Enabling Efficient Partial Order Reductions for Model Checking Object-Oriented Programs Using Static Calculation of Program Dependences

Venkatesh Prasad Ranganath[1], John Hatcliff[2], and Robby[2]

[1] venkateshprasad.ranganath@gmail.com
[2] Department of Computing and Information Sciences
Kansas State University
{hatcliff,robby}@ksu.edu

Abstract. In our previous work, we developed partial order reduction techniques that now form the core reduction strategies of several popular Java model checking frameworks including Bogor and JPF. These techniques work by *dynamically* collecting independence information needed to drive POR. In this paper, we consider several additional variants of partial order reductions and how independence information needed to drive these reductions can be calculated *statically* via a collection of Java analyses techniques including points-to, escape analysis, and smart call-graph construction. As part of this investigation, we also give a stateful implementation of Flanagan and Godefroid's dynamic POR framework capable of handling cyclic systems. We show that (a) the performance of these variants compares favorably in many cases to existing state-of-the-art techniques and (b) the variants suggest opportunities for further optimizations. In addition, our results give an alternative implementation strategy in which the gathering of dependence information can be lifted out of the model checking engine and performed in a separate static analysis phase.

1 Introduction

Partial order reduction (POR) is one of the primary optimization techniques for model checking concurrent systems. Central to POR is the notion of a statement/transition independence relation which captures situations in which two program statements from different threads are *non-interfering* – a model checking algorithm can safely avoid exploring some interleavings of executions of non-interfering statements. POR techniques are often challenging to apply for model checking software written in object-oriented (OO) languages. OO language features such as heap-allocated data, aliasing, and dynamically created threads make it difficult to determine when pairs of program statements are guaranteed not to interfere.

In previous work [1], we developed partial order reduction techniques that now form the core reduction strategies of several popular Java model checking frameworks including Bogor [2] and JPF [3]. These techniques work by *dynamically* collecting independence information needed to drive POR. In [1], we also explored calculating independence information using static escape analysis (to determine when transitions only accessed objects that were not shared between threads), but experimental results showed that dynamic calculation of independence information in the model checking engine itself yielded significantly better performance.

Since [1] was published, we have significantly enhanced our static analysis capabilities in a variety of directions. Most relevant to our present work, we have implemented interesting forms of points-to, escape analysis, and smart call-graph construction that can be used to identify groups of Java statements from multiple threads that are "coupled" due to potential interactions on the same shared heap objects. These analyses are implemented in our Indus framework [4] for mining program dependences and slicing concurrent Java programs.

In this paper, we consider several variants of partial order reductions for concurrent OO systems and we explore how independence information needed to drive these reductions can be calculated using the enhanced static analysis capabilities described above. This represents an interesting research direction since very little work has been done on using static analysis of concurrent OO programs to drive POR techniques, even though simple forms of static analysis are the basic source of independence information for POR in tools like SPIN [5].

The specific contributions of this paper are as follows:

- We describe how our static analysis framework can be used to drive a POR strategy based on conditional stubborn sets (CSS) [6]. Traditionally, conditional stubborn sets have been viewed as difficult to compute because they require reasoning about possible interference of transitions that may occur in the future. Yet, we provide experimental studies that show that our approach is practical in that in most cases it improves on our previous approach [1] that forms the basis of the JPF and Bogor POR implementations. Moreover, we show that combining this CSS approach with that of [1] yields additional improvements.
- We extend Flanagan and Godefroid's dynamic POR (DPOR) framework [7] that works only for stateless acyclic systems to work for stateful checking of systems whose state space may be cyclic.[3] We provide experimental studies that show that our previous POR approach [1] and our proposed stateful DPOR approach are complementary in the sense that their combination gives better performance than the individual approaches alone.
- Considering much of the work of POR for software system has focused on extending the model checking engine to dynamically calculate independence information, our results give an alternative implementation strategy in which the gathering of dependence information can be lifted out of the model checking engine and performed in a separate static analysis phase. This can be especially useful in situations in which it is difficult or undesirable to modify an existing model checking engine (e.g., when translating OO programs into a framework like Spin). Moreover, since our experiments show that our static approach complements existing dynamic approaches, the techniques in this paper could be applied with relatively little effort to enhance the performance of other tools such as JPF.

The rest of this paper is organized as follows: Section 2 provides background information on POR, Section 3 describes how we use static analysis to compute CSS, Section 4 presents our extension of Flanagan and Godefroid's DPOR to handle stateful search of cyclic and acyclic state space, Section 5 presents experimental results (additional experimental data is given in a tech report on the Bogor web site[4]), Section 6 surveys related work, Section 7 describes few possible future directions, and Section 8 summarizes the contributions.

2 Background

2.1 State Space

A program P is a set of threads R along with a set of global variables G. Each thread r is a 1-offset finite sequence of commands along with a set of local variables V. The offset value of each command is called the *program counter* of the command. $C = \bigcup_{r \in R} r$ is the set of commands in the program. The threads in a program are command disjoint.

[3] In [7], the authors note that "it is not obvious how to combine DPOR with stateful search".
[4] http://bogor.projects.cis.ksu.edu

```
SelectiveSearch(S_0)
1  visited ← ∅
2  for each s_0 ∈ S_0 \ visited
3  do visited ← visited ∪ {s_0}
4       visited ← DFS(s_0, visited)
5  return

DFS(s, visited)
1  selected(s) ← Select(s)
2  while selected(s) ≠ ∅
3  do α ← remove(selected(s))
4       s' ← α(s)
5       if s' ∉ visited
6          then visited ← visited ∪ {s'}
7               visited ← DFS(s', visited)
8  return visited
```

Fig. 1: The selective search algorithm (SSA). *remove* deletes and returns an arbitrary element from the given set.

A *state* $s \in \mathcal{S}$ is a $(1+|R|)$-ary tuple $\langle \mathcal{G}, (\rho_1, \mathcal{V}_1), (\rho_2, \mathcal{V}_2), \dots \rangle$ where $\mathcal{G}$ maps each global variable to a value and the thread local state of each thread r_i is represented by $(\rho_i, \mathcal{V}_i)$ in which $\mathcal{V}_i$ maps each local variable in r_i to a value and $\rho_i : \mathbb{N}$ is the (program counter) index of the command to be executed next by r_i. If thread r_i has completed execution, then $\rho_i = |r_i| + 1$.

Each command α is interpreted as a relation on $\mathcal{S} \times \mathcal{S}$. For simplicity (and following convention [6, 8]), we assume that each command is deterministic and thus forms a function on states – the non-determinism within a program is limited to scheduling non-determinism (this matches the semantics of Java).

The *state space* Δ^P of program P is defined as a tuple $\langle \mathcal{S}, \mathcal{T}, \mathcal{S}_0 \rangle$ where $\mathcal{S}$ is the set of states of P, $\mathcal{T} \subseteq \mathcal{S} \times C \times \mathcal{S}$ is the set of labeled transitions connecting the states of P, and $\mathcal{S}_0$ is the set of states from which P begins execution depending on the inputs. A transition $t = (s, \alpha, s')$ denotes the execution of the command α in state s leading to state s', and such a transition exists iff $\alpha(s) = s'$. $current(s)$ denotes that set of commands positioned at a program counter in s, i.e., the set of commands that could possibly execute at s. $enabled(s) \subseteq current(s)$ denotes the set of enabled commands in s. Commands in $current(s)$ but not in $enabled(s)$ are *disabled*. For example, disabled commands include `java.lang.Object.wait()` statements waiting on notifications, `synchronized` statements blocked while waiting to acquire a lock, and `java.lang.Thread.join()` statements waiting on the termination of a child thread.

A *path* π_s in Δ^P is a 1-offset alternating sequence of states and commands that begins with s and ends with a state and, for every consecutive s_i, α, and s_j in the sequence, $(s_i, \alpha, s_j) \in \mathcal{T}$. The set of all s-rooted paths is denoted by Π_s. The *command projection* π^C of path π is a 1-offset sequence of commands such that $\pi^C[i] = \pi[2i]$. The set of all commands reachable from s is denoted by $reachable_C(s) = \{\alpha \,|\exists \pi \in \Pi_s . \alpha \in \pi^C\}$.

For a program P, each s_0-rooted path in Δ^P represents an execution trace of P. Hence, every possible behavior of P can be explored by exploring every s_0-rooted path in Δ^P.

Selective Search Algorithm (SSA) Given the program P, the parametric algorithm in Fig. 1 explores Δ^P by performing depth first search. The Select parameter influences the search by selecting a nonempty subset of enabled commands to be explored at a given state; in state s, this subset is referred to as the *selected set* and denoted as $selected(s)$.

The choice of the definition of SELECT can be driven by different goals (e.g. explore all possible paths, explore at least one path from every equivalence class of paths, explore based on certain heuristics, etc). As a trivial example, exhaustive state space exploration can be performed by instantiating SSA (as eSSA) with SELECT that returns the $selected(s) = enabled(s)$ for every visited state s. For optimization, one typically seeks to have the select set be as small as possible while maintaining enough elements in the set to guarantee a particular correctness criteria. Two instantiations of SSA that attempt to optimize the search by only exploring paths to a particular notion of equivalence are described in Section 3.

2.2 Partial Order Reduction (POR)

Partial order reduction strategies for optimizing state space search are generally based on a notion of *command dependence* (which can also be cast as *transition dependence*). We base our work on the following notion of *conditional dependence* (i.e., state-sensitive dependence) as defined by Godefroid [6, p. 36]. [5] This definition defines conditions on a valid *dependence relation* $\mathcal{D}$ between commands at a particular state.

Definition 1 (dependence relation conditions). *Let C be the set of commands for program P and $\mathcal{D} \subseteq (C \times C \times \mathcal{S})$. The relation $\mathcal{D}$ is a valid dependency relation for P iff for all $\alpha_1, \alpha_2 \in C, s \in \mathcal{S}$, $(\alpha_1, \alpha_2, s) \notin \mathcal{D}$ implies $(\alpha_2, \alpha_1, s) \notin \mathcal{D}$ and the following two properties hold in s,*

- Independent commands can neither enable nor disable each other: *If $\alpha_1 \in enabled(s)$ and $\alpha_1(s) = s'$ then $\alpha_2 \in enabled(s)$ iff $\alpha_2 \in enabled(s')$.*
- Commutativity of enabled independent commands: *If $\alpha_1, \alpha_2 \in enabled(s)$ then there is a unique state s' such that $\alpha_1(\alpha_2(s)) = s' = \alpha_2(\alpha_1(s))$.*

It is convenient to refer to the *independence relation* $\mathcal{I}$ that is defined as the complement of $\mathcal{D}$.

Although we will not give a formal definition of trace equivalence (such a definition can be given based on the notion of Mazurkiewicz trace equivalence [10] as in [6]), we will simply state that, given Δ^P and $\mathcal{D}$, *two paths in Δ^P are equivalent if they can be obtained from each other by successively permuting adjacent independent commands.* The technique of avoiding the exploration of equivalent paths in a state space is referred to as *partial order reduction*. The subgraph of Δ^P containing only the states and the transitions explored by the effect of partial order reduction is referred to as *reduced state space ($\Delta^P_{\mathcal{D}}$)*.

In concurrent programs, properties of dependence are tied to inter-process communication/interaction, and in the case of Java this manifests itself as different threads interacting with shared heap data or static (global) object fields. For example, in a Java program, a concurrent read operation depends on a concurrent write operation when the operations share the operand.

3 Static Program Dependence-based Conditional Stubborn Sets (SPD-CSS)

3.1 Static Program Dependences

In program understanding scenarios, it is desirable to query for program points that influence or are influenced by the control/data aspect of program points under scrutiny (e.g., identify the write operations

[5] This notion is similar to the notion of *detecting conditional dependences using collapses* proposed by Katz and Peled [9].

that affect a given read operation). Such influential relations between various program points are commonly referred to as *program dependences*.

Accurate program dependences can be *dynamically* calculated based on all possible executions of programs; however, most often there are an exponential number of possible executions for non-trivial programs that leverage language features such as procedures, object orientation, dynamic memory allocation, and concurrency. As a scalable (but less accurate) alternative, program dependences can be *statically* calculated based on a sound abstraction of all possible executions of such programs.

In an earlier effort [11, 12], we proposed an equivalence class based analysis to statically calculate information to help identify static program dependences in concurrent Java programs. This analysis is implemented as part of our Indus analysis program slicing framework [4]. Specifically, the analysis information aided in identifying the following two kinds of inter-thread program dependences.

Read-write and write-write program dependences: When one thread can execute a field (array cell) read (write) operation on an object (array) while another thread can execute a field (array cell) write operation on the same object (array), the outcome depends on the order in which these operations are executed.

Synchronization based program dependences: The outcome of a lock acquisition operation by a thread depends on the lock release operation by a thread holding the lock. Similarly, the outcome of a wait operation (`java.lang.Object.wait`) by a thread depends on the notification operation (`java.lang.Object.notify`) by another thread.

To identify the above dependences, the equivalence class based analysis calculates the following types of information:

1. *Thread-local objects (universal statement independence)* – the analysis identifies objects that never escape the creating thread context (they are never shared); reads/writes and synchronization on such objects can never give rise to dependences with actions of other threads, and
2. *Operation coupling (binary statement dependence)* – if an object can be shared, the analysis identifies specific situations in which dependences can arise by collecting specific pairs of read(write)-write operations or synchronization operations that may operate concurrently on a shared object (we refer to this notion as *(read/write-write, lock, wait-notify) coupling*).

The analysis proceeds in two major phases. In the first phase, the analysis builds a sound approximation of the object graph in terms of the variables (references to objects) in each procedure in the program. The analysis also records read/write and synchronization operations in the nodes associated with the involved variables. The parts of the object graph reachable from the parameters to the procedures are propagated and combined with the object graphs in the calling procedures at various procedure invocation sites. This propagation captures the escaping of objects from their creating procedures via object subgraphs rooted at parameters. Similarly, it exposes the operations on objects shared between procedures. If the invoked procedure corresponds to a thread creation operation (`java.lang.Thread.start()` in Java), the appropriate nodes (objects) in the combined object graph are marked as escaping. Similarly, while combining two nodes with appropriate recorded operations (e.g., read on one node and write on another), the corresponding node in the combined object graph is marked as participating in possibly concurrent execution of the recorded operation. This construction and propagation is done in a bottom-up order on the call graph of the input program.

In the second phase, the analysis propagates the previously calculated and recorded information from the combined object graph nodes in the calling procedures into the corresponding object graph nodes in the

called procedures at various invocation sites. The propagation proceeds in the top-down direction in the call graph of the input program.

The analysis framework also enables calculation of a *mayfollow* $\triangleright$ relation on commands such that $\alpha_1 \triangleright \alpha_2$ when α_2 is reachable from α_1 via a control flow path in P. This relation is calculated by combining the unified call graph of the program (i.e. a combination of the call graphs of the threads in the program) with the intra-procedural control flow graphs of the program. The *mayfollow* relation is a sound approximation of execution paths such that whenever there exists a path π in Δ^P with command projection $\pi^C = \alpha_1, \ldots, \alpha_2$, it is the case that $\alpha_1 \triangleright \alpha_2$ holds.

Please refer to [11, 12] for implementation details and empirical evaluation of the analysis. An open source implementation of the analysis is available as part of the Indus distribution.

3.2 Conditional Stubborn Sets (CSS)

In his dissertation [6], Godefroid noted that several existing partial order reduction algorithms selected subsets from the set of enabled transitions that had a particular set of properties that guaranteed the overall correctness of each algorithm. He referred to such subsets as *persistent sets* PS. A set of enabled commands PS in a state s is *persistent in* s iff for all non-empty sequences of commands from s in Δ^P

$$s = s_1 \stackrel{\alpha_1}{\rightarrow} s_2 \stackrel{\alpha_2}{\rightarrow} s_3 \cdots \stackrel{\alpha_{n-1}}{\rightarrow} s_n \stackrel{\alpha_n}{\rightarrow} s_{n+1}$$

including only commands $\alpha_i \notin PS$, $1 \leq i \leq n$, α_n is independent in s_n with all commands in PS.

Godefroid presented three existing algorithms to calculate different realizations of persistent sets based on the notions of *conflicting transitions [13–15]* and Valmari's *stubborn sets [16]*. He then proposed an improved (less restrictive) set selection criteria for computing a persistent set known as *conditional stubborn set*.

A set of commands *CSS* is a conditional stubborn set in state s if *CSS* contains at least one enabled command, and if for all commands α in *CSS*, the following condition holds:

> for all sequences $s = s_1 \stackrel{\alpha_1}{\rightarrow} s_2 \stackrel{\alpha_2}{\rightarrow} s_3 \cdots \stackrel{\alpha_{n-1}}{\rightarrow} s_n \stackrel{\alpha_n}{\rightarrow} s_{n+1}$ of commands from s in Δ^P such that α and α_n are dependent in s_n, at least one of the $\alpha_1, \cdots, \alpha_n$ is also in *CSS*.

Although conditional stubborn sets improved on the other persistent set algorithms (i.e. it yielded smaller selected sets), Godefroid acknowledged that it was difficult to come up with a practical algorithm to detect when future transitions from a given state might by dependent on transitions in the current stubborn set. In this section, we propose a practical approach based on static analysis to calculate conditional stubborn sets in the context of programs written in Java-like languages.

Regarding correctness, Godefroid proved that the reduced state space resulting from the application of persistent sets (in all forms – including conditional stubborn sets) preserved deadlocking behaviors. We rely on these proofs to establish the correctness of our approach. Preservation of general safety properties requires the addition of other algorithmic features to avoid the "ignoring problem" (we omit presentation of these issues since they are orthogonal to our concerns in this paper).

3.3 The Approach

Given the above definition of CSS, the information from the analysis in Section 3.1 can be used to realize a simple algorithm to construct conditional stubborn sets. The realization is broken down into two phases.

```
CSS-SELECT(s)
 1  α ← choose(enabled(s))
 2  result ← {α}
 3  if ISINDEPENDENT(α)
 4     then return result
 5  tmp1 ← enabled(s)\result
 6  for  each β ∈ tmp1
 7  do if ISINDEPENDENT(β)
 8        then return {β}
 9        else  for  each α′ ∈ result
10              do if ISDEPENDENTON(α′, β) ∨
11                    ISDEPENDENTONSUCCS(α′, β)
12                    then result ← result ∪ {t}
13  return result
```

Fig. 2: A conditional stubborn set calculating parameter of the selective search algorithm. *choose* function selects an element from the given set.

Phase 1 For a given Java program, the following predicates are constructed directly from the previously described equivalence class based analysis information.

IsIndependent: $C \rightarrow \{\textbf{\textit{true}}, \textbf{\textit{false}}\}$ holds true when the given command is independent of all commands in all states from other threads (universally independent). This captures situations in which the given command operates only on non-escaping objects or on method-local scalar data.

IsDependentOn: $C \times C \rightarrow \{\textbf{\textit{true}}, \textbf{\textit{false}}\}$ holds true when the two commands are coupled via lock coupling, read-write and write-write coupling, and wait-notify coupling as described in the previous section (this relation is symmetric as we take the symmetric closure of the analysis information).

IsDependentOnSuccs: $C \times C \rightarrow \{\textbf{\textit{true}}, \textbf{\textit{false}}\}$ Let α_1 and α_2 be the first and second arguments to the predicate. The predicate holds true when there exists a command α_2' such that $\alpha_2 \triangleright \alpha_2'$ and ISDEPENDENTON(α_1, α_2').

Phase 2 Using the above information, a SELECT parameter for calculating CSS is instantiated following the algorithm in Fig. 2. The algorithm begins by non-deterministically choosing an command α enabled at s as a candidate CSS held in the variable *result*. In the ideal case, *IsIndependent*(α) holds (i.e., α is universally independent). For such cases, there is no need to consider possible interference with commands in other threads reachable from the current state s since α is independent of all other commands. If an enabled universally independent command is not found for s, the algorithm iterates through the remaining enabled commands β at s held in variable *tmp1*. If any β if found to be universally independent, a singleton containing β is immediately returned as the CSS. Otherwise, the algorithm considers each command α' from the candidate set *result* against each remaining β. For each such pair of commands, the algorithm checks if α' is dependent directly on β or dependent on a command that may follow β in a execution trace leading from s. If such a dependence is found, β is added to the result set.

Our previous POR work [1] that forms the basis of the algorithms implemented in Bogor and JPF only leverages *IsIndependent* information. Thus, for commands that were not universally independent, we made the worse case assumption that they were universally dependent. The approach presented above leverages static analysis information to safely remove assumed but overly-conservative dependences for non-universally-independent commands (significantly improving upon the previous worse-case universally dependent assumption) using the *IsDependentOn* coupling information. While we believe the strategy of leveraging

static analysis could be applied to other strategies/frameworks for computing persistent sets, we found the structure of the analysis information most naturally fit within the CSS framework. Note that our approach above does not take advantage of *conditional* (i.e., state-sensitive) (in)dependence (it is relatively difficult to do this with a totally static approach since information about specific state values is summarized), and this presents an additional opportunity for reduction by blending static and dynamic approaches (where dynamically gathered information can generate state-sensitive independences). Additional perspective is given in Section 6.

Correctness Due to the correctness of our static analysis, *IsDependentOn* and *IsIndependent* both give rise to two valid dependence relations $\mathcal{D}_{IDO}$ and $\mathcal{D}_{II}$. $\mathcal{D}_{IDO}$ is defined as $(\forall s \in \mathcal{S}.\mathcal{D}_{IDO}(\alpha_1, \alpha_2, s)) \Leftrightarrow \mathit{IsDependentOn}(\alpha_1, \alpha_2)$. $\mathcal{D}_{II}$ is the symmetric closure of the command arguments of the relation $\mathcal{D}_{II}^{pre}$ where $(\forall s \in \mathcal{S}.\forall \alpha_2 \in C.\mathcal{D}_{II}^{pre}(\alpha_1, \alpha_2, s)) \Leftrightarrow \neg \mathit{IsIndependent}(\alpha_1)$.

For establishing that the algorithm actually produces a CSS, it is trivially correct when there exists an independent enabled command (lines 3-4 and 7-8). In other cases, when the CSS contains more than one element, each enabled non-CSS command β that is dependent on a command α' in the CSS (line 10) is added to CSS. Similarly, each enabled non-CSS command β is added to the CSS if a CSS command α' is dependent on a successor of β (line 11). Together these cases fulfill the conditions required of a CSS.

4 Stateful Dynamic POR (SDPOR)

Constructing effective POR strategies involves the challenging problem of detecting situations in which current transitions are guaranteed not to interfere with transitions *in the future* of system execution. Flanagan and Godefroid [7] proposed a strategy they referred to as "dynamic partial order reduction" (DPOR) to deal with this challenge. DPOR actually involves two different notions of dynamism. First, as with our earlier work [1], their algorithm extends the state space exploration algorithm to dynamically check for independence of transitions. Second, the most important idea behind their technique was to aggressively construct the most optimal selected set (a singleton) and then to dynamically expand the selected set by injecting unselected enabled commands when deemed necessary based on dynamically discovered transition dependences. The dynamic discovery relied on the maintenance of the path from the initial state s_0 to the current state s. For each explored transition t (in thread r), its dependence on a previously explored transition t' in the current path is calculated; if dependent, then an enabled command from thread r at s' is injected into $\mathit{selected}(s')$. If there are no transitions from thread r at s', then all un-selected enabled commands at s' are injected into $\mathit{selected}(s')$. Each injected command is processed when the exploration backtracks to s'. This technique was aimed at stateless state space exploration of acyclic state space.

In this section, we present a simple adaptation of DPOR that we refer to as Stateful Dynamic POR (SDPOR) that is capable of handling cyclic state spaces.

4.1 The Algorithm

An extended version of the selective search algorithm and the SELECT parameter needed to realize our stateful DPOR-based selection are given in Fig. 3 and Fig. 4, respectively.

In E-DFS (Fig. 3), for every explored transition t, EXPANDSELECTEDSET parameter is called to expand the selected set of states that occur on the current exploration path. For this purpose, the explored transition and the emanating state are captured in *stack*.

```
E-SelectiveSearch(S_0)
 1  visited ← ∅
 2  for each s_0 ∈ S_0\visited
 3  do visited ← visited ∪ {s_0}
 4     stack ← ⊥
 5     visited ← E-DFS(s_0, visited, stack)
 6  return

E-DFS(s, visited, stack)
 1  selected(s) ← Select(s)
 2  while selected(s) ≠ ∅
 3  do α ← remove(selected(s))
 4     (s, _, s') ≡ t ← α(s)
 5     ExpandSelectedSet(t, stack)
 6     if s' ∉ visited
 7        then visited ← visited ∪ {s'}
 8             push(stack, ⟨t, selected(s)⟩)
 9             visited ← E-DFS(s', visited, stack)
10             pop(stack)
11  return visited
```

Fig. 3: An extended version of the selective search algorithm (Fig. 1). The changes to the original algorithm are underlined.

In DPOR-ExpandSelectedSet (Fig. 4), the current transition or transitions reachable from a fully explored state are selected (in lines 4-6) as candidates to detect dynamic dependences. For each such candidate transition t', the algorithm checks for dependence between the transition t' and every transition t'' on the current path. If a dependent preceding transition t'' is found, then the selected set of originating state s_l of t'' is expanded as in DPOR.

Implication of Statefulness In stateless state space exploration, fully explored states may be revisited as they are not "remembered" to avoid re-exploration. Hence, such re-exploration correctly forces the dynamic expansion of the selected sets. In stateful state space exploration, every fully explored state is remembered to avoid re-exploration; hence, in a naive adaptation of DPOR algorithm to a stateful setting, upon arriving at a fully explored state s, mutual dependences between the transitions reachable from state s and the transitions leading to state s are not considered; hence, selected sets are (incorrectly) not dynamically expanded leading to incorrect and over-aggressive reduction.

To address this issue, at each fully explored state, the SDPOR algorithm stores the transitions reachable from a state along with their executing threads. Upon arriving at a fully explored state, the algorithm retrieves the reachable transitions to dynamically discover dependences[6] and expand selected sets at ancestor states; hence, the SDPOR algorithm will induce correct reductions as it simulates the behavior of stateless DPOR algorithm.

Correctness Argument The correctness of the above algorithm relies on the correctness of discovering dynamic dependences involving transitions reachable from fully explored states. As the transitions and states reachable from a fully explored state s are fixed and are independent of the path leading to s, it is safe

[6] This observation is encoded in line three of Fig. 4.

```
SDPOR-Select(s)
 1  return {choose(enabled(s))}

SDPOR-ExpandSelectedSet(t, stack)
 1  if IsIndependent(t)
 2     then return
 3  (s_i, _, s_j) = t
 4  if s_j ∈ visited ∧ ⟨(_, _, s_j), _⟩ ∉ stack
 5     then m ← getReachableTransitions(s_j)
 6     else  m ← {t}
 7  for  each t' ∈ m
 8  do (s_k, α, _) = t'
 9     for  each ⟨t'', selected(s_l)⟩ ∈ stack (in top-down order)
10     do if IsDependentOn(α, t''#2, s_l)
11           then s_l ← t''#1
12                β ← getCommandFor(getThreadFor(α, s_k), s_l)
13                if β ≠ ⊥
14                   then selected(s_l) ← selected(s_l) ∪ {β}
15                   else  selected(s_l) ← enabled(s_l)
16                return
```

Fig. 4: A stateful dynamic POR/CSS realizing parameter of the extended selective search algorithm. *getReachableTransitions(s)* returns the reachable transitions from the fully explored state s; an empty set is returned if s is not fully explored. *getThreadFor*(α, s) returns the thread executing transition α in state s. *getCommandFor*(r, s) returns the enabled command of thread r in state s, if any. The extended version of IsDependentOn dynamically detects if the given transitions are mutually dependent in the given state.

and correct approximation[7] to consider reachable transitions to discover the dynamic dependences. This is a partial correctness argument, and we shall address partialness via FESC in Section 4.2.

4.2 Extensions to SDPOR

In this section, we present a compensatory technique to address over-aggressive reduction stemming from ignored transitions, a trivial optimization for DPOR, and two alternative realizations of SDPOR.

Full Enabled Set Coverage (FESC) Consider the state space in Fig. 5 (a). State s_i occurs on the current path $\pi_i = s_i \dots s_j$ (the dashed edge) to s_j and there is a path $\pi_j = s_j \dots s_i$ (the solid edge). In the above SDPOR algorithm, the transitions in the shaded area reachable from state s_i will not be considered to expand the selected sets at states in the cycle formed by splicing π_j and π_i. This can lead to over-aggressive reductions.

In case of cyclic state space (as illustrated in Fig. 5 (b)), suppose $t_\alpha = (s_i, \alpha, \_)$ and $t_\beta = (s_i, \beta, \_)$ are dependent transitions in different threads at state s_i. Let the shaded area denote the state space reachable via t_α. If t_β is chosen as the only member of $selected(s_i)$, then t_α should be injected into $selected(s_i)$ upon exploring (s_j, α, s_k). However, it is possible that (s_j, γ, s_j) will arbitrarily selected and explored. Consequently, $(s_j, \alpha, \_)$ will not be explored; hence, the $selected(s)$ will (incorrectly) not be expanded to explore the shaded

[7] It is an approximation as it allows for transitions t" to dynamically depend on t even when there exists an intermediate t' along the path from t to t" such that t" dynamically depends on t'.

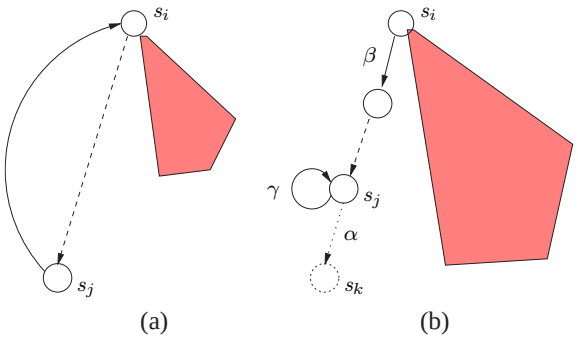

Fig. 5: State spaces that warrant FESC corrections. The nodes represent the states, the solid edges represent transitions, the dashed edges represent the current path between the represented states, the dotted edges and nodes represent unexplored transition and states, and the shaded area represents the ignored part of the state space.

area of the state space. A similar situation can occur in an acyclic state space when all paths starting with the selected transitions from a state end in error states.

This issues of *neglected transitions phenomenon* can be addressed by *exploring every enabled command at a state along a path from that state*.[8] To accomplish this, the algorithm maintains a set of non-selected enabled commands at each state being explored. Upon backtracking into a state s after exhausting $selected(s)$, the algorithm checks if every non-selected enabled command were executed along an explored path from s. If not, then an arbitrary non-selected enabled command α is selected (injected into $selected(s)$) and the state space reachable via $(s, \alpha,)$ is explored. This process is repeated until each enabled command at s is explored. Hence, during exploration, a state is marked as fully explored (visited) and backtracked from only after every enabled command at that state has been explored.

Dependence-based Equivalence Classes (DEC) In SDPOR, the current command is processed against every preceding command in the current execution path. This is sub-optimal as it includes situations such as checking for dependence between an array access operation and a lock operation. Given the incompatibility of these operations (they can never interfere), we can resort to incompatibility detection to avoid unnecessary processing. As each command be classified based on the sort (locking, read-write, wait-notify) of program dependence it can trigger, it suffices to process the current command against every command in the dependence sort restriction of the command projection of the current path.

Although this optimization only reduce the complexity by a constant factor, this optimization will in practice reduce the overall cost in case of large state spaces and long exploration paths.

Pure Dynamic Dependences (PDD) As the exact state of the program is available during state space exploration, a purely dynamic approach based on the state information can be used to dynamically calculate accurate transition dependence. Based on the programming language semantics, the approach relies on comparing the appropriate parts of commands to detect dependences. In the context of Java programs, two commands are dependent if any of the program dependences listed in Section 3.1 occurs at runtime.

To enable the above detection, the algorithm maintains the variable to value binding for variables in transitions associated on field/array access, wait/notify invocation, and lock acquisition commands. Similar

[8] This is similar to C3 proviso in the ample set POR framework [8].

to reachable transitions maintained to be statefully correct, the variable to value binding is maintained for each fully explored state. While this provides high level of accuracy, the space required to maintain variable to value binding information is proportional to the size of the explored state space; hence, this approach will be memory intensive in case of programs with large state space. The correctness of the algorithm based on this approach trivially follows as all of the dependences leading to alternative behaviors are considered.

As a minor optimization, static program dependence information can be leveraged to cheaply and quickly detect possible dependences followed by stronger dynamic information based validation.

5 Experimental Evaluation

We have experimentally validated the realization and application of the proposed POR techniques. The POR techniques were implemented in Bogor as modular scheduling strategies that could easily be combined with various existing optimizations such as thread and heap symmetry and collapse compression already implemented in Bogor [2, 17]. The static analysis was realized as part of the program dependence framework in Indus [4]. These realizations were combined in the Java program verification tool pipeline provided by Bandera [18].

5.1 Setup

In our experiments, we considered 13 programs from the Bandera examples repository as experimental subjects/inputs.[9] For brevity, we present data for five programs.[10]

Alarm Clock (AC1) A clock continually updates time and notifies clients registered for alarms.
Disk Scheduler (DS1, DS7) A disk scheduler is shared by disk readers concurrently read various cylinders on a disk.
Molecular Dynamic (MD3) A parallelized simulation of molecular dynamics.
Ray Tracing (RT3) A parallelized 3D ray tracing program.[11].
Replicated Workers (RP15) A realization of the common *replicated workers* concurrency design pattern.

Of these programs, only *AlarmClock* and *Disk Scheduler* programs contain errors (e.g. ArrayIndexOutOfBoundException and NullPointerException). For each program, we exercised the pipeline in the following nine configurations.

E is Bogor's current POR implementation – a highly optimal POR technique (EPOR) that relies on dynamic detection of independent transitions based on the non-reachability of shared objects [1] (certain aspects of the implementation that derive additional independence information based detecting properly locked objects are omitted to remove this factor from comparison with the algorithms presented in this paper which do not consider locking disciplines as a source of additional independences).
SC SPD-CSS was applied.
SC+E SPD-CSS and EPOR techniques were applied in order.

[9] We use the naming convention (e.g. AC1, DS1) from Bandera examples repository to aid pursuant readers.
[10] Please refer to Chapter 6 and Appendix B in Ranganath's dissertation [12] and to the extended version of this paper available on the Bogor web-site for the complete data set.
[11] MD3 and RT3 are available as part of Java Grande benchmark

SD SDPOR based on PDD was applied with DEC optimization.
E+SD EPOR and SDPOR (as in SD) were applied in order.
SC+F SPD-CSS was applied with full enabled set coverage.
SC+E+F SPD-CSS and EPOR were applied in order with FESC.
SD+F SDPOR (as in SD) was applied with FESC.
E+SD+F EPOR and SDPOR (as in E+SD) were applied with FESC.

We considered EPOR as the reference external technique in the evaluation as it was a most effective reduction technique that was available by default in the Bogor model checker and was based on independence information. In the configurations that combined techniques, we used EPOR to filter out non-existent dependences detected by static analysis and to identify transitions to be considered for dynamic dependence detection.

Besides the mentioned distinctions, the configurations were identical in terms of other components/sub-configurations (e.g. program slicing, counter-example serialization, thread and heap symmetry, collapse compression, etc) used and the end goal of detecting deadlocks in the input programs. The pipeline was executed on SUN JVM version 1.5.0 with maximum heap space of 14GB on Linux boxes with 16GB RAM. Executions that exceeded 10 hours were terminated. In the data tables, an asterisk (*) is noted for programs for which every execution was terminated.

5.2 Results

The data pertaining to total number of states, transitions, and errors encountered in various executions along with the maximum time and memory consumed by the executions are given in Tables 1a-1e.

Time The data in Table 1a indicates that SDPOR (SD) provides the relatively better time reduction when compared to SPD-CSS (SC) and EPOR (E) (in isolation) while the combination of SDPOR and EPOR (SD+E) provides the best time reduction. In case of FESC experiments, these observations hold true despite a decrease in reduction across all FESC experiments. Contrary to our intuition that the net effect of the combination of two POR techniques should be better than the effect of the POR techniques in isolation, the combination of SPD-CSS and EPOR (SC+E) contributes to a considerable increase in execution time when compared to EPOR in case of RT3 and RP15.

Memory Contrary to the previous observations, the data in Table 1b indicates that SDPOR is memory inefficient in comparison with SPD-CSS and EPOR both in non-FESC and FESC modes and SPD-CSS is most memory efficient in non-FESC mode. Due to numerous terminated FESC experiments and non-correlation between time and memory data, it is hard to conclude the memory efficiency of the techniques when FESC is employed. However, based on the FESC specific data for RT3, we conjecture that both SPD-CSS (SC+F) and SDPOR (SD+F) will be memory inefficient in comparison with their combinations with EPOR (SC+F+E, E+SD+F).

States and Transitions From the data in Table 1c, we observe that the number of explored states and transitions are mostly proportional to the execution time and, hence, the observation about execution time applies to states and transitions in both non-FESC and FESC modes. In cases such as RT3, contrary to this observation, the proposed techniques visit more states and less transitions than EPOR even while performing better than EPOR. This deviation is due to the accumulated cost of performing expensive dynamic escape analysis on relatively large number of transitions in EPOR.

Conf	*SC*	*SD*	*SC+E*	*E+SD*	E	*SC+F*	*SD+F*	*SC+F+E*	*E+SD+F*
AC1	10128	1574	7351	588	36013	36052	36065	36049	11151
MD3	36019	36247	5469	1564	36012	36081	36288	5497	1629
RT3	36011	13314	36014	1830	24263	36034	13287	36027	1827
RP15	14357	5956	16302	2619	13515	19593	6816	18419	2654

(a) Maximum time (seconds) taken by various executions.

Conf	*SC*	*SD*	*SC+E*	*E+SD*	E	*SC+F*	*SD+F*	*SC+F+E*	*E+SD+F*
AC1	2163	2228	3103	3124	4209	6178	6280	6905	7414
*DS1**	3075	12172	6221	14043	6579	8462	14268	13351	14300
*DS7**	2618	8645	4969	7226	3599	7247	7967	7358	7282
MD3	5937	14842	6135	5932	5890	10539	14407	6207	5966
RT3	5526	6381	5960	5607	5783	6942	6640	6666	5652
RP15	5119	13922	6405	7468	6382	12220	14390	10279	7954

(b) Maximum memory (MB) consumed by various executions.

Conf	*SC*	*SD*	*SC+E*	*E+SD*	E	*SC+F*	*SD+F*	*SC+F+E*	*E+SD+F*
AC1	161172	47522	78869	9240	256479	35922	35478	18928	8248
*DS1**	484320	280717	245323	154538	318652	107674	186871	170504	110420
*DS7**	802545	247030	136911	77217	157684	111268	74080	75774	43290
MD3	11908	9150	4419	2478	200622	10383	8463	4419	2478
RT3	1831	22909	2298	171	4616	1812	22909	2186	171
RP15	1043247	251698	568928	95657	612954	1043247	251698	568928	95657

(c) Total states encountered in various executions.

Conf	*SC*	*SD*	*SC+E*	*E+SD*	E	*SC+F*	*SD+F*	*SC+F+E*	*E+SD+F*
AC1	2617743	619812	1471300	149699	5599303	506774	425965	363324	125174
*DS1**	13130000	8420000	10761570	6858336	14480000	2980661	5605795	7540000	4895085
*DS7**	14401190	4104736	4603123	2522397	5143947	2597494	1721830	2576842	1417963
MD3	7251107	2762960	529098	103302	8287084	6303446	2105125	529098	103302
RT3	1455033	435667	711010	42604	1247000	1437594	435667	702675	42604
RP15	4823439	1047995	3259745	413817	5709125	4823439	1047995	3259745	413817

(d) Total transitions traversed in various executions.

Conf	*SC*	*SD*	*SC+E*	*E+SD*	E	*SC+F*	*SD+F*	*SC+F+E*	*E+SD+F*
AC1	5531	250	1982	77	14668	3650	3476	3727	1144
*DS1**	6922	3	448	0	0	984	0	0	0
*DS7**	10604	2490	4346	2409	4155	2679	1977	2228	2027

(e) Error counts detected in various executions.

Table 1: Data from the experiments.

Errors The error discovery data captured in Table 1e suggest that the proposed techniques uncover errors quickly both in isolation and in combination. Hence, the proposed techniques would be well suited to uncover situations (via counter-examples) that lead to known errors in the system.

Summary The experimental data supports our primary claims that (a) a purely static approach for detecting dependences can be competitive with and even improve upon existing approaches that require extending the model checker to detect dependences dynamically, (b) sophisticated static analyses can provide a basis for effective computation of conditional stubborn sets, and (c) a straightforward stateful adaption of Flanagan and Godefroid's DPOR is competive w.r.t. other techniques.

On the other hand, the success of each technique is sensitive to the structure of the individual examples as there is no consistent "winner" among the techniques. A very interesting line of work is the combination of these techniques; our initial exploration of this idea as reflected in the data above suggests that this approach can be quite fruitful, and we are pursuing a much broader empirical study to help determine which combination strategies are most beneficial.

6 Related Work

As mentioned in the earlier sections, Godefroid described a collection of POR techniques in his dissertation [6]. He acknowledged that conditional stubborn sets, the optimal of the available partial order reduction techniques, was purely of theoretical interest due to lack of a practical approach to calculate transitions reachable from a state. The proposed SPD-CSS technique leverages program analysis to address this limitation of conditional stubborn sets in a sound and relatively accurate manner.

Kurshan et al. [19] described an approach based on compile-time calculation of partial order reduction. Their approach was focused on ensuring the reduction techniques were non-intrusive of the state space exploration algorithm. Also, they focused on breadth first variant of the exploration algorithm. In comparison, SPD-CSS also relies on compile-time calculation of partial order relevant information but it modifies the exploration algorithm and is targeted towards the depth first variant of the exploration algorithm.

Stoller [20] described a collection of POR techniques focused on model checking Java programs. These techniques were based on common synchronization pattern in Java programs — *use of locks to protect shared variables*. The basic idea was to increase (and lengthen) the number of same-thread transition sequences by only allowing context switches before lock acquisition operations and not before access to variables protected by locks. These techniques relied on user input to appropriately classify objects as required by the techniques. Stoller's approach to leverage locking information inspired the locking-associated techniques we introduced in [1], and the automatic approaches (both static and dynamic) to detecting escaping objects of [1] significantly enhanced the performance and useablity of Stoller's ideas.

An alternate approach [21, 22] for detecting when transitions can be commuted is based on the notion of left- and right-movers as proposed in Lipton's Reduction Theory [23]. The basic idea is to detect and move transitions in an execution history to the increase (and lengthen) the number of same-thread transition (atomic) sub-sequences while honoring the ordering imposed by inter-thread communication. The mover approach has been used as an interesting reduction technique for model checking similar to POR, e.g., in [24]. However, mover-based strategies have not been extended to automatically detect leverage information about non-escaping objects and specific forms of coupling. Also, the basic correctness property used in mover-based model-checking reductions does not guarantee preservation of deadlock properties (extra mechanisms to guarantee liveness properties of atomic regions are needed). The static analysis that we have presented in this work can be applied to directly to optimize existing mover-based approaches.

Dwyer et al. [1] proposed a dynamic escape analysis based approach to efficiently calculate accurate independence information and effect partial order reduction. That work was based on Peled's *ample set* framework and followed the presentation in [8]. While the [1] approach is applicable in state space exploration frameworks where dynamic escape analysis can be easily implemented, the proposed approach (SPD-CSS) can be applied in any framework that supports selected set calculation. In the same effort, Dwyer et al. observed that the reduction due to independent transition detection based simple static escape analysis information was negligible in comparison with that based on dynamic escape information. In contrast, our present work shows that the reduction due to dependence detection based on sophisticated static analysis information (beyond escape information) is significant.

We have previously discussed the work of Flanagan and Godefroid [7] and Godefroid's dissertation [6] which inspired some of the key elements of our present paper.

7 Future Work

The strategies that we have proposed are direct in both the application of static dependences to realize CSS-based POR and the injection of statefulness into DPOR, and they do not incorporate a number of optimizations that could further reduce overall costs.

As the analytical and empirical data suggests that SDPOR can be memory intensive, it would be interesting to explore if any of the existing memory optimization techniques (such as collapse compression techniques) used in model checking could be used to lower the memory requirements of SDPOR.

While we have analytically compared our techniques to other techniques based on locking discipline and atomicity, it would be interesting to do an experimental/empirical evaluation to compare the techniques in isolation and combination and to identify ideal situations for their application. A much more elaborate empirical evaluation is needed to help identify situations that are best suited to employ each of the proposed techniques and their combinations. In particular, it may be possible to develop an approach in which an initial static analysis phase is used to indicate which POR strategy would likely be the most effective.

8 Conclusion

We have demonstrated that statically and efficiently calculated dependence information can be easily leveraged to realize efficient partial order reduction in the context of model checking object oriented programs. Also, we have proposed the first stateful variant of dynamic partial order reduction that is capable of handling cyclic state space. Finally, we have experimentally demonstrated that the proposed approaches and combinations are comparable to or better than existing POR approaches in terms of achieved reduction.

References

1. Dwyer, M.B., Hatcliff, J., Robby, Ranganath, V.P.: Exploiting object escape and locking information in partial-order reductions for concurrent object-oriented programs. Formal Methods in System Design **25** (2004) 199–240
2. Robby, Dwyer, M.B., Hatcliff, J.: Bogor: An Extensible and Highly-Modular Model Checking Framework. In: Proceedings of the Fourth Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE'03). (2003)
3. Brat, G., Havelund, K., Park, S., Visser, W.: Java pathfinder – a second generation of a Java model-checker. In: Proceedings of the Workshop on Advances in Verification. (2000)
4. Ranganath, V.P., Hatcliff, J.: Slicing Concurrent Java Programs using Indus and Kaveri. International Journal on Software Tools for Technology Transfer (STTT) (2007) Accepted for publication.
5. Holzmann, G.J.: The model checker SPIN. IEEE Transactions on Software Engineering (TOSE) **23** (1997) 279–294
6. Godefroid, P.: Partial Order Methods for the Verification of Concurrent Systems. PhD thesis, Facultè des Sciences Appliquèes, Universite De Liege (1995)
7. Flanagan, C., Godefroid, P.: Dynamic Partial-Order Reduction for Model Checking Software. In: Proceedings of Symposium on Principles of programming languages (POPL'05). (2005)
8. Edmund M. Clarke, J., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press (1999)
9. Katz, S., Peled, D.: Defining Conditional Independence Using Collapses. Theoretical Computer Science **101** (1992) 337–359 Selected papers of the International BCS-FACS Workshop on Semantics for Concurrency.
10. Mazurkiewicz, A.W.: Trace theory. In: Advances in Petri Nets. (1986) 279–324
11. Ranganath, V.P., Hatcliff, J.: Pruning Interference and Ready Dependences for Slicing Concurrent Java Programs. In Duesterwald, E., ed.: Proceedings of Compiler Construction (CC'04). Volume 2985 of Lecture Notes in Computer Science., Springer-Verlag (2004) 39–56
12. Ranganath, V.P.: Scalable and Accurate Approaches to Program Dependence Analysis, Slicing, and Verification of Concurrent Object Oriented Programs. PhD thesis, Kansas State University (2006) Available at http://hdl.handle.net/2097/248.

13. Godefroid, P., Wolper, P.: Using partial orders for the efficient verification of deadlock freedom and safety properties. In: Computer Aided Verification (CAV'91). (1991) 332–342
14. Godefroid, P., Pirottin, D.: Refining dependencies improves partial-order verification methods. In: Computer Aided Verification (CAV'93). (1993) 438–449
15. Overman, W.T.: Verification of Concurrent Systems: Function and Timing. PhD thesis, University of California, Los Angeles (1981)
16. Valmari, A.: Stubborn sets for reduced state space generation. In: Applications and Theory of Petri Nets. (1989) 491–515
17. Robby, Dwyer, M.B., Hatcliff, J., Iosif, R.: Space-Reduction Strategies for Model Checking Dynamic Systems. In: In Proceedings of the 2003 Workshop on Software Model Checking (SMC 2003). (2003)
18. Corbett, J.C., Dwyer, M.B., Hatcliff, J., Laubach, S., Păsăreanu, C.S., Robby, Zheng, H.: Bandera: Extracting Finite-state Models from Java source code. In: Proceedings of the 22nd International Conference on Software Engineering (ICSE'00). (2000) 439–448
19. Kurshan, R.P., Levin, V., Minea, M., Peled, D., Yenigun, H.: Static Partial Order Reduction. In: Proceedings of Tools and Algorithms for the Construction and Analysis of System (TACAS'98). (1998) 345–357
20. Stoller, S.: Model-checking multi-threaded distributed Java programs. International Journal on Software Tools for Technology Transfer **4** (2002) 71–91
21. Flanagan, C., Qadeer, S.: Transactions for Software Model Checking. Electronic Notes in Theoretical Computer Science **89** (2003)
22. Flanagan, C., Freund, S.N.: Atomizer: a dynamic atomicity checker for multithreaded programs. In: Proceedings of Symposium on Principles of programming languages (POPL'04). (2004)
23. Richard J. Lipton: Reduction: A Method of Proving Properties of Parallel Programs. Communications of the ACM **18** (1975) 717–721
24. Qadeer, S., Rajamani, S., Rehof, J.: Summarizing procedures in concurrent programs. In: Proceedings of the 31st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 04). (2004)

A Empirical Evaluation

A.1 Implementation

The proposed POR techniques was realized by implementing the sufficient set selection algorithm as a scheduling strategist in Bogor[12] model checker. Additional strategists to combine the proposed techniques with other POR techniques was also implemented. The static dependence information calculation has been layered on top of the equivalence class implementation available in the Indus[13] project. These implementations have been composed in Bandera[14] to realize a pipeline that accepts Java class files as inputs, performs slicing to preserve deadlocks, generates Bogor compatible model of the slice, and checks model for property violation.

A.2 Experimental Setup

Various examples from the Bandera project were used as test inputs to evaluate the proposed techniques. Brief descriptions of these examples are given below.

Alarm Clock (AC) A clock continually updates time and notifies clients registered for alarms. AC1, AC2, and AC3 are three variations of this example.

Bounded Buffer (BB) Two clients exchange data via add and remove operations through a bounded buffer. BB1, BB4, and BB8 are three variations of this example.

[12] http://bogor.projects.cis.ksu.edu
[13] http://indus.projects.cis.ksu.edu
[14] http://bandera.projects.cis.ksu.edu

Deadlock (DL) The classic deadlock example involving two locks being obtained in different order. DL1, DL2, and DL3 are three variations of this example.

Dining Philosopers (DP) The classic problem of dining philosophers. DP1, DP2, DP3, DP4, DP5, and DP6 are six variations of this example.

Disk Scheduler (DS) A disk scheduler is shared by a few disk Readers to concurrent read various cylinders on a disk. DS1, DS2, DS4, and DS7 are four variations of this example.

Molecular Dynamic (MD) A parallelized simulation of molecular dynamics (available as part of Java Grande benchmark). MD3 is a variation of this example.

Ray Tracing (RT) A parallelized 3D ray tracing program. RT3 is a variation of this example.

Pipeline (PL) A simulation of a work pipeline in which each stage is handled by a different thread. PL1 is a variation of this example.

Producer-Consumer (PC) Producers communicate data to consumers via a common buffer. Unlike in the bounded buffer program, the data flow is unidirectional. PC3 and PC4 are two variations of this example.

Readers-Writers (RW) The classic problem of concurrent readers and writers. RW1, RW2, RW3, RW4, and RW5 are five variations of this example.

Replicated Workers (RP) A realization of the common replicated workers pattern. RP12, RP13, RP14, RP15, and RP18 are five variations of this example.

Sleeping Barbers (SB) The classic problem of sleeping barbers. SB1, SB2, and SB4 are three variations of this example.

In terms of complexity, the code base of the examples is small (in the order of few KB of application class bytecodes), but they are inherently complex as each example involves at least three concurrent threads. More than one variant is considered for all but one example.

For each variant, nine experiments were conducted. These experiments were identical in terms of the input program and the generated slice and model. They differed in the kind of POR technique employed during model checking. The nine different POR configurations are given below.

E The POR technique (EPOR) that relies on dynamic detection of independent transitions based on the non-reachability of shared objects [1] was applied.

SC Conditional stubborn set (SPD-CSS) based on static program dependence information was applied.

SC+E SPD-CSS and EPOR techniques were applied in order.

SD Stateful dynamic POR technique (SDPOR) based on PDD was applied along with DEC optimizations.

E+SD EPOR and SDPOR (as in SD configuration) were applied in order.

SC+F SPD-CSS was applied with full enabled set coverage.

SC+E+F SPD-CSS and EPOR were applied in order with FESC.

SD+F SDPOR (as in SD configuration) was applied with FESC.

E+SD+F EPOR and SDPOR (as in E+SD configuration) were applied with FESC.

In all of the nine configurations, heap and process symmetry reductions [17] and collapse compression optimizations were applied along with the technique to aggregate transitions involving local variables.

All experiments were executed on the JVM available as part of Sun JDK 1.5.0_08 with 14GB of maximum heap space on multi-processor Linux boxes. As the pipeline and its components were single threaded, only one processor was used for computation. All experiments were allowed to run for up to 10 hours (36,000 seconds). Experiments that did not finish within the allotted time were terminated.

A.3 System Level Experimental Results

For each input program and each configuration, the raw exploration data was collected in terms (aspects) of the number of explored states *(States)*, matched states *(Matched)*, explored transitions *(Transitions)* and uncovered errors *(Errors)*. The total time and maximum memory required to execute the entire analysis and model checking pipeline *(Time)* was also recored.

The data for each aspect of each input program and configuration combination is given in the tables Table 2 thru Table 10. The data for each aspect is split into two tables: a) the data corresponding to completing executions and b) the data corresponding to terminated executions (due to time limit).

Time The data in Table 2 corresponds to the total time taken to execute the entire pipeline on each input program in different configurations. There were a few input programs that were terminated after 10 hours of execution, and these are not presented in the table.

The data indicates that the proposed approaches require time comparable to that required by EPOR-based approach. In case of long running model checks, the combination of EPOR and SDPOR based on pure dynamic approach (SD+E) affects 96-69% and 99-80% reduction in state space exploration time with and without FESC, respectively. SDPOR approach without EPOR provides comparable reductions as well. In many examples, the time taken by SPD-CSS (SC) approach is comparable to that of EPOR (E) approach.

In terms of time, in non-FESC setting, the proposed approaches are comparable or better than the most optimal EPOR approach (available in Bogor).

Space The data in Table 3 corresponds to the maximum memory required to execute the entire pipeline on each input program in different configurations. The data was collected by sampling the JVM at 5 second interval for the maximum allocated memory. Allocated memory is the memory procured by the JVM from the OS, and not all of the allocated memory is used by the heap. Hence, the data may be inaccurate.

In case of completed configurations with non-trivial state space, while there are cases where the proposed approaches fair better than E configuration and vice versa, the memory consumption of the proposed approaches are comparable to that of the EPOR approach. In case of FESC configurations, the configurations follow the complexity analysis and consume more memory as they cover larger state space.

In almost all terminated FESC configurations, the maximum allowed heap space is consumed. Hence, it is safe to conclude that FESC configurations on programs with non-trivial state space will be heavily memory intensive. In other terminated non-FESC configurations, SDPOR configurations are more memory intensive than the other configurations; this follows the complexity analysis of SDPOR. Interestingly, every terminated non-FESC SC configurations require less memory in comparison with the non-FESC E configuration. Hence, SC configurations may be better suited for memory-scarce environments.

States and Transitions The data in Table 4 corresponds to the number of visited states while model checking the input programs in various configurations. The data relative to E configuration is given in Table 5. Similarly, the number of matched states (revisited fully explored states) and number of executed transitions while model checking are given in Table 6 (Table 7) and Table 8 (Table 9), respectively.

In case of completed configurations, from the data in Table 5a, more states are explored in SC and SC+F configurations due to the inaccuracies in the static program dependence. This is also true for SD and SD+F

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
AC1	10128	1574	7351	588	36013	36052	36065	36049	11151
AC2	684	474	442	147	476	1182	830	691	297
AC3	844	566	425	138	424	966	622	443	152
BB1	69	56	45	46	42	49	49	52	49
BB4	66	60	51	50	67	158	157	90	76
BB8	60	58	50	48	52	44	48	44	50
DL1	52	52	53	48	50	39	53	39	43
DL2	75	48	44	53	40	47	37	37	37
DL3	64	47	46	54	41	40	39	39	50
DP1	54	57	39	50	47	40	50	42	41
DP2	57	57	49	49	37	45	43	49	51
DP3	64	52	50	38	33	54	52	56	51
DP4	65	58	45	47	53	46	41	50	52
DP5	69	51	47	50	49	65	59	56	54
DP6	64	40	41	44	34	64	65	55	53
MD3	36019	36247	5469	1564	36012	36081	36288	5497	1629
RT3	36011	13314	36014	1830	24263	36034	13287	36027	1827
PL1	1446	128	714	55	119	1746	126	748	53
PC3	75	53	47	49	36	45	42	40	43
PC4	61	60	52	49	46	62	41	43	45
RW1	53	48	41	49	36	65	53	50	49
RW2	123	131	94	95	60	113	151	82	95
RW3	69	50	45	43	40	46	46	53	40
RW4	70	57	51	51	49	63	53	51	49
RW5	78	73	60	57	62	78	69	63	59
RP13	337	162	325	338	619	317	156	338	332
RP15	14357	5956	16302	2619	13515	19593	6816	18419	2654
RP18	629	377	621	478	506	628	364	640	474
SB1	63	48	41	34	64	42	34	39	33
SB4	151	201	135	130	2415	142	210	139	135

Table 2: Total time (in seconds) taken to execute various configuration on various input programs. All unmentioned input programs ran for the entire 10 hours.

configurations in the context of programs with trivial state space. The configurations that combine EPOR and the proposed approaches almost always perform better than the E configuration.

In case of terminated configurations, the above observations hold in the allotted execution time (Table 5b). Hence, it is most likely, by projection, the above observations will hold if these configurations are executed to completion.

In case of input programs with zero errors, as expected, the state count in non-FESC configuration are very close or identical to that of their FESC counterpart.

All of the above observations also hold for matched states and transition data as well.

Errors The data in Table 10 corresponds to the number of errors uncovered in the input programs in various configurations. There were a few input programs with zero errors, and these are not presented in the table.

From the data corresponding to the completely executed configurations (in Table 10a), when FESC is not used, the number of errors detected by the proposed approaches is always less than or equal to those detected by EPOR approach. When FESC is used, the number of errors detected by the proposed approaches

is lower than those detected by EPOR approach for AC1 but higher for BB4. This clearly indicates that there are programs for which either approaches can perform relatively better.

In the case of terminated configurations, the data indicates that SPD-CSS (SC) approach quickly uncovers errors in the input programs; SPD-CSS uncovers errors quicker than the more optimal SDPOR (SD) approach. This behavior may be due to the non-determinism in the construction of the sufficient sets (the data for DS4 supports this observation). In cases (DS7) where all configurations detect errors, more errors are detected in the SC configuration in comparison with the E configuration. However, due to incompletion and based on the previous results, it is possible that the E configuration will detect more errors than the other configuration upon completion.

Based on the preliminary data and theoretical reasoning, I conjecture that the proposed approaches will uncover errors quickly with lower number of false positives in comparison with the optimal EPOR-based approach. However, given the small sample space and incomplete experiments (in terms of execution), more experimentation is required to empirical prove the above conjecture.

A.4 State Level Experimental Results

While the above evaluation is useful to assess the relative merit of the approaches at a system level, it is insufficient to evaluate the approaches at a micro (state) level. Further, it is insufficient to evaluate the approaches in case of terminated experiments. Hence, to facilitate a local and complimentary evaluation, local (reduction) data at the state level was collected. This data contains the total number of enabled *(E)*, sufficient *(A)*, and independent *(I)* transitions encountered at the visited states of the system. Further, the weighted average of the ratio of sufficient and enabled transition *(A/E)* at each visited states of the system was also collected. In case of configurations involving SDPOR, the total number of dynamic dependences discovered merely via static dependence information *(S)* and via pure dynamic approach *(D)* was collected and the ratio of between these numbers *(D/S)* was calculated. In the data for each input program, this data is presented in the (c) table.

Observations The relative data indicates that the proposed non-FESC SPD-CSS based approaches are comparable to EPOR based approach in case of programs with trivial state space and better in case of programs with non-trivial state space.

From the reduction data, the weighted average of the ratio of the size of the sufficient set to the size of enabled set at a state is similar in case of the SC and SD configurations. This is true independent of the completion and FESC nature of the configuration. Hence, the effect of the approaches distributes evenly across the reduced state space.

Similarly, the ratio of the number of dynamic dependences triggered via PDD and via static program dependences[15] is less than or equal to 0.01. This implies that, if possible, dynamic POR based on PDD should be preferred.

Although experiments dedicated to SDD approach were not conducted, based on the data from (S) and (D) and the relation between SDD and PDD, it follows that the data for SDD approach will be the same or better than that for static program dependence. Upon factoring in state local thread aliveness information, the SDD approach will most likely provide reduction that is better than SPD-CSS approach but still not better than PDD approach.

[15] These dependences are not pruned based on state local thread aliveness information.

B Data From POR Experiments

The data collected in the partial order reduction related experiments are presented. Please refer to Section A for the conclusions drawn from the experiments.

B.1 Data Description

For each input program and each configuration, the raw exploration data was collected in terms (aspects) of the number of explored states *(States)*, matched states *(Matched)*, explored transitions *(Transitions)* and uncovered errors *(Errors)*. The total time *(Time)* and maximum memory *(Memory)* required to execute the entire analysis and model checking pipeline was also recored. In the data for each input program, this data is presented in the (a) table.

To determine the relative merit of the proposed approach, the data from the non-E configurations were compared with the data from the E configuration by subtracting the former from the latter, e.g. *States* in SC was subtracted from *States* in E. In the data for each input program, this data is presented in the (b) table.

While the above evaluation is useful to assess the relative merit of the approaches at a system level, it is insufficient to evaluate the approaches at a micro (state) level. Further, it is insufficient to evaluate the approaches in case of terminated experiments. Hence, to facilitate a local and complimentary evaluation, local (reduction) data at the state level was collected. This data contains the total number of enabled *(E)*, ample *(A)*, and independent *(I)* transitions encountered at the visited states of the system. Further, the weighted average of the ratio of ample and enabled transition *(A/E)* at each visited states of the system was also collected. In case of configurations involving SDPOR, the total number of dynamic dependences discovered merely via static dependence information *(S)* and via pure dynamic approach *(D)* was collected and the ratio of between these numbers *(D/S)* was calculated. In the data for each input program, this data is presented in the (c) table.

The data is separated based on the completion of execution of at least one of the configuration.

B.2 Completed Configurations

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
AC1	2163	2228	3103	3124	4209	6178	6280	6905	7414
AC2	2079	1016	1009	516	769	6311	7184	7243	7113
AC3	1013	1419	779	578	495	1774	1865	1859	1698
BB1	594	934	1197	1144	94	1252	98	1252	1148
BB4	60	73	93	93	92	100	98	100	100
BB8	59	72	91	91	91	92	92	92	92
DL1	60	72	92	91	91	92	92	92	92
DL2	59	71	91	91	90	91	91	91	91
DL3	59	71	91	91	90	91	91	91	91
DP1	533	71	91	91	91	117	117	117	117
DP2	615	72	91	91	91	128	128	128	128
DP3	533	72	91	91	91	128	128	128	128
DP4	903	92	92	92	92	867	194	194	194
DP5	596	92	93	93	92	820	152	153	153
DP6	515	92	93	93	92	692	151	151	151
MD3	5937	14842	6135	5932	5890	10539	14407	6207	5966
RT3	5526	6381	5960	5607	5783	6942	6640	6666	5652
PL1	1358	1151	829	232	799	1641	1269	1481	248
PC3	430	73	93	93	92	95	95	95	95
PC4	1046	1143	1352	1351	95	1356	1356	1460	1356
RW1	654	1022	106	106	106	1283	1152	107	107
RW2	1422	981	563	717	505	1182	826	475	628
RW3	144	92	93	93	93	94	94	94	94
RW4	987	1039	94	94	93	1353	1143	95	95
RW5	1019	625	60	61	59	693	630	64	65
RP13	2227	2074	1322	1286	872	1409	1347	1631	1956
RP15	5119	13922	6405	7468	6382	12220	14390	10279	7954
RP18	2736	1480	1486	2151	2306	2971	3126	3612	4041
SB1	392	74	96	95	3314	632	715	129	127
SB4	1150	604	851	1015	628	830	760	849	824

(a)

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
DS1	3075	12172	6221	14043	6579	8462	14268	13351	14300
DS2	2033	12356	6665	14247	5549	10150	14252	13792	14293
DS4	4447	10549	7910	13811	7261	12701	14272	13918	14315
DS7	2618	8645	4969	7226	3599	7247	7967	7358	7282
RP12	7325	9671	7541	14613	8189	14940	11424	14495	14735
RP14	6378	9378	6990	9033	7357	12882	12178	11224	11003
SB2	7284	11667	8253	10583	8723	12376	14894	12529	12682

(b)

Table 3: Maximum memory (in MB) consumed to execute in various configuration on various input programs. (a) is the memory consumption for input programs on which the end-to-end execution completed within 10 hours for at least one configuration. (b) is the memory consumption for input programs on which the end-to-end execution was terminated for all configurations.

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
AC1	161172	47522	78869	9240	256479	35922	35478	18928	8248
AC2	18051	12766	5568	1857	11889	18051	12766	5568	1857
AC3	27470	16053	6815	1946	11030	27470	16053	6815	1946
BB1	1586	929	794	644	90	1586	929	794	644
BB4	46	44	7	5	122	257	255	109	50
BB8	152	149	21	23	28	152	149	21	23
DL1	173	113	115	89	178	173	135	115	111
DL2	47	34	8	8	8	47	34	8	8
DL3	61	43	7	4	7	61	46	7	7
DP1	746	304	30	16	30	746	304	30	16
DP2	798	311	51	18	51	798	325	51	28
DP3	798	311	51	18	51	798	325	51	28
DP4	1618	373	30	16	30	1621	376	30	16
DP5	1910	442	8	9	153	1930	448	8	9
DP6	1910	442	8	9	153	1930	448	8	9
MD3	11908	9150	4419	2478	200622	10383	8463	4419	2478
RT3	1831	22909	2298	171	4616	1812	22909	2186	171
PL1	301293	12710	69682	82	7379	301293	12710	69682	82
PC3	291	287	48	48	57	291	287	48	48
PC4	1007	582	88	88	103	1007	582	88	88
RW1	1403	861	102	72	124	1415	875	104	72
RW2	13338	9100	5186	2880	1214	13342	10335	4947	4215
RW3	1403	861	102	72	124	1415	875	104	72
RW4	1661	979	121	88	134	1671	984	121	89
RW5	5113	2616	173	177	58	5161	2625	180	183
RP13	5760	1619	1788	1587	6577	5760	1619	1788	1587
RP15	1043247	251698	568928	95657	612954	1043247	251698	568928	95657
RP18	48232	20550	23408	14497	26854	48232	20550	23408	14497
SB1	1870	743	442	54	11205	1870	803	442	114
SB4	31343	22866	19349	11167	213865	31346	23298	19353	11599

(a)

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
DS1	484320	280717	245323	154538	318652	107674	186871	170504	110420
DS2	586207	312253	302288	175784	439430	126848	192269	193605	119919
DS4	927413	199748	256656	122645	282599	245509	164164	151804	96442
DS7	802545	247030	136911	77217	157684	111268	74080	75774	43290
RP12	2854899	320514	1757174	636581	1010626	1849804	316597	1526923	571418
RP14	1496403	523174	916175	387088	480884	1222803	503619	850187	377415
SB2	3567474	1025894	1458629	599683	722112	2035248	1001193	1167223	568748

(b)

Table 4: The count of encountered states in various configuration on various input programs. (a) is the state count for input programs on which the end-to-end execution completed within 10 hours for at least one configuration. (b) is the state count for input programs on which the end-to-end execution was terminated for all configurations.

Conf	SC	SD	SC+E	E+SD	SC+F	SD+F	SC+F+E	E+SD+F
AC1	-95307	-208957	-177610	-247239	-220557	-221001	-237551	-248231
AC2	6162	877	-6321	-10032	6162	877	-6321	-10032
AC3	16440	5023	-4215	-9084	16440	5023	-4215	-9084
BB1	1496	839	704	554	1496	839	704	554
BB4	-76	-78	-115	-117	135	133	-13	-72
BB8	124	121	-7	-5	124	121	-7	-5
DL1	-5	-65	-63	-89	-5	-43	-63	-67
DL2	39	26	0	0	39	26	0	0
DL3	54	36	0	-3	54	39	0	0
DP1	716	274	0	-14	716	274	0	-14
DP2	747	260	0	-33	747	274	0	-23
DP3	747	260	0	-33	747	274	0	-23
DP4	1588	343	0	-14	1591	346	0	-14
DP5	1757	289	-145	-144	1777	295	-145	-144
DP6	1757	289	-145	-144	1777	295	-145	-144
MD3	-188714	-191472	-196203	-198144	-190239	-192159	-196203	-198144
RT3	-2785	18293	-2318	-4445	-2804	18293	-2430	-4445
PL1	293914	5331	62303	-7297	293914	5331	62303	-7297
PC3	234	230	-9	-9	234	230	-9	-9
PC4	904	479	-15	-15	904	479	-15	-15
RW1	1279	737	-22	-52	1291	751	-20	-52
RW2	12124	7886	3972	1666	12128	9121	3733	3001
RW3	1279	737	-22	-52	1291	751	-20	-52
RW4	1527	845	-13	-46	1537	850	-13	-45
RW5	5055	2558	115	119	5103	2567	122	125
RP13	-817	-4958	-4789	-4990	-817	-4958	-4789	-4990
RP15	430293	-361256	-44026	-517297	430293	-361256	-44026	-517297
RP18	21378	-6304	-3446	-12357	21378	-6304	-3446	-12357
SB1	-9335	-10462	-10763	-11151	-9335	-10402	-10763	-11091
SB4	-182522	-190999	-194516	-202698	-182519	-190567	-194512	-202266

(a)

Conf	SC	SD	SC+E	E+SD	SC+F	SD+F	SC+F+E	E+SD+F
DS1	165668	-37935	-73329	-164114	-210978	-131781	-148148	-208232
DS2	146777	-127177	-137142	-263646	-312582	-247161	-245825	-319511
DS4	644814	-82851	-25943	-159954	-37090	-118435	-130795	-186157
DS7	644861	89346	-20773	-80467	-46416	-83604	-81910	-114394
RP12	1844273	-690112	746548	-374045	839178	-694029	516297	-439208
RP14	1015519	42290	435291	-93796	741919	22735	369303	-103469
SB2	2845362	303782	736517	-122429	1313136	279081	445111	-153364

(b)

Table 5: The E configuration relative count of encountered states in various configuration on various input programs. (a) is the state count for input programs on which the end-to-end execution completed within 10 hours for at least one configuration. (b) is the state count for input programs on which the end-to-end execution was terminated for all configurations.

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
AC1	114976	14334	52419	1441	415793	26061	12167	13901	2324
AC2	11475	5099	3889	485	13067	11475	5099	3889	485
AC3	21477	7469	4863	342	11239	21477	7469	4863	342
BB1	835	309	413	233	90	838	343	420	265
BB4	27	25	1	0	88	242	239	104	45
BB8	56	52	13	10	28	56	52	14	15
DL1	83	43	43	27	114	83	56	43	40
DL2	20	11	1	1	1	20	11	1	1
DL3	24	14	0	0	0	24	14	0	0
DP1	792	272	36	4	57	800	274	41	6
DP2	831	276	63	5	99	843	284	70	14
DP3	831	276	63	5	99	843	284	70	14
DP4	2056	334	36	4	57	2092	336	41	6
DP5	2304	417	2	1	346	2359	423	3	3
DP6	2304	417	2	1	346	2359	423	3	3
MD3	743	0	647	43	197416	471	0	647	43
RT3	156	5276	1119	5	3569	155	5276	1033	5
PL1	362901	13425	78117	17	15859	362901	13425	78117	17
PC3	87	84	31	31	44	87	84	31	31
PC4	615	284	64	64	85	615	284	64	64
RW1	1453	733	127	66	296	1493	770	139	79
RW2	10795	6438	3480	1606	2559	11328	7562	3758	2876
RW3	1453	733	127	66	296	1493	770	139	79
RW4	1647	835	154	79	299	1702	854	157	93
RW5	4744	1937	114	75	108	4831	1995	129	84
RP13	1764	747	641	495	6562	1764	747	641	495
RP15	753132	108054	328218	43268	1010878	753132	108054	328218	43268
RP18	34787	10783	13496	7225	40226	34787	10783	13496	7225
SB1	1650	663	147	8	22139	1650	683	147	24
SB4	24638	16879	12790	6408	493092	24770	17254	12922	6783

(a)

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
DS1	176323	78041	97748	60075	137135	37712	51978	67193	42890
DS2	213654	86737	121775	68830	188600	44942	53477	76500	46644
DS4	219615	44681	83012	38192	96813	57062	36282	48727	30033
DS7	528986	124269	78195	30770	129614	61175	28997	40260	17457
RP12	1773213	157402	1028687	309869	2168202	1143564	155417	893958	281616
RP14	701784	207986	421679	153376	961519	566817	200552	389611	149814
SB2	3870150	1223108	2250433	922882	3246748	2163348	981585	1825146	881842

(b)

Table 6: The count of encountered matched states in various configuration on various input programs. (a) is the matched state count for input programs on which the end-to-end execution completed within 10 hours for at least one configuration. (b) is the matched state count for input programs on which the end-to-end execution was terminated for all configurations.

Conf	SC	SD	SC+E	E+SD	SC+F	SD+F	SC+F+E	E+SD+F
AC1	-300817	-401459	-363374	-414352	-389732	-403626	-401892	-413469
AC2	-1592	-7968	-9178	-12582	-1592	-7968	-9178	-12582
AC3	10238	-3770	-6376	-10897	10238	-3770	-6376	-10897
BB1	745	219	323	143	748	253	330	175
BB4	-61	-63	-87	-88	154	151	16	-43
BB8	28	24	-15	-18	28	24	-14	-13
DL1	-31	-71	-71	-87	-31	-58	-71	-74
DL2	19	10	0	0	19	10	0	0
DL3	24	14	0	0	24	14	0	0
DP1	735	215	-21	-53	743	217	-16	-51
DP2	732	177	-36	-94	744	185	-29	-85
DP3	732	177	-36	-94	744	185	-29	-85
DP4	1999	277	-21	-53	2035	279	-16	-51
DP5	1958	71	-344	-345	2013	77	-343	-343
DP6	1958	71	-344	-345	2013	77	-343	-343
MD3	-196673	-197416	-196769	-197373	-196945	-197416	-196769	-197373
RT3	-3413	1707	-2450	-3564	-3414	1707	-2536	-3564
PL1	347042	-2434	62258	-15842	347042	-2434	62258	-15842
PC3	43	40	-13	-13	43	40	-13	-13
PC4	530	199	-21	-21	530	199	-21	-21
RW1	1157	437	-169	-230	1197	474	-157	-217
RW2	8236	3879	921	-953	8769	5003	1199	317
RW3	1157	437	-169	-230	1197	474	-157	-217
RW4	1348	536	-145	-220	1403	555	-142	-206
RW5	4636	1829	6	-33	4723	1887	21	-24
RP13	-4798	-5815	-5921	-6067	-4798	-5815	-5921	-6067
RP15	-257746	-902824	-682660	-967610	-257746	-902824	-682660	-967610
RP18	-5439	-29443	-26730	-33001	-5439	-29443	-26730	-33001
SB1	-20489	-21476	-21992	-22131	-20489	-21456	-21992	-22115
SB4	-468454	-476213	-480302	-486684	-468322	-475838	-480170	-486309

(a)

Conf	SC	SD	SC+E	E+SD	SC+F	SD+F	SC+F+E	E+SD+F
DS1	39188	-59094	-39387	-77060	-99423	-85157	-69942	-94245
DS2	25054	-101863	-66825	-119770	-143658	-135123	-112100	-141956
DS4	122802	-52132	-13801	-58621	-39751	-60531	-48086	-66780
DS7	399372	-5345	-51419	-98844	-68439	-100617	-89354	-112157
RP12	-394989	-2010800	-1139515	-1858333	-1024638	-2012785	-1274244	-1886586
RP14	-259735	-753533	-539840	-808143	-394702	-760967	-571908	-811705
SB2	623402	-2023640	-996315	-2323866	-1083400	-2265163	-1421602	-2364906

(b)

Table 7: The E configuration relative count of encountered matched states in various configuration on various input programs. (a) is the matched state count for input programs on which the end-to-end execution completed within 10 hours for at least one configuration. (b) is the matched state count for input programs on which the end-to-end execution was terminated for all configurations.

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
AC1	2617743	619812	1471300	149699	5599303	506774	425965	363324	125174
AC2	443403	273092	165798	38522	418901	443403	273092	165798	38522
AC3	828223	482608	238414	55827	499105	828223	482608	238414	55827
BB1	7772	4677	5576	4493	2448	7775	4711	5583	4525
BB4	301	282	101	83	869	727	707	306	230
BB8	810	796	379	408	507	810	796	380	413
DL1	665	388	491	324	809	665	512	491	448
DL2	182	132	77	77	77	182	132	77	77
DL3	198	133	86	65	86	198	154	86	86
DP1	2426	925	266	173	287	2434	927	271	175
DP2	2584	950	391	192	427	2596	990	398	243
DP3	2589	955	396	197	432	2601	995	403	248
DP4	6197	1268	321	228	342	6238	1275	326	230
DP5	7123	1564	142	144	1434	7200	1578	143	146
DP6	7115	1556	138	140	1426	7192	1570	139	142
MD3	7251107	2762960	529098	103302	8287084	6303446	2105125	529098	103302
RT3	1455033	435667	711010	42604	1247000	1437594	435667	702675	42604
PL1	1081729	46687	323185	553	76291	1081729	46687	323185	553
PC3	2189	2177	1363	1375	1637	2189	2177	1363	1375
PC4	9986	7488	3460	3469	4057	9986	7488	3460	3469
RW1	5480	3564	1404	989	1746	5578	3689	1432	1002
RW2	47032	30692	22955	12789	12312	47612	35327	22517	18766
RW3	5480	3564	1404	989	1746	5578	3689	1432	1002
RW4	7063	4426	1794	1311	2091	7190	4490	1797	1335
RW5	22702	12243	2975	2665	2116	23098	12346	3058	2787
RP13	98168	31180	63785	60595	362027	98168	31180	63785	60595
RP15	4823439	1047995	3259745	413817	5709125	4823439	1047995	3259745	413817
RP18	252853	105190	154699	96363	321962	252853	105190	154699	96363
SB1	5656	2090	1502	256	47317	5656	2300	1502	462
SB4	90649	63264	56343	31290	984758	90794	64574	56489	32600

(a)

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
DS1	13130000	8420000	10761570	6858336	14480000	2980661	5605795	7540000	4895085
DS2	15861823	9359041	13180000	7805424	19900000	3491532	5770000	8560000	5312019
DS4	29990000	6570000	12170000	5874295	13800000	8020000	5430000	7180000	4635284
DS7	14401190	4104736	4603123	2522397	5143947	2597494	1721830	2576842	1417963
RP12	9006579	967201	5700000	2055708	7522067	5880000	955702	4980000	1812125
RP14	5400914	2018866	3419295	1599607	6417914	4373007	1962390	3205777	1570605
SB2	12208052	3970000	6286847	2661671	10869856	6910000	3534082	5009925	2521013

(b)

Table 8: The count of executed transitions in various configuration on various input programs. (a) is the transition count for input programs on which the end-to-end execution completed within 10 hours for at least one configuration. (b) is the transition count for input programs on which the end-to-end execution was terminated for all configurations.

Conf	SC	SD	SC+E	E+SD	SC+F	SD+F	SC+F+E	E+SD+F
AC1	-2981560	-4979491	-4128003	-5449604	-5092529	-5173338	-5235979	-5474129
AC2	24502	-145809	-253103	-380379	24502	-145809	-253103	-380379
AC3	329118	-16497	-260691	-443278	329118	-16497	-260691	-443278
BB1	5324	2229	3128	2045	5327	2263	3135	2077
BB4	-568	-587	-768	-786	-142	-162	-563	-639
BB8	303	289	-128	-99	303	289	-127	-94
DL1	-144	-421	-318	-485	-144	-297	-318	-361
DL2	105	55	0	0	105	55	0	0
DL3	112	47	0	-21	112	68	0	0
DP1	2139	638	-21	-114	2147	640	-16	-112
DP2	2157	523	-36	-235	2169	563	-29	-184
DP3	2157	523	-36	-235	2169	563	-29	-184
DP4	5855	926	-21	-114	5896	933	-16	-112
DP5	5689	130	-1292	-1290	5766	144	-1291	-1288
DP6	5689	130	-1288	-1286	5766	144	-1287	-1284
MD3	-1035977	-5524124	-7757986	-8183782	-1983638	-6181959	-7757986	-8183782
RT3	208033	-811333	-535990	-1204396	190594	-811333	-544325	-1204396
PL1	1005438	-29604	246894	-75738	1005438	-29604	246894	-75738
PC3	552	540	-274	-262	552	540	-274	-262
PC4	5929	3431	-597	-588	5929	3431	-597	-588
RW1	3734	1818	-342	-757	3832	1943	-314	-744
RW2	34720	18380	10643	477	35300	23015	10205	6454
RW3	3734	1818	-342	-757	3832	1943	-314	-744
RW4	4972	2335	-297	-780	5099	2399	-294	-756
RW5	20586	10127	859	549	20982	10230	942	671
RP13	-263859	-330847	-298242	-301432	-263859	-330847	-298242	-301432
RP15	-885686	-4661130	-2449380	-5295308	-885686	-4661130	-2449380	-5295308
RP18	-69109	-216772	-167263	-225599	-69109	-216772	-167263	-225599
SB1	-41661	-45227	-45815	-47061	-41661	-45017	-45815	-46855
SB4	-894109	-921494	-928415	-953468	-893964	-920184	-928269	-952158

(a)

Conf	SC	SD	SC+E	E+SD	SC+F	SD+F	SC+F+E	E+SD+F
DS1	-1350000	-6060000	-3718430	-7621664	-11499339	-8874205	-6940000	-9584915
DS2	-4038177	-10540959	-6720000	-12094576	-16408468	-14130000	-11340000	-14587981
DS4	16190000	-7230000	-1630000	-7925705	-5780000	-8370000	-6620000	-9164716
DS7	9257243	-1039211	-540824	-2621550	-2546453	-3422117	-2567105	-3725984
RP12	1484512	-6554866	-1822067	-5466359	-1642067	-6566365	-2542067	-5709942
RP14	-1017000	-4399048	-2998619	-4818307	-2044907	-4455524	-3212137	-4847309
SB2	1338196	-6899856	-4583009	-8208185	-3959856	-7335774	-5859931	-8348843

(b)

Table 9: The E configuration relative count of executed transitions in various configuration on various input programs. (a) is the transition count for input programs on which the end-to-end execution completed within 10 hours for at least one configuration. (b) is the transition count for input programs on which the end-to-end execution was terminated for all configurations.

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
AC1	5531	250	1982	77	14668	3650	3476	3727	1144
BB4	5	4	2	1	32	216	215	104	46
DL1	2	2	2	2	2	2	2	2	2
DL2	1	1	1	1	1	1	1	1	1
DL3	2	1	2	1	2	2	2	2	2
DP1	1	1	1	1	1	1	1	1	1
DP2	1	1	1	1	1	1	1	1	1
DP3	1	1	1	1	1	1	1	1	1
DP4	1	1	1	1	1	1	1	1	1
DP5	1	1	1	1	1	1	1	1	1
DP6	1	1	1	1	1	1	1	1	1
SB1	6	2	6	2	6	6	6	6	6

(a)

Conf	SC	SD	SC+E	E+SD	E	SC+F	SD+F	SC+F+E	E+SD+F
DS1	6922	3	448	0	0	984	0	0	0
DS2	8672	5	865	1	0	1366	0	0	0
DS4	21	0	0	0	0	0	0	0	0
DS7	10604	2490	4346	2409	4155	2679	1977	2228	2027
SB2	1	1	1	1	1	1	1	1	1

(b)

Table 10: The count of errors detected in various configuration on various input programs. (a) is the error count for input programs on which the end-to-end execution completed within 10 hours for at least one configuration. (b) is the error count for input programs on which the end-to-end execution was terminated for all configurations. For unmentioned input programs, the error count was zero in all configurations.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	10128	2163	161172	114976	2617743	5531
SD	1574	2228	47522	14334	619812	250
SC+E	7351	3103	78869	52419	1471300	1982
E+SD	588	3124	9240	1441	149699	77
E	36013	4209	256479	415793	5599303	14668
SC+F	36052	6178	35922	26061	506774	3650
SD+F	36065	6280	35478	12167	425965	3476
SC+F+E	36049	6905	18928	13901	363324	3727
E+SD+F	11151	7414	8248	2324	125174	1144

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-25885	-2046	-95307	-300817	-2981560	-9137
SD	-34439	-1981	-208957	-401459	-4979491	-14418
SC+E	-28662	-1106	-177610	-363374	-4128003	-12686
E+SD	-35425	-1085	-247239	-414352	-5449604	-14591
SC+F	39	1969	-220557	-389732	-5092529	-11018
SD+F	52	2071	-221001	-403626	-5173338	-11192
SC+F+E	36	2696	-237551	-401892	-5235979	-10941
E+SD+F	-24862	3205	-248231	-413469	-5474129	-13524

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	6050668	2617641	2340181	3710487	277460	0.09	0	0	0
SD	1438791	614811	560289	878502	54522	0.09	854233	4981	0.01
SC+E	3086258	1476912	1321733	1764525	155179	0.12	0	0	0
E+SD	165678	148590	136662	29016	11928	0.90	84256	1111	0.01
SC+F	1123359	504036	445600	677759	58436	0.12	0	0	0
SD+F	946075	419355	382243	563832	37112	0.10	504248	3178	0.01
SC+F+E	752479	361384	328420	424059	32964	0.14	0	0	0
E+SD+F	135661	123174	112860	22801	10314	0.90	76378	885	0.01

(c)

Table 11: The raw and EPOR-relative exploration data and reduction data from alarm clock AC1 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	684	2079	18051	11475	443403	0
SD	474	1016	12766	5099	273092	0
SC+E	442	1009	5568	3889	165798	0
E+SD	147	516	1857	485	38522	0
E	476	769	11889	13067	418901	0
SC+F	1182	6311	18051	11475	443403	0
SD+F	830	7184	12766	5099	273092	0
SC+F+E	691	7243	5568	3889	165798	0
E+SD+F	297	7113	1857	485	38522	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	208	1310	6162	-1592	24502	0
SD	-2	247	877	-7968	-145809	0
SC+E	-34	240	-6321	-9178	-253103	0
E+SD	-329	-253	-10032	-12582	-380379	0
SC+F	706	5542	6162	-1592	24502	0
SD+F	354	6415	877	-7968	-145809	0
SC+F+E	215	6474	-6321	-9178	-253103	0
E+SD+F	-179	6344	-10032	-12582	-380379	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	967950	443359	396090	571860	47269	0.18	0	0	0
SD	624603	272112	250020	374583	22092	0.11	364679	987	0.00
SC+E	280233	167247	138646	141587	28601	0.40	0	0	0
E+SD	41470	38179	33896	7574	4283	0.92	37811	340	0.01
SC+F	967950	443359	396090	571860	47269	0.18	0	0	0
SD+F	624603	272112	250020	374583	22092	0.11	364679	987	0.00
SC+F+E	280233	167247	138646	141587	28601	0.40	0	0	0
E+SD+F	41470	38179	33896	7574	4283	0.92	37811	340	0.01

(c)

Table 12: The raw and EPOR-relative exploration data and reduction data from alarm clock AC2 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	844	1013	27470	21477	828223	0
SD	566	1419	16053	7469	482608	0
SC+E	425	779	6815	4863	238414	0
E+SD	138	578	1946	342	55827	0
E	424	495	11030	11239	499105	0
SC+F	966	1774	27470	21477	828223	0
SD+F	622	1865	16053	7469	482608	0
SC+F+E	443	1859	6815	4863	238414	0
E+SD+F	152	1698	1946	342	55827	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	420	518	16440	10238	329118	0
SD	142	924	5023	-3770	-16497	0
SC+E	1	284	-4215	-6376	-260691	0
E+SD	-286	83	-9084	-10897	-443278	0
SC+F	542	1279	16440	10238	329118	0
SD+F	198	1370	5023	-3770	-16497	0
SC+F+E	19	1364	-4215	-6376	-260691	0
E+SD+F	-272	1203	-9084	-10897	-443278	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	1868586	828060	757631	1110955	70429	0.14	0	0	0
SD	1127213	479693	453561	673652	26132	0.08	565788	2878	0.01
SC+E	419007	240040	206271	212736	33769	0.33	0	0	0
E+SD	60114	55624	51214	8900	4410	0.92	22137	200	0.01
SC+F	1868586	828060	757631	1110955	70429	0.14	0	0	0
SD+F	1127213	479693	453561	673652	26132	0.08	565788	2878	0.01
SC+F+E	419007	240040	206271	212736	33769	0.33	0	0	0
E+SD+F	60114	55624	51214	8900	4410	0.92	22137	200	0.01

(c)

Table 13: The raw and EPOR-relative exploration data and reduction data from alarm clock AC3 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	69	594	1586	835	7772	0
SD	56	934	929	309	4677	0
SC+E	45	1197	794	413	5576	0
E+SD	46	1144	644	233	4493	0
E	42	94	90	90	2448	0
SC+F	49	1252	1586	838	7775	0
SD+F	49	98	929	343	4711	0
SC+F+E	52	1252	794	420	5583	0
E+SD+F	49	1148	644	265	4525	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	27	500	1496	745	5324	0
SD	14	840	839	219	2229	0
SC+E	3	1103	704	323	3128	0
E+SD	4	1050	554	143	2045	0
SC+F	7	1158	1496	748	5327	0
SD+F	7	4	839	253	2263	0
SC+F+E	10	1158	704	330	3135	0
E+SD+F	7	1054	554	175	2077	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	11059	7771	4196	6863	3575	0.54	0	0	0
SD	6723	4476	2610	4113	1866	0.49	95171	200	0.00
SC+E	7720	5642	3133	4587	2509	0.60	0	0	0
E+SD	4974	4324	2558	2416	1766	0.85	95002	168	0.00
SC+F	11059	7771	4196	6863	3575	0.54	0	0	0
SD+F	6723	4476	2610	4113	1866	0.49	97550	200	0.00
SC+F+E	7720	5642	3133	4587	2509	0.60	0	0	0
E+SD+F	4974	4324	2558	2416	1766	0.85	95544	168	0.00

(c)

Table 14: The raw and EPOR-relative exploration data and reduction data from bounded buffer BB1 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	66	60	46	27	301	5
SD	60	73	44	25	282	4
SC+E	51	93	7	1	101	2
E+SD	50	93	5	0	83	1
E	67	92	122	88	869	32
SC+F	158	100	257	242	727	216
SD+F	157	98	255	239	707	215
SC+F+E	90	100	109	104	306	104
E+SD+F	76	100	50	45	230	46

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-1	-32	-76	-61	-568	-27
SD	-7	-19	-78	-63	-587	-28
SC+E	-16	1	-115	-87	-768	-30
E+SD	-17	1	-117	-88	-786	-31
SC+F	91	8	135	154	-142	184
SD+F	90	6	133	151	-162	183
SC+F+E	23	8	-13	16	-563	72
E+SD+F	9	8	-72	-43	-639	14

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	489	295	197	292	98	0.43	0	0	0
SD	457	277	182	275	95	0.45	0	0	0
SC+E	165	109	77	88	32	0.48	0	0	0
E+SD	86	81	62	24	19	0.94	0	0	0
SC+F	1014	506	408	606	98	0.24	0	0	0
SD+F	982	488	393	589	95	0.25	286	0	0
SC+F+E	368	211	179	189	32	0.24	0	0	0
E+SD+F	188	183	164	24	19	0.97	22	0	0

(c)

Table 15: The raw and EPOR-relative exploration data and reduction data from bounded buffer BB4 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	60	59	152	56	810	0
SD	58	72	149	52	796	0
SC+E	50	91	21	13	379	0
E+SD	48	91	23	10	408	0
E	52	91	28	28	507	0
SC+F	44	92	152	56	810	0
SD+F	48	92	149	52	796	0
SC+F+E	44	92	21	14	380	0
E+SD+F	50	92	23	15	413	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	8	-32	124	28	303	0
SD	6	-19	121	24	289	0
SC+E	-2	0	-7	-15	-128	0
E+SD	-4	0	-5	-18	-99	0
SC+F	-8	1	124	28	303	0
SD+F	-4	1	121	24	289	0
SC+F+E	-8	1	-7	-14	-127	0
E+SD+F	-2	1	-5	-13	-94	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	1054	809	397	657	412	0.69	0	0	0
SD	1032	789	392	640	397	0.68	2546	6	0.00
SC+E	462	386	199	263	187	0.80	0	0	0
E+SD	423	402	211	212	191	0.95	1088	5	0.00
SC+F	1054	809	397	657	412	0.69	0	0	0
SD+F	1032	789	392	640	397	0.68	2546	6	0.00
SC+F+E	462	386	199	263	187	0.80	0	0	0
E+SD+F	423	402	211	212	191	0.95	1124	5	0.00

(c)

Table 16: The raw and EPOR-relative exploration data and reduction data from bounded buffer BB8 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	52	60	173	83	665	2
SD	52	72	113	43	388	2
SC+E	53	92	115	43	491	2
E+SD	48	91	89	27	324	2
E	50	91	178	114	809	2
SC+F	39	92	173	83	665	2
SD+F	53	92	135	56	512	2
SC+F+E	39	92	115	43	491	2
E+SD+F	43	92	111	40	448	2

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	2	-31	-5	-31	-144	0
SD	2	-19	-65	-71	-421	0
SC+E	3	1	-63	-71	-318	0
E+SD	-2	0	-89	-87	-485	0
SC+F	-11	1	-5	-31	-144	0
SD+F	3	1	-43	-58	-297	0
SC+F+E	-11	1	-63	-71	-318	0
E+SD+F	-7	1	-67	-74	-361	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	930	664	348	582	316	0.59	0	0	0
SD	532	352	191	341	161	0.49	368	35	0.10
SC+E	662	494	265	397	229	0.63	0	0	0
E+SD	372	292	168	204	124	0.73	244	31	0.13
SC+F	930	664	348	582	316	0.59	0	0	0
SD+F	693	463	262	431	201	0.50	511	44	0.09
SC+F+E	662	494	265	397	229	0.63	0	0	0
E+SD+F	503	403	239	264	164	0.75	387	40	0.10

(c)

Table 17: The raw and EPOR-relative exploration data and reduction data from deadlock DL1 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	75	59	47	20	182	1
SD	48	71	34	11	132	1
SC+E	44	91	8	1	77	1
E+SD	53	91	8	1	77	1
E	40	90	8	1	77	1
SC+F	47	91	47	20	182	1
SD+F	37	91	34	11	132	1
SC+F+E	37	91	8	1	77	1
E+SD+F	37	91	8	1	77	1

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	35	-31	39	19	105	0
SD	8	-19	26	10	55	0
SC+E	4	1	0	0	0	0
E+SD	13	1	0	0	0	0
SC+F	7	1	39	19	105	0
SD+F	-3	1	26	10	55	0
SC+F+E	-3	1	0	0	0	0
E+SD+F	-3	1	0	0	0	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	268	181	107	161	74	0.55	0	0	0
SD	186	128	76	110	52	0.59	99	3	0.03
SC+E	102	80	50	52	30	0.74	0	0	0
E+SD	80	74	50	30	24	0.93	19	2	0.11
SC+F	268	181	107	161	74	0.55	0	0	0
SD+F	186	128	76	110	52	0.59	99	3	0.03
SC+F+E	102	80	50	52	30	0.74	0	0	0
E+SD+F	80	74	50	30	24	0.93	19	2	0.11

(c)

Table 18: The raw and EPOR-relative exploration data and reduction data from deadlock DL2 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	64	59	61	24	198	2
SD	47	71	43	14	133	1
SC+E	46	91	7	0	86	2
E+SD	54	91	4	0	65	1
E	41	90	7	0	86	2
SC+F	40	91	61	24	198	2
SD+F	39	91	46	14	154	2
SC+F+E	39	91	7	0	86	2
E+SD+F	50	91	7	0	86	2

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	23	-31	54	24	112	0
SD	6	-19	36	14	47	-1
SC+E	5	1	0	0	0	0
E+SD	13	1	-3	0	-21	-1
SC+F	-1	1	54	24	112	0
SD+F	-2	1	39	14	68	0
SC+F+E	-2	1	0	0	0	0
E+SD+F	9	1	0	0	0	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	282	197	89	193	108	0.58	0	0	0
SD	198	132	62	136	70	0.52	64	0	0
SC+E	115	91	50	65	41	0.75	0	0	0
E+SD	69	64	40	29	24	0.94	5	0	0
SC+F	282	197	89	193	108	0.58	0	0	0
SD+F	219	152	72	147	80	0.58	87	0	0
SC+F+E	115	91	50	65	41	0.75	0	0	0
E+SD+F	89	84	50	39	34	0.95	14	0	0

(c)

Table 19: The raw and EPOR-relative exploration data and reduction data from deadlock DL3 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	54	533	746	792	2426	1
SD	57	71	304	272	925	1
SC+E	39	91	30	36	266	1
E+SD	50	91	16	4	173	1
E	47	91	30	57	287	1
SC+F	40	117	746	800	2434	1
SD+F	50	117	304	274	927	1
SC+F+E	42	117	30	41	271	1
E+SD+F	41	117	16	6	175	1

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	7	442	716	735	2139	0
SD	10	-20	274	215	638	0
SC+E	-8	0	0	-21	-21	0
E+SD	3	0	-14	-53	-114	0
SC+F	-7	26	716	743	2147	0
SD+F	3	26	274	217	640	0
SC+F+E	-5	26	0	-16	-16	0
E+SD+F	-6	26	-14	-51	-112	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	5800	2425	922	4878	1503	0.12	0	0	0
SD	2203	901	348	1855	553	0.15	12511	23	0.00
SC+E	581	289	144	437	145	0.46	0	0	0
E+SD	214	168	110	104	58	0.87	199	4	0.02
SC+F	5800	2425	922	4878	1503	0.12	0	0	0
SD+F	2203	901	348	1855	553	0.15	12620	23	0.00
SC+F+E	581	289	144	437	145	0.46	0	0	0
E+SD+F	214	168	110	104	58	0.87	208	4	0.02

(c)

Table 20: The raw and EPOR-relative exploration data and reduction data from dining philosophers DP1 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	57	615	798	831	2584	1
SD	57	72	311	276	950	1
SC+E	49	91	51	63	391	1
E+SD	49	91	18	5	192	1
E	37	91	51	99	427	1
SC+F	45	128	798	843	2596	1
SD+F	43	128	325	284	990	1
SC+F+E	49	128	51	70	398	1
E+SD+F	51	128	28	14	243	1

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	20	524	747	732	2157	0
SD	20	-19	260	177	523	0
SC+E	12	0	0	-36	-36	0
E+SD	12	0	-33	-94	-235	0
SC+F	8	37	747	744	2169	0
SD+F	6	37	274	185	563	0
SC+F+E	12	37	0	-29	-29	0
E+SD+F	14	37	-23	-85	-184	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	6175	2583	977	5198	1606	0.13	0	0	0
SD	2256	925	350	1906	575	0.16	13171	24	0.00
SC+E	874	414	189	685	225	0.38	0	0	0
E+SD	240	186	112	128	74	0.86	271	5	0.02
SC+F	6175	2583	977	5198	1606	0.13	0	0	0
SD+F	2373	957	367	2006	590	0.15	13886	27	0.00
SC+F+E	874	414	189	685	225	0.38	0	0	0
E+SD+F	323	228	133	190	95	0.81	608	9	0.01

(c)

Table 21: The raw and EPOR-relative exploration data and reduction data from dining philosophers DP2 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	64	533	798	831	2589	1
SD	52	72	311	276	955	1
SC+E	50	91	51	63	396	1
E+SD	38	91	18	5	197	1
E	33	91	51	99	432	1
SC+F	54	128	798	843	2601	1
SD+F	52	128	325	284	995	1
SC+F+E	56	128	51	70	403	1
E+SD+F	51	128	28	14	248	1

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	31	442	747	732	2157	0
SD	19	-19	260	177	523	0
SC+E	17	0	0	-36	-36	0
E+SD	5	0	-33	-94	-235	0
SC+F	21	37	747	744	2169	0
SD+F	19	37	274	185	563	0
SC+F+E	23	37	0	-29	-29	0
E+SD+F	18	37	-23	-85	-184	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	6179	2587	990	5189	1597	0.13	0	0	0
SD	2260	929	363	1897	566	0.17	13171	24	0.00
SC+E	878	418	202	676	216	0.39	0	0	0
E+SD	244	190	125	119	65	0.86	271	5	0.02
SC+F	6179	2587	990	5189	1597	0.13	0	0	0
SD+F	2377	961	380	1997	581	0.16	13886	27	0.00
SC+F+E	878	418	202	676	216	0.39	0	0	0
E+SD+F	327	232	146	181	86	0.81	608	9	0.01

(c)

Table 22: The raw and EPOR-relative exploration data and reduction data from dining philosophers DP3 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	65	903	1618	2056	6197	1
SD	58	92	373	334	1268	1
SC+E	45	92	30	36	321	1
E+SD	47	92	16	4	228	1
E	53	92	30	57	342	1
SC+F	46	867	1621	2092	6238	1
SD+F	41	194	376	336	1275	1
SC+F+E	50	194	30	41	326	1
E+SD+F	52	194	16	6	230	1

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	12	811	1588	1999	5855	0
SD	5	0	343	277	926	0
SC+E	-8	0	0	-21	-21	0
E+SD	-6	0	-14	-53	-114	0
SC+F	-7	775	1591	2035	5896	0
SD+F	-12	102	346	279	933	0
SC+F+E	-3	102	0	-16	-16	0
E+SD+F	-1	102	-14	-51	-112	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	14926	6196	2524	12402	3672	0.09	0	0	0
SD	3246	1244	564	2682	680	0.11	14549	23	0.00
SC+E	768	390	199	569	191	0.38	0	0	0
E+SD	275	223	165	110	58	0.89	199	4	0.02
SC+F	14939	6204	2526	12413	3678	0.09	0	0	0
SD+F	3265	1249	566	2699	683	0.10	14664	24	0.00
SC+F+E	768	390	199	569	191	0.38	0	0	0
E+SD+F	275	223	165	110	58	0.89	208	4	0.02

(c)

Table 23: The raw and EPOR-relative exploration data and reduction data from dining philosophers DP4 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	69	596	1910	2304	7123	1
SD	51	92	442	417	1564	1
SC+E	47	93	8	2	142	1
E+SD	50	93	9	1	144	1
E	49	92	153	346	1434	1
SC+F	65	820	1930	2359	7200	1
SD+F	59	152	448	423	1578	1
SC+F+E	56	153	8	3	143	1
E+SD+F	54	153	9	3	146	1

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	20	504	1757	1958	5689	0
SD	2	0	289	71	130	0
SC+E	-2	1	-145	-344	-1292	0
E+SD	1	1	-144	-345	-1290	0
SC+F	16	728	1777	2013	5766	0
SD+F	10	60	295	77	144	0
SC+F+E	7	61	-145	-343	-1291	0
E+SD+F	5	61	-144	-343	-1288	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	17599	7121	2961	14638	4160	0.08	0	0	0
SD	3967	1539	706	3261	833	0.12	16910	23	0.00
SC+E	197	151	103	94	48	0.72	0	0	0
E+SD	153	141	104	49	37	0.93	18	1	0.06
SC+F	17696	7146	2980	14716	4166	0.08	0	0	0
SD+F	4001	1547	711	3290	836	0.12	17153	24	0.00
SC+F+E	197	151	103	94	48	0.72	0	0	0
E+SD+F	153	141	104	49	37	0.93	27	2	0.07

(c)

Table 24: The raw and EPOR-relative exploration data and reduction data from dining philosophers DP5 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	64	515	1910	2304	7115	1
SD	40	92	442	417	1556	1
SC+E	41	93	8	2	138	1
E+SD	44	93	9	1	140	1
E	34	92	153	346	1426	1
SC+F	64	692	1930	2359	7192	1
SD+F	65	151	448	423	1570	1
SC+F+E	55	151	8	3	139	1
E+SD+F	53	151	9	3	142	1

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	30	423	1757	1958	5689	0
SD	6	0	289	71	130	0
SC+E	7	1	-145	-344	-1288	0
E+SD	10	1	-144	-345	-1286	0
SC+F	30	600	1777	2013	5766	0
SD+F	31	59	295	77	144	0
SC+F+E	21	59	-145	-343	-1287	0
E+SD+F	19	59	-144	-343	-1284	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	17591	7113	2953	14638	4160	0.08	0	0	0
SD	3959	1531	698	3261	833	0.12	16910	23	0.00
SC+E	193	147	99	94	48	0.71	0	0	0
E+SD	149	137	100	49	37	0.93	18	1	0.06
SC+F	17688	7138	2972	14716	4166	0.08	0	0	0
SD+F	3993	1539	703	3290	836	0.12	17153	24	0.00
SC+F+E	193	147	99	94	48	0.71	0	0	0
E+SD+F	149	137	100	49	37	0.93	27	2	0.07

(c)

Table 25: The raw and EPOR-relative exploration data and reduction data from dining philosophers DP6 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	36019	5937	11908	743	7251107	0
SD	36247	14842	9150	0	2762960	0
SC+E	5469	6135	4419	647	529098	0
E+SD	1564	5932	2478	43	103302	0
E	36012	5890	200622	197416	8287084	0
SC+F	36081	10539	10383	471	6303446	0
SD+F	36288	14407	8463	0	2105125	0
SC+F+E	5497	6207	4419	647	529098	0
E+SD+F	1629	5966	2478	43	103302	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	7	47	-188714	-196673	-1035977	0
SD	235	8952	-191472	-197416	-5524124	0
SC+E	-30543	245	-196203	-196769	-7757986	0
E+SD	-34448	42	-198144	-197373	-8183782	0
SC+F	69	4649	-190239	-196945	-1983638	0
SD+F	276	8517	-192159	-197416	-6181959	0
SC+F+E	-30515	317	-196203	-196769	-7757986	0
E+SD+F	-34383	76	-198144	-197373	-8183782	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	7268871	7254294	4619080	2649791	2635214	1.00	0	0	0
SD	5520859	2762954	2755684	2765175	7270	0.00	45275	0	0
SC+E	758550	632726	289442	469108	343284	0.76	0	0	0
E+SD	109860	103254	58298	51562	44956	0.94	3501994	41	0.00
SC+F	6319678	6306597	4111038	2208640	2195559	1.00	0	0	0
SD+F	4205189	2105119	2097879	2107310	7240	0.00	44975	0	0
SC+F+E	758550	632726	289442	469108	343284	0.76	0	0	0
E+SD+F	109860	103254	58298	51562	44956	0.94	3501994	41	0.00

(c)

Table 26: The raw and EPOR-relative exploration data and reduction data from molecular dynamics MD3 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	36011	5526	1831	156	1455033	0
SD	13314	6381	22909	5276	435667	0
SC+E	36014	5960	2298	1119	711010	0
E+SD	1830	5607	171	5	42604	0
E	24263	5783	4616	3569	1247000	0
SC+F	36034	6942	1812	155	1437594	0
SD+F	13287	6640	22909	5276	435667	0
SC+F+E	36027	6666	2186	1033	702675	0
E+SD+F	1827	5652	171	5	42604	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	11748	-257	-2785	-3413	208033	0
SD	-10949	598	18293	1707	-811333	0
SC+E	11751	177	-2318	-2450	-535990	0
E+SD	-22433	-176	-4445	-3564	-1204396	0
SC+F	11771	1159	-2804	-3414	190594	0
SD+F	-10976	857	18293	1707	-811333	0
SC+F+E	11764	883	-2430	-2536	-544325	0
E+SD+F	-22436	-131	-4445	-3564	-1204396	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	1457291	1455148	1057730	399561	397418	1.00	0	0	0
SD	841343	435325	403207	438136	32118	0.06	82471867	336	0.00
SC+E	1098039	802388	519537	578502	282851	0.58	0	0	0
E+SD	43966	42595	32894	11072	9701	0.97	56594	3	0.00
SC+F	1439849	1437709	1045264	394585	392445	1.00	0	0	0
SD+F	841343	435325	403207	438136	32118	0.06	82471867	336	0.00
SC+F+E	1081541	792354	513095	568446	279259	0.59	0	0	0
E+SD+F	43966	42595	32894	11072	9701	0.97	56594	3	0.00

(c)

Table 27: The raw and EPOR-relative exploration data and reduction data from ray tracer RT3 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	1446	1358	301293	362901	1081729	0
SD	128	1151	12710	13425	46687	0
SC+E	714	829	69682	78117	323185	0
E+SD	55	232	82	17	553	0
E	119	799	7379	15859	76291	0
SC+F	1746	1641	301293	362901	1081729	0
SD+F	126	1269	12710	13425	46687	0
SC+F+E	748	1481	69682	78117	323185	0
E+SD+F	53	248	82	17	553	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	1327	559	293914	347042	1005438	0
SD	9	352	5331	-2434	-29604	0
SC+E	595	30	62303	62258	246894	0
E+SD	-64	-567	-7297	-15842	-75738	0
SC+F	1627	842	293914	347042	1005438	0
SD+F	7	470	5331	-2434	-29604	0
SC+F+E	629	682	62303	62258	246894	0
E+SD+F	-66	-551	-7297	-15842	-75738	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	2732043	1081728	435388	2296655	646340	0.05	0	0	0
SD	129663	44038	20943	108720	23095	0.03	2218888	2648	0.00
SC+E	865735	362456	144030	721705	218426	0.07	0	0	0
E+SD	722	538	342	380	196	0.83	2837	14	0.00
SC+F	2732043	1081728	435388	2296655	646340	0.05	0	0	0
SD+F	129663	44038	20943	108720	23095	0.03	2218888	2648	0.00
SC+F+E	865735	362456	144030	721705	218426	0.07	0	0	0
E+SD+F	722	538	342	380	196	0.83	2837	14	0.00

(c)

Table 28: The raw and EPOR-relative exploration data and reduction data from pipeline PL1 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	75	430	291	87	2189	0
SD	53	73	287	84	2177	0
SC+E	47	93	48	31	1363	0
E+SD	49	93	48	31	1375	0
E	36	92	57	44	1637	0
SC+F	45	95	291	87	2189	0
SD+F	42	95	287	84	2177	0
SC+F+E	40	95	48	31	1363	0
E+SD+F	43	95	48	31	1375	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	39	338	234	43	552	0
SD	17	-19	230	40	540	0
SC+E	11	1	-9	-13	-274	0
E+SD	13	1	-9	-13	-262	0
SC+F	9	3	234	43	552	0
SD+F	6	3	230	40	540	0
SC+F+E	4	3	-9	-13	-274	0
E+SD+F	7	3	-9	-13	-262	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	2839	2188	1066	1773	1122	0.70	0	0	0
SD	2818	2153	1060	1758	1093	0.69	5531	23	0.00
SC+E	1668	1380	665	1003	715	0.79	0	0	0
E+SD	1383	1350	671	712	679	0.98	3207	24	0.01
SC+F	2839	2188	1066	1773	1122	0.70	0	0	0
SD+F	2818	2153	1060	1758	1093	0.69	5531	23	0.00
SC+F+E	1668	1380	665	1003	715	0.79	0	0	0
E+SD+F	1383	1350	671	712	679	0.98	3207	24	0.01

(c)

Table 29: The raw and EPOR-relative exploration data and reduction data from producer-consumer PC3 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	61	1046	1007	615	9986	0
SD	60	1143	582	284	7488	0
SC+E	52	1352	88	64	3460	0
E+SD	49	1351	88	64	3469	0
E	46	95	103	85	4057	0
SC+F	62	1356	1007	615	9986	0
SD+F	41	1356	582	284	7488	0
SC+F+E	43	1460	88	64	3460	0
E+SD+F	45	1356	88	64	3469	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	15	951	904	530	5929	0
SD	14	1048	479	199	3431	0
SC+E	6	1257	-15	-21	-597	0
E+SD	3	1256	-15	-21	-588	0
SC+F	16	1261	904	530	5929	0
SD+F	-5	1261	479	199	3431	0
SC+F+E	-3	1365	-15	-21	-597	0
E+SD+F	-1	1261	-15	-21	-588	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	14235	9985	6014	8221	3971	0.57	0	0	0
SD	10274	7285	4529	5745	2756	0.59	25889	202	0.01
SC+E	4325	3514	1991	2334	1523	0.76	0	0	0
E+SD	3542	3419	1997	1545	1422	0.96	8888	49	0.01
SC+F	14235	9985	6014	8221	3971	0.57	0	0	0
SD+F	10274	7285	4529	5745	2756	0.59	25889	202	0.01
SC+F+E	4325	3514	1991	2334	1523	0.76	0	0	0
E+SD+F	3542	3419	1997	1545	1422	0.96	8888	49	0.01

(c)

Table 30: The raw and EPOR-relative exploration data and reduction data from producer-consumer PC4 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	53	654	1403	1453	5480	0
SD	48	1022	861	733	3564	0
SC+E	41	106	102	127	1404	0
E+SD	49	106	72	66	989	0
E	36	106	124	296	1746	0
SC+F	65	1283	1415	1493	5578	0
SD+F	53	1152	875	770	3689	0
SC+F+E	50	107	104	139	1432	0
E+SD+F	49	107	72	79	1002	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	17	548	1279	1157	3734	0
SD	12	916	737	437	1818	0
SC+E	5	0	-22	-169	-342	0
E+SD	13	0	-52	-230	-757	0
SC+F	29	1177	1291	1197	3832	0
SD+F	17	1046	751	474	1943	0
SC+F+E	14	1	-20	-157	-314	0
E+SD+F	13	1	-52	-217	-744	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	10923	5479	2380	8543	3099	0.35	0	0	0
SD	6544	3389	1650	4894	1739	0.42	121289	215	0.00
SC+E	2080	1486	817	1263	669	0.77	0	0	0
E+SD	1108	951	588	520	363	0.92	17209	56	0.00
SC+F	11030	5543	2418	8612	3125	0.35	0	0	0
SD+F	6666	3481	1706	4960	1775	0.43	129951	216	0.00
SC+F+E	2105	1503	828	1277	675	0.77	0	0	0
E+SD+F	1108	951	588	520	363	0.92	18790	60	0.00

(c)

Table 31: The raw and EPOR-relative exploration data and reduction data from readers-writers RW1 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	123	1422	13338	10795	47032	0
SD	131	981	9100	6438	30692	0
SC+E	94	563	5186	3480	22955	0
E+SD	95	717	2880	1606	12789	0
E	60	505	1214	2559	12312	0
SC+F	113	1182	13342	11328	47612	0
SD+F	151	826	10335	7562	35327	0
SC+F+E	82	475	4947	3758	22517	0
E+SD+F	95	628	4215	2876	18766	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	63	917	12124	8236	34720	0
SD	71	476	7886	3879	18380	0
SC+E	34	58	3972	921	10643	0
E+SD	35	212	1666	-953	477	0
SC+F	53	677	12128	8769	35300	0
SD+F	91	321	9121	5003	23015	0
SC+F+E	22	-30	3733	1199	10205	0
E+SD+F	35	123	3001	317	6454	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	111536	47021	21506	90030	25515	0.15	0	0	0
SD	73866	28259	14133	59733	14126	0.11	8059290	2471	0.00
SC+E	52055	23489	11912	40143	11577	0.20	0	0	0
E+SD	17889	11854	6883	11006	4971	0.66	2254047	973	0.00
SC+F	111317	46815	21517	89800	25298	0.15	0	0	0
SD+F	84418	32110	16225	68193	15885	0.10	11334045	2800	0.00
SC+F+E	50191	22396	11496	38695	10900	0.20	0	0	0
E+SD+F	25939	17129	9787	16152	7342	0.65	4278026	1312	0.00

(c)

Table 32: The raw and EPOR-relative exploration data and reduction data from readers-writers RW2 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	69	144	1403	1453	5480	0
SD	50	92	861	733	3564	0
SC+E	45	93	102	127	1404	0
E+SD	43	93	72	66	989	0
E	40	93	124	296	1746	0
SC+F	46	94	1415	1493	5578	0
SD+F	46	94	875	770	3689	0
SC+F+E	53	94	104	139	1432	0
E+SD+F	40	94	72	79	1002	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	29	51	1279	1157	3734	0
SD	10	-1	737	437	1818	0
SC+E	5	0	-22	-169	-342	0
E+SD	3	0	-52	-230	-757	0
SC+F	6	1	1291	1197	3832	0
SD+F	6	1	751	474	1943	0
SC+F+E	13	1	-20	-157	-314	0
E+SD+F	0	1	-52	-217	-744	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	10923	5479	2380	8543	3099	0.35	0	0	0
SD	6544	3389	1650	4894	1739	0.42	121289	215	0.00
SC+E	2080	1486	817	1263	669	0.77	0	0	0
E+SD	1108	951	588	520	363	0.92	17209	56	0.00
SC+F	11030	5543	2418	8612	3125	0.35	0	0	0
SD+F	6666	3481	1706	4960	1775	0.43	129951	216	0.00
SC+F+E	2105	1503	828	1277	675	0.77	0	0	0
E+SD+F	1108	951	588	520	363	0.92	18790	60	0.00

(c)

Table 33: The raw and EPOR-relative exploration data and reduction data from readers-writers RW3 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	70	987	1661	1647	7063	0
SD	57	1039	979	835	4426	0
SC+E	51	94	121	154	1794	0
E+SD	51	94	88	79	1311	0
E	49	93	134	299	2091	0
SC+F	63	1353	1671	1702	7190	0
SD+F	53	1143	984	854	4490	0
SC+F+E	51	95	121	157	1797	0
E+SD+F	49	95	89	93	1335	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	21	894	1527	1348	4972	0
SD	8	946	845	536	2335	0
SC+E	2	1	-13	-145	-297	0
E+SD	2	1	-46	-220	-780	0
SC+F	14	1260	1537	1403	5099	0
SD+F	4	1050	850	555	2399	0
SC+F+E	2	2	-13	-142	-294	0
E+SD+F	0	2	-45	-206	-756	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	12948	7062	3135	9813	3927	0.46	0	0	0
SD	7459	4252	2032	5427	2220	0.51	148940	223	0.00
SC+E	2409	1862	1006	1403	856	0.82	0	0	0
E+SD	1432	1278	751	681	527	0.93	24306	46	0.00
SC+F	13053	7140	3178	9875	3962	0.46	0	0	0
SD+F	7518	4299	2059	5459	2240	0.52	156159	226	0.00
SC+F+E	2409	1862	1006	1403	856	0.82	0	0	0
E+SD+F	1445	1288	757	688	531	0.93	25621	46	0.00

(c)

Table 34: The raw and EPOR-relative exploration data and reduction data from readers-writers RW4 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	78	1019	5113	4744	22702	0
SD	73	625	2616	1937	12243	0
SC+E	60	60	173	114	2975	0
E+SD	57	61	177	75	2665	0
E	62	59	58	108	2116	0
SC+F	78	693	5161	4831	23098	0
SD+F	69	630	2625	1995	12346	0
SC+F+E	63	64	180	129	3058	0
E+SD+F	59	65	183	84	2787	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	16	960	5055	4636	20586	0
SD	11	566	2558	1829	10127	0
SC+E	-2	1	115	6	859	0
E+SD	-5	2	119	-33	549	0
SC+F	16	634	5103	4723	20982	0
SD+F	7	571	2567	1887	10230	0
SC+F+E	1	5	122	21	942	0
E+SD+F	-3	6	125	-24	671	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	45673	22701	11068	34605	11633	0.35	0	0	0
SD	23187	11613	6314	16873	5299	0.35	1231920	652	0.00
SC+E	4178	3029	1688	2490	1341	0.76	0	0	0
E+SD	2915	2600	1552	1363	1048	0.93	77497	82	0.00
SC+F	46124	23032	11245	34879	11787	0.36	0	0	0
SD+F	23258	11658	6338	16920	5320	0.36	1219570	651	0.00
SC+F+E	4265	3100	1725	2540	1375	0.77	0	0	0
E+SD+F	3037	2713	1621	1416	1092	0.93	82508	82	0.00

(c)

Table 35: The raw and EPOR-relative exploration data and reduction data from readers-writers RW5 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	337	2227	5760	1764	98168	0
SD	162	2074	1619	747	31180	0
SC+E	325	1322	1788	641	63785	0
E+SD	338	1286	1587	495	60595	0
E	619	872	6577	6562	362027	0
SC+F	317	1409	5760	1764	98168	0
SD+F	156	1347	1619	747	31180	0
SC+F+E	338	1631	1788	641	63785	0
E+SD+F	332	1956	1587	495	60595	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-282	1355	-817	-4798	-263859	0
SD	-457	1202	-4958	-5815	-330847	0
SC+E	-294	450	-4789	-5921	-298242	0
E+SD	-281	414	-4990	-6067	-301432	0
SC+F	-302	537	-817	-4798	-263859	0
SD+F	-463	475	-4958	-5815	-330847	0
SC+F+E	-281	759	-4789	-5921	-298242	0
E+SD+F	-287	1084	-4990	-6067	-301432	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	129497	98163	71277	58220	26886	0.67	0	0	0
SD	37132	30547	20861	16271	9686	0.78	189609	630	0.00
SC+E	80099	64186	45758	34341	18428	0.75	0	0	0
E+SD	62388	60185	43827	18561	16358	0.96	173626	405	0.00
SC+F	129497	98163	71277	58220	26886	0.67	0	0	0
SD+F	37132	30547	20861	16271	9686	0.78	189609	630	0.00
SC+F+E	80099	64186	45758	34341	18428	0.75	0	0	0
E+SD+F	62388	60185	43827	18561	16358	0.96	173626	405	0.00

(c)

Table 36: The raw and EPOR-relative exploration data and reduction data from replicated workers RP13 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	14357	5119	1043247	753132	4823439	0
SD	5956	13922	251698	108054	1047995	0
SC+E	16302	6405	568928	328218	3259745	0
E+SD	2619	7468	95657	43268	413817	0
E	13515	6382	612954	1010878	5709125	0
SC+F	19593	12220	1043247	753132	4823439	0
SD+F	6816	14390	251698	108054	1047995	0
SC+F+E	18419	10279	568928	328218	3259745	0
E+SD+F	2654	7954	95657	43268	413817	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	842	-1263	430293	-257746	-885686	0
SD	-7559	7540	-361256	-902824	-4661130	0
SC+E	2787	23	-44026	-682660	-2449380	0
E+SD	-10896	1086	-517297	-967610	-5295308	0
SC+F	6078	5838	430293	-257746	-885686	0
SD+F	-6699	8008	-361256	-902824	-4661130	0
SC+F+E	4904	3897	-44026	-682660	-2449380	0
E+SD+F	-10861	1572	-517297	-967610	-5295308	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	9466092	4823436	2741042	6725050	2082394	0.26	0	0	0
SD	1888152	951921	578447	1309705	373474	0.29	121278050	96143	0.00
SC+E	6736469	3422551	1967867	4768602	1454684	0.26	0	0	0
E+SD	503251	379445	220569	282682	158876	0.73	19058347	34379	0.00
SC+F	9466092	4823436	2741042	6725050	2082394	0.26	0	0	0
SD+F	1888152	951921	578447	1309705	373474	0.29	121278050	96143	0.00
SC+F+E	6736469	3422551	1967867	4768602	1454684	0.26	0	0	0
E+SD+F	503251	379445	220569	282682	158876	0.73	19058347	34379	0.00

(c)

Table 37: The raw and EPOR-relative exploration data and reduction data from replicated workers RP15 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	629	2736	48232	34787	252853	0
SD	377	1480	20550	10783	105190	0
SC+E	621	1486	23408	13496	154699	0
E+SD	478	2151	14497	7225	96363	0
E	506	2306	26854	40226	321962	0
SC+F	628	2971	48232	34787	252853	0
SD+F	364	3126	20550	10783	105190	0
SC+F+E	640	3612	23408	13496	154699	0
E+SD+F	474	4041	14497	7225	96363	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	123	430	21378	-5439	-69109	0
SD	-129	-826	-6304	-29443	-216772	0
SC+E	115	-820	-3446	-26730	-167263	0
E+SD	-28	-155	-12357	-33001	-225599	0
SC+F	122	665	21378	-5439	-69109	0
SD+F	-142	820	-6304	-29443	-216772	0
SC+F+E	134	1306	-3446	-26730	-167263	0
E+SD+F	-32	1735	-12357	-33001	-225599	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	463428	252850	146396	317032	106454	0.34	0	0	0
SD	185304	97196	60228	125076	36968	0.34	5617528	8004	0.00
SC+E	288128	160836	92825	195303	68011	0.37	0	0	0
E+SD	112458	91342	57566	54892	33776	0.81	3934334	5034	0.00
SC+F	463428	252850	146396	317032	106454	0.34	0	0	0
SD+F	185304	97196	60228	125076	36968	0.34	5617528	8004	0.00
SC+F+E	288128	160836	92825	195303	68011	0.37	0	0	0
E+SD+F	112458	91342	57566	54892	33776	0.81	3934334	5034	0.00

(c)

Table 38: The raw and EPOR-relative exploration data and reduction data from replicated workers RP18 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	63	392	1870	1650	5656	6
SD	48	74	743	663	2090	2
SC+E	41	96	442	147	1502	6
E+SD	34	95	54	8	256	2
E	64	3314	11205	22139	47317	6
SC+F	42	632	1870	1650	5656	6
SD+F	34	715	803	683	2300	6
SC+F+E	39	129	442	147	1502	6
E+SD+F	33	127	114	24	462	6

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-1	-2922	-9335	-20489	-41661	0
SD	-16	-3240	-10462	-21476	-45227	-4
SC+E	-23	-3218	-10763	-21992	-45815	0
E+SD	-30	-3219	-11151	-22131	-47061	-4
SC+F	-22	-2682	-9335	-20489	-41661	0
SD+F	-30	-2599	-10402	-21456	-45017	0
SC+F+E	-25	-3185	-10763	-21992	-45815	0
E+SD+F	-31	-3187	-11091	-22115	-46855	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	12587	5655	2059	10528	3596	0.21	0	0	0
SD	4346	1981	609	3737	1372	0.20	11558	108	0.01
SC+E	3248	1560	736	2512	824	0.33	0	0	0
E+SD	339	246	135	204	111	0.77	92	9	0.10
SC+F	12587	5655	2059	10528	3596	0.21	0	0	0
SD+F	4618	2167	681	3937	1486	0.24	16482	126	0.01
SC+F+E	3248	1560	736	2512	824	0.33	0	0	0
E+SD+F	579	432	207	372	225	0.74	248	27	0.11

(c)

Table 39: The raw and EPOR-relative exploration data and reduction data from sleeping barbers SB1 input program.

C Terminated Configurations

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	151	1150	31343	24638	90649	0
SD	201	604	22866	16879	63264	0
SC+E	135	851	19349	12790	56343	0
E+SD	130	1015	11167	6408	31290	0
E	2415	628	213865	493092	984758	0
SC+F	142	830	31346	24770	90794	0
SD+F	210	760	23298	17254	64574	0
SC+F+E	139	849	19353	12922	56489	0
E+SD+F	135	824	11599	6783	32600	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-2264	522	-182522	-468454	-894109	0
SD	-2214	-24	-190999	-476213	-921494	0
SC+E	-2280	223	-194516	-480302	-928415	0
E+SD	-2285	387	-202698	-486684	-953468	0
SC+F	-2273	202	-182519	-468322	-893964	0
SD+F	-2205	132	-190567	-475838	-920184	0
SC+F+E	-2276	221	-194512	-480170	-928269	0
E+SD+F	-2280	196	-202266	-486309	-952158	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	203245	90598	33559	169686	57039	0.14	0	0	0
SD	140961	55928	22176	118785	33752	0.06	18708267	7341	0.00
SC+E	122152	56852	22530	99622	34322	0.17	0	0	0
E+SD	47683	27564	12399	35284	15165	0.49	6769341	3714	0.00
SC+F	203281	90612	33569	169712	57043	0.14	0	0	0
SD+F	144472	57053	22675	121797	34378	0.06	19156149	7487	0.00
SC+F+E	122191	56867	22540	99651	34327	0.17	0	0	0
E+SD+F	49993	28689	12898	37095	15791	0.48	7212881	3860	0.00

(c)

Table 40: The raw and EPOR-relative exploration data and reduction data from sleeping barbers SB4 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	36028	3075	484320	176323	13130000	6922
SD	36316	12172	280717	78041	8420000	3
SC+E	36075	6221	245323	97748	10761570	448
E+SD	36308	14043	154538	60075	6858336	0
E	36115	6579	318652	137135	14480000	0
SC+F	36189	8462	107674	37712	2980661	984
SD+F	36347	14268	186871	51978	5605795	0
SC+F+E	36280	13351	170504	67193	7540000	0
E+SD+F	36310	14300	110420	42890	4895085	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-87	-3504	165668	39188	-1350000	0
SD	201	5593	-37935	-59094	-6060000	0
SC+E	-40	-358	-73329	-39387	-3718430	0
E+SD	193	7464	-164114	-77060	-7621664	0
SC+F	74	1883	-210978	-99423	-11499339	0
SD+F	232	7689	-131781	-85157	-8874205	0
SC+F+E	165	6772	-148148	-69942	-6940000	0
E+SD+F	195	7721	-208232	-94245	-9584915	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	14307229	13130041	7218084	7089145	5911957	0.92	0	0	0
SD	9046789	8355377	4833261	4213528	3522116	0.92	45743350	64668	0.00
SC+E	11554895	10772540	6151656	5403239	4620884	0.93	0	0	0
E+SD	6865277	6803916	3935919	2929358	2867997	0.99	33516766	54461	0.00
SC+F	3237344	2980704	1656118	1581226	1324586	0.93	0	0	0
SD+F	6022115	5562942	3218717	2803398	2344225	0.92	29535614	42886	0.00
SC+F+E	8091637	7546938	4321143	3770494	3225795	0.93	0	0	0
E+SD+F	4899828	4856372	2808718	2091110	2047654	0.99	23200199	38743	0.00

(c)

Table 41: The raw and EPOR-relative exploration data and reduction data from disk scheduler DS1 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	36030	2033	586207	213654	15861823	8672
SD	36242	12356	312253	86737	9359041	5
SC+E	36064	6665	302288	121775	13180000	865
E+SD	36199	14247	175784	68830	7805424	1
E	36081	5549	439430	188600	19900000	0
SC+F	36131	10150	126848	44942	3491532	1366
SD+F	36256	14252	192269	53477	5770000	0
SC+F+E	36184	13792	193605	76500	8560000	0
E+SD+F	36228	14293	119919	46644	5312019	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-51	-3516	146777	25054	-4038177	0
SD	161	6807	-127177	-101863	-10540959	0
SC+E	-17	1116	-137142	-66825	-6720000	0
E+SD	118	8698	-263646	-119770	-12094576	0
SC+F	50	4601	-312582	-143658	-16408468	0
SD+F	175	8703	-247161	-135123	-14130000	0
SC+F+E	103	8243	-245825	-112100	-11340000	0
E+SD+F	147	8744	-319511	-141956	-14587981	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	17294081	15861867	8703982	8590099	7157885	0.93	0	0	0
SD	10056316	9287145	5372443	4683873	3914702	0.92	51005785	71949	0.00
SC+E	14165557	13193854	7518151	6647406	5675703	0.93	0	0	0
E+SD	7813389	7743107	4477593	3335796	3265514	0.99	38710023	62361	0.00
SC+F	3794063	3491574	1935889	1858174	1555685	0.93	0	0	0
SD+F	6198527	5725851	3313147	2885380	2412704	0.92	30483575	44180	0.00
SC+F+E	9186673	8568005	4904530	4282143	3663475	0.93	0	0	0
E+SD+F	5317229	5269891	3047699	2269530	2222192	0.99	25331009	42158	0.00

(c)

Table 42: The raw and EPOR-relative exploration data and reduction data from disk scheduler DS2 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	36050	4447	927413	219615	29990000	21
SD	36256	10549	199748	44681	6570000	0
SC+E	36085	7910	256656	83012	12170000	0
E+SD	36226	13811	122645	38192	5874295	0
E	36117	7261	282599	96813	13800000	0
SC+F	36275	12701	245509	57062	8020000	0
SD+F	36367	14272	164164	36282	5430000	0
SC+F+E	36264	13918	151804	48727	7180000	0
E+SD+F	36338	14315	96442	30033	4635284	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-67	-2814	644814	122802	16190000	0
SD	139	3288	-82851	-52132	-7230000	0
SC+E	-32	649	-25943	-13801	-1630000	0
E+SD	109	6550	-159954	-58621	-7925705	0
SC+F	158	5440	-37090	-39751	-5780000	0
SD+F	250	7011	-118435	-60531	-8370000	0
SC+F+E	147	6657	-130795	-48086	-6620000	0
E+SD+F	221	7054	-186157	-66780	-9164716	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	33244817	29990097	17137631	16107186	12852466	0.94	0	0	0
SD	7265168	6525446	3766129	3499039	2759317	0.89	38910802	44621	0.00
SC+E	13444299	12180543	6954328	6489971	5226215	0.90	0	0	0
E+SD	5880275	5832322	3366566	2513709	2465756	0.99	32683258	42042	0.00
SC+F	8868206	8019155	4597822	4270384	3421333	0.89	0	0	0
SD+F	6001490	5393296	3115502	2885988	2277794	0.89	31589272	36775	0.00
SC+F+E	7929954	7186737	4101738	3828216	3084999	0.90	0	0	0
E+SD+F	4640117	4602231	2657909	1982208	1944322	0.99	25567266	33124	0.00

(c)

Table 43: The raw and EPOR-relative exploration data and reduction data from disk scheduler DS4 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	36024	2618	802545	528986	14401190	10604
SD	36114	8645	247030	124269	4104736	2490
SC+E	36035	4969	136911	78195	4603123	4346
E+SD	36082	7226	77217	30770	2522397	2409
E	36037	3599	157684	129614	5143947	4155
SC+F	36080	7247	111268	61175	2597494	2679
SD+F	36109	7967	74080	28997	1721830	1977
SC+F+E	36081	7358	75774	40260	2576842	2228
E+SD+F	36111	7282	43290	17457	1417963	2027

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-13	-981	644861	399372	9257243	6449
SD	77	5046	89346	-5345	-1039211	-1665
SC+E	-2	1370	-20773	-51419	-540824	191
E+SD	45	3627	-80467	-98844	-2621550	-1746
SC+F	43	3648	-46416	-68439	-2546453	-1476
SD+F	72	4368	-83604	-100617	-3422117	-2178
SC+F+E	44	3759	-81910	-89354	-2567105	-1927
E+SD+F	74	3683	-114394	-112157	-3725984	-2128

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	17608612	14401228	7722382	9886230	6678846	0.85	0	0	0
SD	5113093	4023061	2281952	2831141	1741109	0.80	79471699	96253	0.00
SC+E	5118320	4613372	2429884	2688436	2183488	0.93	0	0	0
E+SD	2543126	2501487	1377355	1165771	1124132	0.99	16245521	26675	0.00
SC+F	2970141	2597531	1364863	1605278	1232668	0.91	0	0	0
SD+F	1955955	1701687	926715	1029240	774972	0.90	16274744	24943	0.00
SC+F+E	2851382	2581851	1373181	1478201	1208670	0.93	0	0	0
E+SD+F	1428121	1405120	777314	650807	627806	0.99	8477691	15091	0.00

(c)

Table 44: The raw and EPOR-relative exploration data and reduction data from disk scheduler DS7 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	36038	7325	2854899	1773213	9006579	0
SD	36054	9671	320514	157402	967201	0
SC+E	36044	7541	1757174	1028687	5700000	0
E+SD	36086	14613	636581	309869	2055708	0
E	36043	8189	1010626	2168202	7522067	0
SC+F	36148	14940	1849804	1143564	5880000	0
SD+F	36073	11424	316597	155417	955702	0
SC+F+E	36134	14495	1526923	893958	4980000	0
E+SD+F	36115	14735	571418	281616	1812125	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-5	-864	1844273	-394989	1484512	0
SD	11	1482	-690112	-2010800	-6554866	0
SC+E	1	-648	746548	-1139515	-1822067	0
E+SD	43	6424	-374045	-1858333	-5466359	0
SC+F	105	6751	839178	-1024638	-1642067	0
SD+F	30	3235	-694029	-2012785	-6566365	0
SC+F+E	91	6306	516297	-1274244	-2542067	0
E+SD+F	72	6546	-439208	-1886586	-5709942	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	19592136	9011127	4251374	15340762	4759753	0.17	0	0	0
SD	2000755	830210	460272	1540483	369938	0.11	719140148	138584	0.00
SC+E	11982175	5771693	2736436	9245739	3035257	0.20	0	0	0
E+SD	2856111	1790584	1029497	1826614	761087	0.62	509945970	266895	0.00
SC+F	12578050	5883808	2783108	9794942	3100700	0.19	0	0	0
SD+F	1977207	820644	454545	1522662	366099	0.11	710342925	136605	0.00
SC+F+E	10448212	5038921	2396965	8051247	2641956	0.21	0	0	0
E+SD+F	2528376	1573384	897617	1630759	675767	0.61	433298735	240537	0.00

(c)

Table 45: The raw and EPOR-relative exploration data and reduction data from replicated workers RP12 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	36029	6378	1496403	701784	5400914	0
SD	36086	9378	523174	207986	2018866	0
SC+E	36040	6990	916175	421679	3419295	0
E+SD	36081	9033	387088	153376	1599607	0
E	36046	7357	480884	961519	6417914	0
SC+F	36130	12882	1222803	566817	4373007	0
SD+F	36128	12178	503619	200552	1962390	0
SC+F+E	36097	11224	850187	389611	3205777	0
E+SD+F	36105	11003	377415	149814	1570605	0

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-17	-979	1015519	-259735	-1017000	0
SD	40	2021	42290	-753533	-4399048	0
SC+E	-6	-367	435291	-539840	-2998619	0
E+SD	35	1676	-93796	-808143	-4818307	0
SC+F	84	5525	741919	-394702	-2044907	0
SD+F	82	4821	22735	-760967	-4455524	0
SC+F+E	51	3867	369303	-571908	-3212137	0
E+SD+F	59	3646	-103469	-811705	-4847309	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	9799819	5402924	2816427	6983392	2586497	0.33	0	0	0
SD	3433479	1821698	1099250	2334229	722448	0.28	141058275	197266	0.00
SC+E	5942178	3420262	1799928	4142250	1620334	0.36	0	0	0
E+SD	1909990	1454631	878457	1031533	576174	0.73	96085136	145076	0.00
SC+F	7893628	4375180	2282539	5611089	2092641	0.32	0	0	0
SD+F	3311365	1772376	1069730	2241635	702646	0.29	134878469	190117	0.00
SC+F+E	5509392	3206814	1692635	3816757	1514179	0.37	0	0	0
E+SD+F	1873030	1428991	863296	1009734	565695	0.73	93884999	141714	0.00

(c)

Table 46: The raw and EPOR-relative exploration data and reduction data from replicated workers RP14 input program.

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	36068	7284	3567474	3870150	12208052	1
SD	36189	11667	1025894	1223108	3970000	1
SC+E	36086	8253	1458629	2250433	6286847	1
E+SD	36139	10583	599683	922882	2661671	1
E	36089	8723	722112	3246748	10869856	1
SC+F	36237	12376	2035248	2163348	6910000	1
SD+F	36278	14894	1001193	981585	3534082	1
SC+F+E	36195	12529	1167223	1825146	5009925	1
E+SD+F	36188	12682	568748	881842	2521013	1

(a)

Conf	Time (s)	Memory (MB)	States	Matched	Transitions	Errors
SC	-21	-1439	2845362	623402	1338196	0
SD	100	2944	303782	-2023640	-6899856	0
SC+E	-3	-470	736517	-996315	-4583009	0
E+SD	50	1860	-122429	-2323866	-8208185	0
SC+F	148	3653	1313136	-1083400	-3959856	0
SD+F	189	6171	279081	-2265163	-7335774	0
SC+F+E	106	3806	445111	-1421602	-5859931	0
E+SD+F	99	3959	-153364	-2364906	-8348843	0

(b)

Conf	E	A	I	E-I	A-I	A/E	S	D	D/S
SC	39104259	12200022	5092116	34012143	7107906	0.04	0	0	0
SD	12202031	3749493	1874935	10327096	1874558	0.04	1945181165	211956	0.00
SC+E	17748650	6636795	2592980	15155670	4043815	0.11	0	0	0
E+SD	3542841	2517959	1100345	2442496	1417614	0.65	-1515683079	126903	NaN
SC+F	20565527	6815904	2950767	17614760	3865137	0.05	0	0	0
SD+F	10650328	3277956	1690070	8960258	1587886	0.04	1599914993	217469	0.00
SC+F+E	13327809	5251111	2035187	11292622	3215924	0.12	0	0	0
E+SD+F	3356494	2385845	1040035	2316459	1345810	0.65	-1705230362	117697	NaN

(c)

Table 47: The raw and EPOR-relative exploration data and reduction data from sleeping barbers SB2 input program.