An Improved Case-Based Approach to LTL Model Checking*

Fei Pu, Wenhui Zhang, and Shaochun Wang

Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences,
P.O.Box 8718, 100080 Beijing, China
{pufei, zwh, scwang}@ios.ac.cn

Abstract. The state space explosion is the key obstacle of model checking. Even a relatively small system specification may yield a very large state space. The case-based approach based on search space partition has been proposed in [18, 19] for reducing model checking complexity. This paper extends the approach by considering wider ranges of case-bases of models and multiple case-bases such that it can be applied to more types of applications. The improved approach also combines the search space partition and static analysis or expert knowledge for guaranteeing the completeness of the cases. The case study demonstrates the potential advantages of the strategy and show that the strategy may improve the efficiency of system verification and therefore scale up the applicability of the verification approach.

1 Introduction

Designing and studying (software) systems using abstract behavioral models is becoming more and more feasible due to the increased capabilities of verification techniques developed lately. The most important of such techniques is model checking, where the validity of formally-specified requirements can be checked automatically on the model. As the state space explosion is the major bottleneck to model checking, much research is devoted to state space reduction techniques. It is crucial that such state space reduction preserves the properties under investigation. Related works can for instance be found in [2, 4–6, 8, 14]. The topics include compositional techniques [1, 16] for splitting verification tasks, bounded model checking [3] for checking satisfiability, symmetric reduction [10, 11] for applying symmetries, abstraction techniques [12] for reducing models and partial order reduction [13] for exploiting the partial order semantics for concurrent systems.

In addition to general techniques which can be used to a wide range of model, it is also important to develop techniques for special types of models in order

* This work is supported by the National Natural Science Foundation of China under Grant No. 60223005, 60373050, 60421001, and 60573012, and the National Grand Fundamental Research 973 Program of China under Grant No. 2002cb312200.

N. Guelfi and A. Savidis (Eds.): RISE 2005, LNCS 3943, pp. 190–202, 2006.

to enhance the applicability of model checking to these types of models. In previous works, we have investigated search space partition using case basis for the purpose of reduction in verification of models with non-deterministic choices and open environment. The reductions have been implemented, and show appropriate size on the subgoal, and therefore allowed for analysis of larger systems. The approach is to partition a model checking task into several cases and prove that the collection of these cases covers all possible cases, in which the former is done by model checking and the latter is done by using static analysis or by expert judgment to decide whether a model satisfies given criteria.

In [18, 19], the condition which the case-based approach can be used to model checking tasks is that there is some variable which is changed at most once during every execution of the model. Here, we extend the condition, so that we do not need to consider the number of the changes of the values of the variable during every execution of the model, as long as there is a corresponding variable satisfying some property when the variable does not take the given value. This is an extension of the condition of [18, 19], as it is a special case of the new condition. We further use multiple case-bases which can divide the task into a set of simpler tasks for reducing the need of memory in model checking. Hence, this work extends the capability of the approach such that it can be applied to wider ranges of models and to more types of applications. The modeling language of systems used in this paper is Promela – a high level specification language for system descriptions used by SPIN [7].

The paper is organized as follows. In section 2, we introduce the improved partition strategy. In section 3, we discuss static analysis of Promela models. In section 4, An application example is presented. Section 5 is concluding remarks and future work.

2 The Improved Partition Strategy

The motivation of the strategy is the verification of models with non-deterministic choice and open environment. The basic idea is to find a method for adequately representing different cases in such models, in order to reduce memory usage in model checking. The strategy partitions a model checking task into several cases and proves that the collection of these cases covers all possible case. We first give an introduction to the background of the basic strategy described in [18, 19] and then discuss the extensions of this strategy to wider ranges of case-bases and to multiple case-bases.

2.1 Basic Strategy

Let M be a system and $\overrightarrow{x}$ be the variable array of M. The system is in the state $\overrightarrow{v}$, if the value of $\overrightarrow{x}$ at the current moment is $\overrightarrow{v}$. A path of M is a sequence of states. The property of such a path can be specified by PLTL (propositional linear temporal logic) formulas [17].

- φ is a PLTL formula, if φ is of the form $z = w$ where $z \in \overrightarrow{x}$ and w is a value.
- Logical connective of PLTL include:
  $\neg(negation)$, $\wedge(conjunction)$, $\vee(disjunction)$ and $\rightarrow$ $(implication)$.
  If φ and ψ are PLTL formulas, then so are $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\varphi \rightarrow \psi$.
- Temporal operations include:
  $X(nexttime)$, $U(until)$, $\Diamond(future)$ and $\Box(always)$.
  If φ and ψ are PLTL formulas, then so are $X\varphi$, $\varphi U \psi$, $\Diamond\varphi$, and $\Box\varphi$.

Let π be a path of M. Let $HEAD(\pi)$ be the first element of π and $TAIL^i(\pi)$ be the path constructed from π by removing the first i elements of π. For convenience, we write $TAIL(\pi)$ for $TAIL^1(\pi)$. Let $\pi \models \varphi$ denote the relation "π satisfies φ".

Definition 1. *$\pi \models \varphi$ is defined as follows:*

$\pi \models x = v$ *iff the statement* $x = v$ *is true in* $HEAD(\pi)$.
$\pi \models \neg\varphi$ *iff* $\pi \not\models \varphi$.
$\pi \models \varphi \wedge \psi$ *iff* $\pi \models \varphi$ *and* $\pi \models \psi$.
$\pi \models \varphi \vee \psi$ *iff* $\pi \models \varphi$ *or* $\pi \models \psi$.
$\pi \models \varphi \rightarrow \psi$ *iff* $\pi \models \varphi$ *implies* $\pi \models \psi$.
$\pi \models X\varphi$ *iff* $TAIL(\pi) \models \varphi$.
$\pi \models \varphi U \psi$ *iff* $\exists\, k$ *such that* $TAIL^k(\pi) \models \psi$ *and* $TAIL^i(\pi) \models \varphi$ *for* $0 \leq i < k$.
$\pi \models \Diamond\varphi$ *iff* $\exists\, k$ *such that* $TAIL^k(\pi) \models \varphi$.
$\pi \models \Box\varphi$ *iff* $\pi \models \varphi$ *and* $TAIL(\pi) \models \Box\varphi$.

Let τ be a set of paths.

Definition 2. *$\tau \models \varphi$ if and only if* $\forall \pi \in \tau : \pi \models \varphi$.

Definition 3. *$\tau \not\models \varphi$ if and only if* $\exists \pi \in \tau : \pi \not\models \varphi$.

Let $HEAD(\tau)$ be the set consisting of $HEAD(\pi)$ for all $\pi \in \tau$ and $TAIL(\tau)$ be the set consisting of $TAIL(\pi)$ for all $\pi \in \tau$. From the above definitions, we can derive the following:

$\tau \models x = v$ *iff the statement* $x = v$ *is true in* s *for all* $s \in HEAD(\tau)$.
$\tau \models \neg\varphi$ *iff* $\tau \not\models \varphi$.
$\tau \models \varphi \vee \psi$ *iff there are* τ' *and* $\tau'' : \tau = \tau' \cup \tau''$ *and* $\tau' \models \varphi$ *and* $\tau'' \models \psi$.
$\tau \models \varphi \wedge \psi$ *iff* $\tau \models \varphi$ *and* $\tau \models \psi$.
$\tau \models \varphi \rightarrow \psi$ *iff there are* τ' *and* $\tau'' : \tau = \tau' \cup \tau''$ *and* $\tau' \not\models \varphi$ *and* $\tau'' \models \psi$.
$\tau \models X\varphi$ *iff* $TAIL(\tau) \models \varphi$.
$\tau \models \varphi U \psi$ *iff there are* τ' *and* $\tau'' :$
$\tau = \tau' \cup \tau''$ *and* $\tau' \models \psi, \tau'' \models \varphi$ *and* $TAIL(\tau'') \models \varphi U \psi$.
$\tau \models \Diamond\varphi$ *iff there are* τ' *and* $\tau'' :$
$\tau = \tau' \cup \tau''$ *and* $\tau' \models \varphi$ *and* $TAIL(\tau'') \models \Diamond\varphi$.
$\tau \models \Box\varphi$ *iff* $\tau \models \varphi$ *and* $TAIL(\tau) \models \Box\varphi$.

Now let τ be the set of the paths of system M and φ be the propositional linear temporal logic formula. Let $M \models \varphi$ denote the relation "M satisfies φ".

Definition 4. *$M \models \varphi$ if and only if $\tau \models \varphi$.*

Suppose that there is a model M and a formula φ, and the task is to check whether φ holds in M, i.e. we would like to prove:

$$M \models \varphi.$$

The principle of case-based partition is to partition the search space of M, so the formula φ can be proved within each portion of the search space. The technique for this characterization is to attach formulas to φ, so that in the verification of $M \models \varphi$, only the paths relevant to the attached formulas are fully explored (or paths irrelevant to the attached formulas are discarded at an early phase of model checking).

Theorem 1. *Let $\psi_1, \ldots, \psi_n$ be formulas such that $M \models \psi_1 \vee \cdots \vee \psi_n$. $M \models \varphi$ if and only if*

$$M \models \psi_i \rightarrow \varphi$$

for all $i \in \{1, 2, \ldots, n\}$.

Proof. It is obviously that $M \models \varphi$ implies $M \models \psi_i \rightarrow \varphi$. We prove that $M \models \psi_i \rightarrow \varphi$ for all $i \in \{1, 2, \ldots, n\}$ implies $M \models \varphi$ as follows.

- Let τ be the set of paths of M. Since $M \models \psi_1 \vee \cdots \vee \psi_n$, there are $\tau_1, \ldots, \tau_n$ such that $\tau = \tau_1 \cup \cdots \cup \tau_n$ and $\tau_i \models \psi_i$ for all i.
- On the other hand, we have $\tau \models \psi_i \rightarrow \varphi$, hence $\tau_i \models \psi_i \rightarrow \varphi$ and $\tau_i \models \psi_i$ for all i. Therefore, $\tau \models \varphi$. □

Remark 1. A similar strategy is the assume-guarantee paradigm in compositional reasoning. But they are different. The strategy of Theorem 1 partitions a model checking task into several cases (the verification of which is done separately by model checking), and proves that the collection of these cases covers all possible case, then establishes the correctness of the entire system. On the other hand, the assume-guarantee technique verifies each component process (or group of component processes) separately by combining the set of assumed and guaranteed properties of component processes in an appropriate manner, such that it can verify the correctness of the entire systems without constructing the global state-transition graph provided that the finite state system is composed of multiple processes running in parallel.

For a given model M, in order to be successful with this strategy, we have to be sure that the proof of $M \models \psi_i \rightarrow \varphi$ is simpler than the proof of $M \models \varphi$ for each i. Therefore τ (the set of paths representing the behavior of M) should have the following properties: τ can be partitioned into τ_i' and τ_i'' such that:

- $\tau_i' \not\models \psi_i$ and $\tau_i'' \models \varphi$;
- $\tau_i' \not\models \psi_i$ can be checked with high efficiency;
- τ_i'' is significantly smaller than τ.

For the selection of ψ_i (which determines τ_i' and τ_i''), it would be better to ensure τ_i'' $(i = 1, \ldots, n)$ be pair-wise disjoint, whenever this is possible. In addition, we shall discharge $M \models \psi_1 \vee \cdots \vee \psi_n$ by one of the following methods:

- Application of static analysis;
- Application of expert knowledge.

The reason for not verifying $M \models \psi_1 \vee \cdots \vee \psi_n$ with model checking is that the verification of this formula is not necessarily simpler than the verification of $M \models \varphi$. In order to be able to discharge $M \models \psi_1 \vee \cdots \vee \psi_n$ easily by the proposed methods, in [18,19] the formula ψ_i is restricted to be of the form $\Box(x = v_0 \vee x = v_i)$ where x is a variable and v_0, v_i are constants. We call the variable x the basic case-basis of a partitioning. Consequently, we have the following theorem.

Theorem 2. [18, 19] *Let v be a variable, $\{v_0, v_{1,}, \ldots, v_n\}$ be the range of v and v_0 be the initial value of v. Suppose that v is changed at most once during every execution of M. $M \models \varphi$ if and only if*

$$M \models \Box(v = v_0 \vee v = v_i) \rightarrow \varphi$$

for all $v_i \neq v_0$ in the range of variable v.

2.2 The Improved Strategy

In contrast to the case-based approach described above, the new approach is based on the new types of case-bases which is a generalization of basic case-bases.

Definition 5. *Let v be a variable of M, and $\{v_0, v_1, \ldots, v_n\}$ be the range of v, where v_0 is the initial value of v. If there exists a variable u satisfying: if variable v first takes value v_k after $v = v_0$, then $\Box((v \neq v_0) \rightarrow (v = v_k \vee \phi(u)))$ is true during the execution of M, where $v_k \in \{v_1, \ldots, v_n\}$ and $\phi(u)$ is a propositional formula related to u, then we call u a conjugate variable of v, and (v, u) the case-basis of M.*

Remark 2. The conjugate variable of v may be v itself. If v satisfies: v is changed at most once during each execution of M, and assume variable v first takes value v_k after $v = v_0$, where $v_k \in \{v_1, \ldots, v_n\}$, then $\Box((v \neq v_0) \rightarrow (v = v_k))$ is true during the execution of M, where $\phi(v) \equiv (v = v_k)$. Hence, the basic case-basis is the special case of the newly defined case-basis.

In order to make the analysis of the case-basis easy, the formula $\phi(u)$ is restricted to be of the form $(u{=}u_p)$ or $(u{\neq}u_p)$, where u_p is a value belonging to its range.

Lemma 1. *Let v be a variable of M and $\{v_0, v_1, \ldots, v_n\}$ be the range of v. Suppose u is a conjugate variable of v. Then we have*

$$M \models \bigvee_{i=1}^{n} \Box(v = v_0 \vee v = v_i \vee \phi(u)).$$

Proof. Let x be an arbitrary execution path of M, assume variable v first takes value v_i $(1 \leq i \leq n)$ in trace x after $v = v_0$. Since u is a conjugate variable of v, $\Box(v \neq v_0 \rightarrow (v = v_i \vee \phi(u)))$ is true during the execution of M. i.e. $\Box(v = v_0 \vee v = v_i \vee \phi(u))$ is true during the execution of M $(i \in \{1, \ldots, n\})$. Hence, $x \models \Box(v = v_0 \vee v = v_i \vee \phi(u))$. Then $M \models \bigvee_{i=1}^{n} \Box(v = v_0 \vee v = v_i \vee \phi(u))$. □

Theorem 3. *Let v be a variable of M and $\{v_0, v_1, \ldots, v_n\}$ be the range of v. Suppose u is a conjugate variable of v. $M \models \varphi$ if and only if*

$$M \models \Box(v = v_0 \vee v = v_i \vee \phi(u)) \rightarrow \varphi$$

for all $v_i \neq v_0$ in the range of variable v.

Proof. It follows from Lemma 1 and Theorem 1 by taking $\psi_i = \Box(v = v_0 \vee v = v_i \vee \phi(u))$. □

2.3 Multiple Case-Bases Strategy

A complicated (software) system may have multiple case-bases, i.e. there may be several (x_j, y_j) satisfying the condition that $M \models \psi_1 \vee \cdots \vee \psi_m$ where ψ_j is of the form $\Box(x_j = v_0 \vee x_j = v_j \vee \phi(y_j))$, and y_j is a conjugate variable of x_j. In case of two case-bases: (u_1, v_1) and (u_2, v_2), we have $M \models \psi_1^1 \vee \cdots \vee \psi_m^1$ for (u_1, v_1) and $M \models \psi_1^2 \vee \cdots \vee \psi_n^2$ for (u_2, v_2). For the model M, it can be divided in even smaller pieces τ_{ij}, where $\tau_{ij} \models \psi_i^1 \wedge \psi_j^2$. Then the verification task $\tau \models \varphi$ can be divided into a set of simpler verification tasks $\tau \models \psi_i^1 \wedge \psi_j^2 \rightarrow \varphi$. Generally, we have the following theorem.

Theorem 4. *Let $(x_i, y_i)(i = 1, \ldots, l)$ be a set of case-bases of M, where y_i is the conjugate variable of x_i, and let $\{v_{i0}, v_{i1}, \ldots, v_{in_i}\}$ be the range of x_i. Assume that $M \models \psi_1^i \vee \cdots \vee \psi_{n_i}^i$ holds, where $\psi_j^i = \Box(x_i = v_{i0} \vee x_i = v_{ij} \vee \phi(y_i))$ $(1 \leq i \leq l, 1 \leq j \leq n_i)$. Then $M \models \varphi$ if and only if*

$$M \models \psi_m^1 \wedge \psi_n^2 \wedge \cdots \wedge \psi_k^l \rightarrow \varphi$$

for $1 \leq m \leq n_1$, $1 \leq n \leq n_2$ and $1 \leq k \leq n_l$.

Proof. It can be proved with induction on the number of case-bases of M by applying Theorem 3. □

This theorem is an extension of Theorem 3, and also an extension of the basic case-basis in [18, 19].

3 Static Analysis

In order to successfully find the variables satisfying the conditions of Theorem 3, we should use static analysis to find the case-bases of models. We

consider models of particular types with a situation where the non-deterministic choice is present. Let the non-deterministic choice be represented by $choice(\overrightarrow{x}, y)$, where $\overrightarrow{x} = \{v_0, v_1, \ldots, v_n\}$ is the set of possible values of variable x and v_0 is the initial value of x, y is the conjugate variable of x. We consider two types of the $choice(\overrightarrow{x}, y)$:

do ... if :: $x == v_1$; ...; $\phi(y)$; ... ⋮ :: $x == v_n$; ...; $\phi(y)$; ... fi; ... od;	do ... if :: run $p(\ldots, v_1, \ldots)$; ...; run $p(\ldots, v_m, \ldots)$;... ⋮ :: run $p(\ldots, v_k, \ldots)$; ...; run $p(\ldots, v_n, \ldots)$;... fi; ... od;

$\phi(y)$ is restricted to be a disjunction of terms like $(u = u_p)$ or $(u \neq u_p)$, where u_p is a value belonging to its range. We refer to the first type as $choice_1(\overrightarrow{x}, y)$ and the second type as $choice_2(\overrightarrow{x}, y)$. The set of paths of a model of these types has the potential (depending on the successfully static analysis) to be divided into subsets such that each subsets satisfies one of the following formulas:

$$\Box(x = v_0 \vee x = v_1 \vee \phi(y)), \ldots, \Box(x = v_0 \vee x = v_n \vee \phi(y)).$$

The purpose of the static analysis is to show that the partition of the search space into these cases is complete, i.e. to show

$$M \models \Box(x = v_0 \vee x = v_1 \vee \phi(y)) \vee \cdots \vee \Box(x = v_0 \vee x = v_n \vee \phi(y)).$$

Basically, we analyze the model in order to determine the set of cases and to ensure that (x, y) is the case-basis of a partition (in accordance with Theorem 1), i.e. checking the following conditions:

- the range of x is $\{v_0, v_1, \ldots, v_n\}$, and the range of y is $\{u_0, u_1, \ldots, u_m\}$.
- $\phi(y)$ is of the form $(y = u_p)$ or $(y \neq u_p)$, where $u_p \in \{u_0, u_1, \ldots, u_m\}$.
- during the execution of the model, if x first takes value v_k, $\Box(x \neq v_0 \rightarrow (x = v_k \vee \phi(y)))$ is true.

To locate $choice(\overrightarrow{x}, y)$, We analyze the structure of Promela programs to find out for a given pair (x, y), whether $choice(\overrightarrow{x}, y)$ satisfies all of the conditions, in order to determine whether (x, y) can be used as the case-basis of the verification task.

Summarizing the above discussion, we refine the steps of the verification strategy as follows:

- Use static analysis to analyze the model to get the case-basis (x, y) and their ranges $\overrightarrow{v}$, $\overrightarrow{u}$.
- For each $v_i \in \overrightarrow{v}$, construct $\varphi_i = \Box(x = v_0 \vee x = v_i \vee \phi(y)) \rightarrow \varphi$ as a subgoal for verification.

– Use the model checker SPIN to check whether $M \models \varphi_i$ holds for $i = 1, \ldots, n$.
– When find more than one case-bases, we pick some or all of them to construct subgoals according to Theorem 4.

Limitations. It is worth mentioning some limitations of our static analysis. The first drawback is that it is hard to trace the value in channel, such that we cannot analyze the accurate range of variable x, if there are communication processes like $ch?x$ or $ch!x$. For the second, it may not find all of the case-bases of the Promela model because the above mentioned types of $choice(\overrightarrow{x}, y)$ do not cover all the cases that may yield the case-bases of the model. Therefore, analysis techniques need to be improved in the future research. Even so, the strategy and static analysis are very useful to model checking large models which is shown in the next section by application example.

Parallel Computation. It is easy to take advantage of parallel and networked computing power when the problem can be decomposed in independent subproblems. One problem is how to fully exploit the available computing resources. It may not be possible (with the proposed strategy) to divide a problem in such a way that all subproblems require approximately the same amount of time. It could be better with respect to the utilization of the available computing power, if there are more subproblems than available computing units. In such cases, we may estimate the difficulty of the subproblem and make a schedule for the subproblems.

4 Case Study

We have chosen an application example from security protocol verification which is of the type $choice_2(\overrightarrow{x}, y)$. The verification of security protocols is the main application area for model checking techniques (e.g. [15]). We have chosen the Needham-Schroeder-Lowe Protocol. Needham-Schroeder-Lowe Protocol is a well known authentication protocol. It aims at establishing mutual authentication between an initiator A and a responder B, after which some session involving the exchange of messages between A and B can take place. We use the model of Needham-Schroeder-Lowe protocol (the fixed version) created to the principle presented in [9]. We first consider a simple version of the protocol, then the complicated version of this protocol. The following is a description of this protocol.

$$\begin{aligned} A &\rightarrow B : \{n_a, A\}_{PK(B)} \\ B &\rightarrow A : \{n_a, n_b, B\}_{PK(A)} \\ A &\rightarrow B : \{n_b\}_{PK(B)} \end{aligned}$$

Here A is an initiator who seeks to establish a session with responder B. A selects a nonce n_a, and sends it along with its identity to B encrypted using B's public key. When B receives this message, it decrypts the message to obtain the nonce n_a. It then returns the nonce n_a along with a new nonce n_b and its identity to A, encrypted using A's public key. When A receives this message, he

should be assured that he is talking to B, since only B should be able to decrypt the first message to obtain n_a. A then returns the nonce n_b to B, encrypted using B's public key. Then B should be assured that he is talking to A.

The simple version of the protocol includes one initiator, one responder and one intruder. The property to be checked can be expressed as following PLTL formulas.

- $\psi_1 : \Box(\Box\neg IniCommitAB \vee (\neg IniCommitAB\ U\ ResRunningAB))$;
- $\psi_2 : \Box(\Box\neg ResCommitAB \vee (\neg ResCommitAB\ U\ IniRunningAB))$.

In which

- $IniRunningAB$ is true iff initiator A takes part in a session of the protocol with B;
- $ResRunningAB$ is true iff responder B takes part in a session of the protocol with A;
- $IniCommitAB$ is true iff initiator A commits to a session with B;
- $ResCommitAB$ is true iff responder B commits to a session with A.

We now consider a more complicated version of this protocol which includes two initiators, one responder, and one intruder, namely, $A1, A2$, B, and I. Similarly, the property to be checked can be represented as:

- $\psi_1 : \Box(\Box\neg IniCommitA1B \vee (\neg IniCommitA1B\ U\ ResRunningA1B))$;
- $\psi_2 : \Box(\Box\neg ResCommitA1B \vee (\neg ResCommitA1B\ U\ IniRunningA1B))$.

The Promela model includes the proctype $PIni$ which has the structure as follows:

```
proctype PIni(mtype self; mtype party; mtype nonce)
{
 mtype g1;
 atomic{
 g1=self;
 IniRunning(self, party);
 ca!self, nonce, self, party;
  }
 atomic{
 ca?eval(self), eval(nonce), g1, eval(self);
 IniCommit(self,party);
 cb!self, g1, party;
 }
}
```

Parameter $self$ represents the identity of the host where the initiator process is running, whereas *party* is the identity of the host with which the $self$ host wants to run a protocol session. Finally, *nonce* is the nonce that the initiator process will use during the protocol run.

In this proctype, Since $\Box((PIni : party \neq 0) \rightarrow (PIni : party = v \vee PIni : g1 \neq 0))$ is true provided that $PIni : party$ first takes value $v(v = I$ or $B)$ during the execution of the Promela model, and proctype $PIni$ is used in the following context:

```
if
:: run PIni(A1, I, Na1); run PIni(A2, I, Na2)
:: run PIni(A1, B, Na1); run PIni(A2, I, Na2)
:: run PIni(A1, I, Na1); run PIni(A2, B, Na2)
fi
```

thus the conditions of $choice_2(\overrightarrow{x}, y)$ are satisfied. We obtain a case-basis $(PIni : party, PIni : g1)$. Note that the range of $PIni : party$ is $\{0, B, I\}$ and $PIni : party$ is changed more than once (it may be changed twice) during an execution of the protocol. The strategy of search space partition in [18, 19] will not be applicable for this case. We also detect $(PIni : self, PIni : g1)$ and $(PIni : nonce, PIni : g1)$ as case-bases, however they have less cases than $PIni : party$ and may be used as a part of multiple base-cases. After choosing $(PIni : party, PIni : g1)$ as the case-basis, for each property ψ_i to be checked, two subgoals are constructed as follows:

- ψ_{i1}: $\Box(PIni : party = 0 \vee PIni : party = I \vee PIni : g1 \neq 0) \rightarrow \psi_i$,
- ψ_{i2}: $\Box(PIni : party = 0 \vee PIni : party = B \vee PIni : g1 \neq 0) \rightarrow \psi_i$.

Table 1. Verification of Needham-Schroeder-Lowe protocol using case-basis

Verification Task	States	Transitions
ψ_1	7179	32090
ψ_{11}	6285	28768
ψ_{12}	901	3329
ψ_2	7172	33740
ψ_{21}	6285	31332
ψ_{22}	894	2415

The verification results are shown in table 1. As shown, the maximum and minimum numbers of states during the verification have been reduced to about 87% and 12% of those of the original task respectively.

Remark 3. To illustrate how the case-based verification works, we arbitrarily choose a formula ψ_{i1}:

$$\Box(PIni : party = 0 \vee PIni : party = I \vee PIni : g1 \neq 0) \rightarrow \psi_i$$

the second term of the precondition of ψ_{i1}, namely, $PIni : party = I$ means that only those options which the first proctype $PIni(...)$ satisfying $PIni : party = I$ can be chosen to run. Thus, in the above context of $PIni$, the first and the third lines (options) can run. If $PIni : party = B$, then only the second line

(option) can run. The third term of the precondition of ψ_{i1}, i.e. $PIni : g1 \neq 0$ guarantees that during the execution of the protocol, $\Box(PIni : party = 0 \vee PIni : party = I \vee PIni : g1 \neq 0)$ is true provided that PIni:party first takes value I after PIni:party=0.

We can also use multiple case-bases to verify this protocol. As previously discussed, $(PIni : self, PIni : g1)$ and $(PIni : nonce, PIni : g1)$ can also be used as case-bases, the ranges of $PIni : self$ and $PIni : nonce$ are $\{0, A1, A2\}$ and $\{0, Na1, Na2\}$ respectively. We take

$$\{(PIni : party, PIni : g1), (PIni : self, PIni : g1), (PIni : nonce, PIni : g1)\}$$

as multiple case-bases. Then for each property to be verified eight subgoals are constructed as follows:

- ψ_{i1}: $\Box((PIni : party = 0) \vee (PIni : party = I) \vee (PIni : g1 \neq 0)) \wedge$
  $\Box((PIni : self = 0) \vee (PIni : self = A1) \vee (PIni : g1 \neq 0)) \wedge$
  $\Box((PIni : nonce = 0) \vee (PIni : nonce = Na1) \vee (PIni : g1 \neq 0)) \rightarrow \psi_i$

- ψ_{i2}: $\Box((PIni : party = 0) \vee (PIni : party = I) \vee (PIni : g1 \neq 0)) \wedge$
  $\Box((PIni : self = 0) \vee (PIni : self = A1) \vee (PIni : g1 \neq 0)) \wedge$
  $\Box((PIni : nonce = 0) \vee (PIni : nonce = Na2) \vee (PIni : g1 \neq 0)) \rightarrow \psi_i$

$\vdots$

- ψ_{i8}: $\Box((PIni : party = 0) \vee (PIni : party = B) \vee (PIni : g1 \neq 0)) \wedge$
  $\Box((PIni : self = 0) \vee (PIni : self = A2) \vee (PIni : g1 \neq 0)) \wedge$
  $\Box((PIni : nonce = 0) \vee (PIni : nonce = Na2) \vee (PIni : g1 \neq 0)) \rightarrow \psi_i$

The verification results are shown in table 2. As shown, the maximum and minimum numbers of states during the verification have been reduced to about 99%, 12% of those of the original task respectively.

Table 2. Verification of Needham-Schroeder-Lowe protocol using multiple case-bases

Verification Task	States	Transitions	Verification Task	States	Transitions
ψ_1	7179	32090	ψ_2	7172	33740
ψ_{11}	6285	28768	ψ_{21}	6285	31332
ψ_{12}	7	7	ψ_{22}	7	7
ψ_{13}	7	7	ψ_{23}	7	7
ψ_{14}	7	7	ψ_{24}	7	7
ψ_{15}	901	3329	ψ_{25}	894	2415
ψ_{16}	7	7	ψ_{26}	7	7
ψ_{17}	7	7	ψ_{27}	7	7
ψ_{18}	7	7	ψ_{28}	7	7

From the above example, we have been assured that the maximum and minimum reduced numbers of states of subgoals using multiple case-bases are not less than any those of subgoals using only one case-basis. If there are several case-bases, it will be hard to determine which case-bases get the best reduction on

the number of states of subgoals before verification of the model, then multiple case-bases are practical in model checking large (software) systems.

5 Concluding Remarks and Future Work

The results presented in this paper extend those achieved previously in [18, 19]. Firstly, the new case-bases which have wider ranges of applications are introduced such that the basic case-bases are the special case of the new case-bases. Secondly, the use of multiple case-bases for further reduction of potential high memory requirements has been considered. Since multiple case-bases can achieve better reduction on the state space than only one case-basis (in the sense of maximum and minimum reduced numbers of state space of subgoals), it may improve the efficiency of (software) system verification. Finally, the principle of static analysis for case-bases exploration is introduced.

With respect to [1], a similar strategy is proposed. The basic idea is to break the proof of temporal property $\Box\varphi$ into cases based on the value of a given variable v. However, there are two differences: first, it is only applicable to safety properties; second, when the strategy is used alone, it does not do well with the model checker SPIN, because the whole state space still have to be searched to check whether the safety property holds (in each of the subtask).

The approach presented here can also be used as the basis for utilizing parallel and networked computing power for model checking (software) systems, although, the complexity of each subgoal with respect to model checking memory (time) may be different.

Further improvement of this approach can be achieved by investigating static analysis techniques that can be used to detect the (multiple) case-bases automatically, and by using symmetric reduction for reducing the number of subgoals when multiple case-bases is used to verifying complicated (software) systems.

Acknowledgements

The authors wish to thank anonymous referees for detailed comments and very helpful suggestions.

References

1. K.L. McMillan. Verification of Infinite State Systems by Compositional Model Checking. In *Proceedings of 10th International Conference on Correct Hardware Design and Verification Methods(CHARME'99), LNCS 1703*, page 219–234. Springer, 1999.
2. R. Bloem, K. Ravi and F. Somenzi. Efficient Decision Procedures for Model Checking of Linear Time Logic Properties. In *Proceedings of 11th International Conference on Computer Aided Verification(CAV'99), LNCS 1633*. page 222–235. Springer, 1999.

3. N. Amla, R. Kurshan, K.L. McMillan, and R. Medel. Experimental Analysis of Different Techniques for Bounded Model Checking. In *Proceedings of 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems(TACAS'03), LNCS 2619*. page 34–48, Springer, 2003.
4. B.B. David, C. Eisner, D. Geist and Y. Wolfsthal. Model Checking at IBM. *Formal Methods in System Design*, 22:101–108, 2003.
5. K. Yorav, Ogumber. Static Analysis for Stats-Space Reductions Preserving Temporal Logics. *Formal Methods in System Design*, 25:67–96, 2004.
6. L.I. Millett, T. Teitelbaum. Issues in Slicing PROMELA and Its Application to Model Checking, Protocol Understanding, and Simulation. *International Journal on Software Tools for Technology Transfer*. 2(4): 343–349, 2000.
7. G.J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley, 2004.
8. E.M. Clark, O. Grumberg and D. Peled. *Model Checking*. The MIT Press, 1999.
9. P. Maggi and R. Sisto. Using SPIN to Verify Security Properties of Cryptographic Protocols. In *Proceedings of 9th International SPIN Workshop on Model Checking of Software(SPIN'02), LNCS 2318*, pages 187–204. Springer, 2002.
10. P. Godefroid, A.P. Sistla. Symmetric and Reduced Symmetry in Model Checking. In *Proceedings of 13th International Conference on Computer Aided Verification(CAV'01), LNCS 2102*. page 91–103. Springer, 2001.
11. D. Bosnacki, D. Dams and L. Holenderski. Symmetric Spin. *International Journal on Software Tools for Technology Transfer*. 4:92–106, 2002.
12. E.M. Clarke, O. Grumberg and D.E. Long. Model Checking and Abstraction. *ACM Transactions on Programming Languages and Systems*. 16(5): 1512–1542, 1994.
13. D. Peled. Ten Years of Partial Order Reduction. In *Proceedings of 10th International Conference on Computer Aided Verification(CAV'99), LNCS 1427*. page 17–28. Springer, 1998.
14. G.J. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, 1991.
15. G. Lowe: Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In *Proceedings of 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems(TACAS'96), LNCS 1055*. page 147–166, Springer, 1996.
16. S. Berezin and S. Campos and E.M. Clarke. Compositional Reasoning in Model Checking. In *Proceedings of International Symposium on Compositionality: The Significant Difference(COMPOS'97), LNCS 1536*, page 81–102, Springer, 1997.
17. E.A. Emerson. *Temporal and Modal Logic*. Handbook of Theoretical Computer Science (B): 997–1072, 1990.
18. B. Su and W. Zhang. Search Space Partition and Case Basis Exploration for Reducing Model Checking Complexity. In *Proceedings of 2th International Symposium on Automated Technology on Verification and Analysis(ATVA'04), LNCS 3299*, page 34–48. Springer, 2004.
19. W. Zhang. Combining Static Analysis and Case-Based Search Space Partition for Reducing Peek Memory in Model Checking. *Journal of Computer Science and Technology*. 18(6): 762–770, 2003.