, \ldots \,;\, *\mathbf{T_k})$ could be very different. This must be factored in when applying techniques like chaining, or those suggested by Lemma 7, Corollary 3 or Theorem 5.

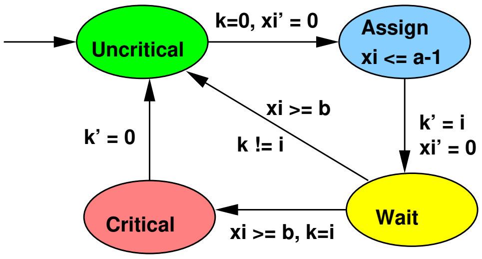

Fig. 1. Fischer's mutual exclusion protocol

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 with chaining, 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 NuSMVDP. Our tool acts as a wrapper on top of the reachability analysis engine of NuSMV [11]. 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 commonly used classes of examples for our experiments – Fischer protocol and gate-level circuits with delays.

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. Assignments to primed variables in Figure 1 refer to values of the corresponding variables after the state transition has happened. 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 [16].

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:

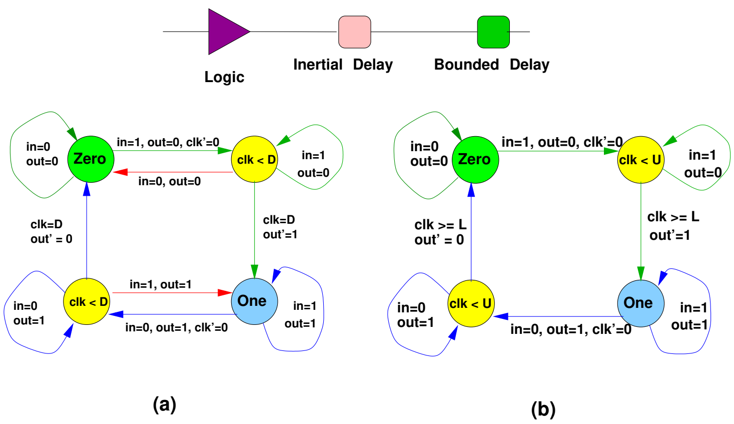

Fig. 2. (a) Inertial delay model (b) Bi-bounded pure delay model

– A *boolean logic block* that gives the boolean value of the output as a function of the boolean values of the inputs.
– 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, this element 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 behaviour 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 on 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}. A set of circuit layouts were chosen and the experiments were run with different delay parameters on this set. The details of circuits used in our experiments are given in [15]. Additional information regarding modeling of circuits as timed automata can be found in the technical report [16].

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 given 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. In our experiments, we look at seven different reachability expressions which we name as $S_0, \cdots, S_6$ respectively. Each of these expressions computes the entire set of reachable states and therefore they are semantically equivalent reachability expressions. We describe these expressions and discuss the difference in performance in computing the set of reachable states using these expressions.

Let Γ be the monolithic transition relation obtained by combining all transitions into a single cluster and let $\mathbf{\Gamma}$ denote its reachability expression. 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 $S_1 = *(\mathbf{T_1} + \cdots + \mathbf{T_k} + \mathbf{T_t})$. While this reduces the effort for each image computation, the number of image computations increases significantly. To control this, we apply Theorem 3, 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 com-

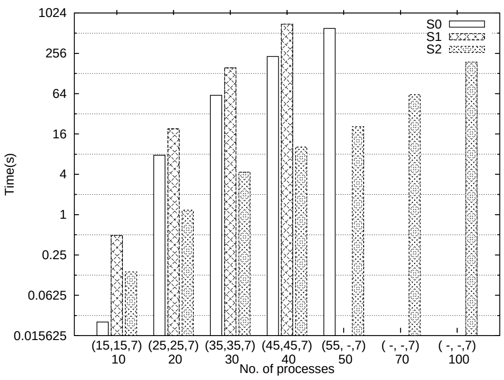

Fig. 3. Analysis of Fischer processes: Time and iteration counts for S_0, S_1, S_2.

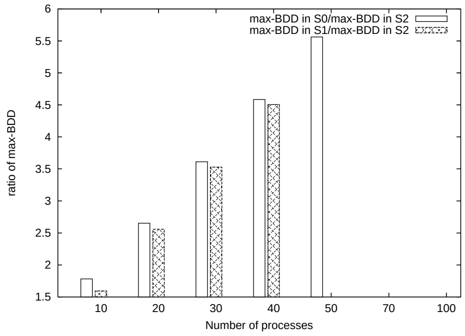

Fig. 4. Analysis of Fischer processes: Ratios of "max BDD".

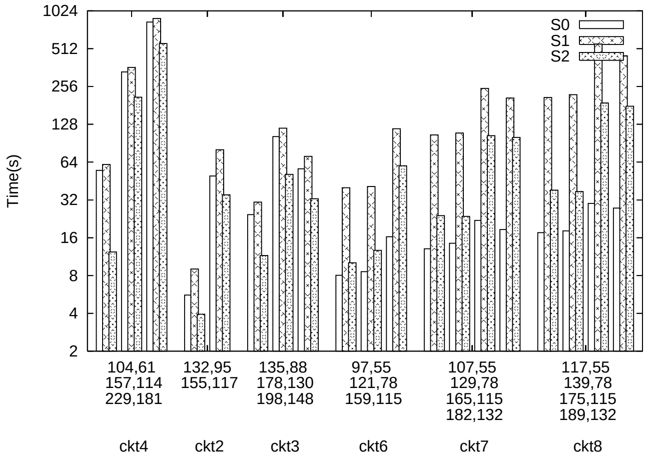

Fig. 5. Analysis of circuits: Time and iteration counts for S_1, S_2. Iteration counts for S_0 are identical to those for S_1.

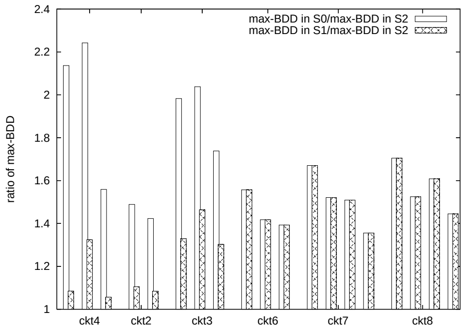

Fig. 6. Analysis of circuits: Ratios of "max BDD".

pute 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 the tables listed in the technical report [15]. For both classes of examples, we present results only for those benchmarks where at least one of the reachability expressions considered resulted in a search terminating within 30 minutes without memory overrun in the current implementation of our tool.

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 3 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 Fischer protocol examples, S_2 has the best performance. The only exception is the example with 10 processes. The reason for S_2 not performing better than S_0 in this example is that 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

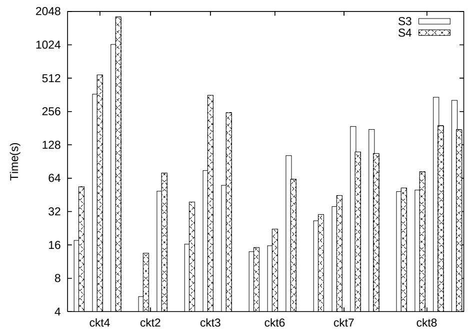

Fig. 7. Analysis of circuits: Time for schedules S_3 and S_4.

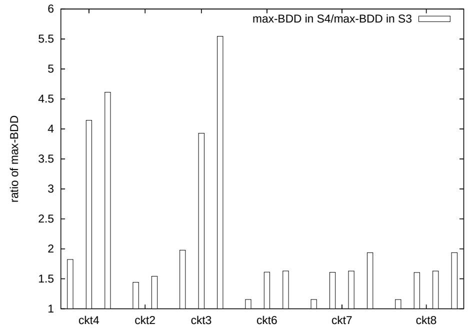

Fig. 8. Analysis of circuits: Ratios of "max BDD" using S_4 to that using S_3.

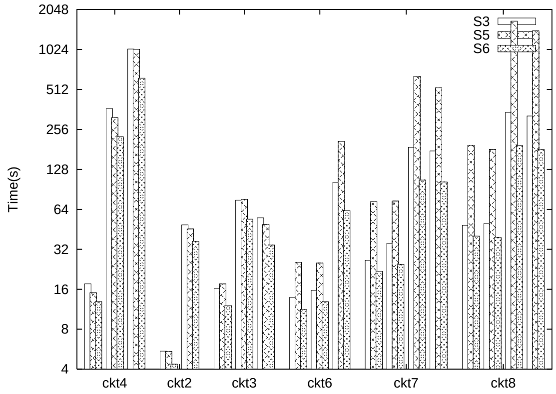

Fig. 9. Analysis of circuits: Time for schedules S_3, S_5 and S_6.

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 of 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 4 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 the reachability expression obtained from Theorem 4. 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 4, 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: $S_4 = *(\mathbf{T_1} ; \cdots ; \mathbf{T_k} ; *\mathbf{T_t})$. Note that semantically $S_0 = S_3 = S_4$, and hence, each one of them computes the entire set of reachable states.

Figure 8 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 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 5 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. We choose the initial state S to be such that all gates are stable (i.e. no gate is scheduled to change its output).

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 2, we have $*\sigma_x = *\sigma_y$ and by Theorem 5, 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 $S_6 = \sigma_z \circ *(\mathbf{T_t} ; \sigma_z)$. 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. Note also that S_3 and S_6 correspond to a naive application of chaining [14,8].

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 fixpoint checks in S_5 and S_3 compared to those in S_6. In Figure 9, the circuits are arranged from left to right in order of increasing topological depth. The increased number of fixpoint 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 9. 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 comparable computational effort.

4 Discussion and Conclusion

Reachability expressions give the user the ability to specify several 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.

NuSMVDP is a novel tool that allows user specified scheduling of transition clusters through reachability expressions. SMART is another tool with similar features, and it implements the saturation algorithm discussed in [9]. Unlike NuSMVDP, which uses standard BDD representations of transition relations, SMART uses multi-way decision diagrams (MDD) and Kronecker operators to encode the next state function. The Kronecker encoding helps in exploiting locality of effect of events on state variables. Both NuSMVDP and SMART have efficient implementations for identity transitions. In SMART, the identity transitions are ignored in the corresponding Kronecker matrix. In NuSMVDP, if a state variable s is not affected by a transition cluster τ_i, s is assumed to undergo the identity transition when image under τ_i is computed. When computing this image using BDDs, NuSMVDP does not existentially quantify s at all; instead, it is renamed to s', indicating that the variable underwent an identity transition. This significantly improves the computation time and memory usage because quantification of variables using BDDs is an expensive operation.

In SMART, a transition relation is partitioned disjunctively along events and conjunctively along sub-models. This makes it possible to express the effect of firing an event α in a global state as the cross product of the local effects of α on each sub-model. The decomposition helps capture event locality and update sets of state variables in a manner that reduces peak memory usage. However, if there are M sub-models and E events, then we need to consider a total of $E \times M$ transition relations. A large number of transition clusters implies an increase in the number of iterations before fixpoint is reached in reachability analysis. Thus, a large number of clusters could indeed increase the computation time. This issue is addressed to a certain extent in [10] where some transitions can be combined to reduce the number of clusters. The problem can be similarly addressed in NuSMVDP as well by selectively combining sub-models for certain events, thereby reducing the number of transition clusters to consider. However, any such combining strategy is likely to increase the sizes of BDDs that must now store the combined transition relations. Thus, although combining transition relations has been shown to work for some examples [10], one has to be very careful when combining or clustering transitions in practical applications. In general, the performance of the search may depend crucially on the way transition relations are combined. Such strategies must therefore be evaluated on a case-by-case basis before using them in an application. A tool like NuSMVDP is particularly useful in this context, since alternative strategies can be implemented with very low cost, simply by feeding in different reachability expressions.

The motivation behind the saturation method in [9] is very similar to our motivation for taking fixpoint with respect to local transition clusters in NuSMVDP. As we have shown in our comparison of schedules S_3 and S_4, taking a local fixpoint is, however, not guaranteed to perform well. If the representation of the cluster is large and the cluster has a large support set of variables, as in the time transition cluster T_t in our examples, it may be better to avoid local fixpoint computations with respect to this transition cluster. This indicates that the effectiveness of the saturation method and also of our approach depends on the decomposition of the transition relation, and also on the strategy used for each individual transition cluster.

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 with chaining 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 circuits with a large number of layers of gates, wherein there is considerable forward propagation of events.

The experiments and their analysis reported in this paper, in conjunction with earlier work in the SMART project [9] and on chaining-based techniques [14], clearly demonstrate that symbolic reachability can be significantly speeded up by employing appropriate search strategies. While we have discussed some results that allow us to understand why certain strategies are expected to perform better than others, the actual performance depends on a large number of other factors like cache performance, BDD library used, etc., that have been ignored in our theoretical discussion. Despite this omission, we observe that carefully chosen strategies lead to performance improvements, as predicted. This demonstrates the importance of reasoning about alternative sequences of image computation operations (or strategies) in making symbolic reachability scale to larger examples. Reachability expressions provide a uniform way of expressing, reasoning about and implementing a large variety of such strategies. We believe that these 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 International Conference on Computer-Aided Verification*, volume 2727 of *Lecture Notes in Computer Science*, pages 122–125. Springer, 2003.
2. R. Bloem, K. Ravi, and F. Somenzi. Symbolic guided search for CTL model checking. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 29–34. ACM Press, 2000.
3. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.
4. J. R. Burch, E. M. Clarke, K. L. McMillan, and D. L. Dill. Sequential circuit verification using symbolic model checking. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 46–61. ACM Press, 1990.
5. G. Cabodi, P. Camurati, L. Lavagno, and S. Quer. Disjunctive partitioning and partial iterative squaring: An effective approach to symbolic traversal of large circuits. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 728–733. ACM Press, 1997.
6. 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 International Conference on Computer-Aided Design*, pages 293–298. IEEE Press, 2001.
7. P. Chauhan, E. M. Clarke, S. Jha, J. Kukula, H. Veith, and D. Wang. Using combinatorial optimization methods for quantification scheduling. In *Proceedings of the Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME)*, volume 2144 of *Lecture Notes in Computer Science*, pages 293–309. Springer, 2001.
8. M.-Y. Chung, G. Ciardo, and A. J. Yu. A fine-grained fullness-guided chaining heuristic for symbolic reachability analysis. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis*, volume 4218 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2006.
9. G. Ciardo, R. Marmorstein, and R. Siminiceanu. The saturation algorithm for symbolic state-space exploration. *International Journal on Software Tools and Technology Transfer*, 8(1):4–25, 2006.
10. G. Ciardo and A. J. Yu. Saturation-based symbolic reachability analysis using conjunctive and disjunctive partitioning. In *Proceedings of the Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME)*, volume 3725 of *Lecture Notes in Computer Science*, pages 146–161. Springer, 2005.
11. 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 checking. In *Proceedings of International Conference on Computer-Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364. Springer, 2002.
12. 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 IEEE Symposium on Logic In Computer Science*, pages 1–33, 1990.
13. 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 Internationl Conference on Computer-Aided Design*, pages 388–393. IEEE Press, 1997.
14. O. Roig, J. Cortadella, and E. Pastor. Verification of asynchronous circuits by BDD-based model checking of petri nets. In *Proceedings of the International Conference on Application and Theory of Petri Nets*, volume 815 of *Lecture Notes in Computer Science*, pages 374–391. Springer, 1995.
15. 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.
16. 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.