Using adaptive symmetry reduction for LTL model checking

Igor V. Konnov, Vladimir A. Zakharov
konnov@cs.msu.su, zakh@cs.msu.su
Faculty of Computational Mathematics and Cybernetics,
Lomonosov Moscow State University, Moscow, RU-119991, Russia

1 Introduction

The main difficulty all model checking tools encounter with is the state explosion problem: the state space of a system model grows exponentially with the number of processes of the system. Various techniques (abstraction, partial order reduction, symmetry reduction, etc.) have been developed to cope with this problem by taking advantages of specific algebraic features of particular models. Symmetry reduction, for example, is based on the fact that models $M$ and $M/G$ are bisimilar for any automorphism group $G$ of a model $M$. The state space of $M/G$ is the set of all orbits of $M$ w.r.t. $G$. Therefore, the size of $M/G$ may be substantially less than that of $M$. Unfortunately, given an arbitrary model $M$, it is not easy to compute a nontrivial automorphism group $G$. To overcome this obstacle the author of [4] offered an adaptive symmetry reduction (ASR) technique which gave a possibility to benefit from symmetry in state reachability analysis without resorting explicitly to the orbits. The key idea of this approach is to start a reachability checking assuming the perfect symmetry of a transition system and refine this assumption by tracking the changes that violate the estimated symmetry. At every step of the analysis the state space is divided into meta-states — the sets of presumably symmetric states. In the course of the checking meta-states are split on-demand to conform the transitions of the system. We found that ASR is applicable to LTL model checking as well. The main contribution of our paper is a new automata-theoretic approach to LTL model checking which combines ASR and double depth-first search algorithm (DDFS) for checking the emptiness of Büchi automata.

2 Preliminaries and Computational Model

We review the basic definitions of symmetry on Kripke structures as introduced in [4].
Partitions. When a distributed system under consideration is composed of $n$ concurrent processes, one may take advantage of a symmetry by dividing the set of processes into classes. A partition of the set $\mathcal{N}_n = \{1, \ldots, n\}$ is a set $P = \{C_i\}_{i=1}^m$ of cells such that $\bigcup_{i=1}^m C_i = \mathcal{N}_n$ and $C_i \cap C_j = \emptyset$ for any $1 \leq i < j \leq m$. Partitions are written in the form $|1,3|2,4|5|$, where $\{1,3\}$, $\{2,4\}$, and $\{5\}$ are the cells. We write $\mathcal{P}_n$ for the set of all partitions of the set $\mathcal{N}_n$. A partition $P$ generates the group $\mathcal{G} (\langle P \rangle$ in symbols) of permutations $\pi$ on $\mathcal{N}_n$ such that $i$ and $\pi(i)$ belong to the same cell of $P$ for any $1 \leq i \leq n$. The partial order $\sqsubseteq$ on $\mathcal{P}_n$ is introduced as expected: $P \sqsubseteq P'$ iff $\langle P \rangle \subseteq \langle P' \rangle$. Meet operation $P \sqcup P'$ is used for computing the greatest lower bound of $P$ and $P'$.

Models. A formal model of a system of $n$ concurrent processes is a Kripke structure induced by a high-level description of the processes. Without loss of generality we assume that each process has the same set of local states $L = \{1, \ldots, \ell\}$. A high-level description $D$ of a system is a finite set of meta-transitions of the form $A \xrightarrow{\varphi,P} B$, where

- $A$ and $B$ are a source and a target local states from $L$,
- $\varphi(x,y)$ is a binary guard predicate, its arguments are a global state $x, x \in L^n$, and an integer $y, y \in N_n$, indicating an active process,
- $P$ is a partition from $\mathcal{P}_n$.

If the system is in a global state $s$ and $\varphi(s,i) = true$ then the process $i$ may pass from the local state $A$ to the local state $B$. The partition $P$ is a characteristic of $\varphi$: it generates the group $\langle P \rangle$ of permutations that preserve $\varphi$ (see below for details).

A high-level description $D$ of a concurrent system induces a Kripke structure $M_D = \langle S, s_0, \rightarrow, L \rangle$, where

- $S = L^n$ is the global state space,
- $s_0 = (1,1,\ldots,1)$ is the initial state
- $\rightarrow \subseteq S \times S$ is the transition relation defined as follows: $(s_1, \ldots, s_n) \rightarrow (t_1, \ldots, t_n)$ iff there exists a process index $i$ and a meta-transition $s_i \xrightarrow{\varphi,P} t_i$ such that $\varphi((s_1, \ldots, s_n), i) = true$ and $t_j = s_j$ holds for every $j, j \neq i$.

Let $s = (s_1, \ldots, s_n)$ be a global state and $\pi$ be a permutation on $\mathcal{N}_n$. Then we write $\pi(s)$ for the state $(s_{\pi(1)}, \ldots, s_{\pi(n)})$. We say that a meta-transition $A \xrightarrow{\varphi,P} B$ is consistent iff $\varphi(s,i) = \varphi(\pi(s), \pi(i))$ holds for every global state $s$, process index $i$, and a permutation $\pi$ from $P$. Checking the consistency of meta-transitions is a separate task which can be solved, say, with the help of SAT-solvers. In what follows we will assume that a high-level description of concurrent system is composed of consistent meta-transitions only.
Meta-states and Orbits. The key idea of [4] is to utilize a symmetry in the algorithms for reachable state exploration through the using of the extended state space \( \hat{S} = \{1, \ldots, \ell\}^n \times \mathcal{P}_n \). We will refer to the elements of \( \hat{S} \) as meta-states. A meta-state \( \hat{s} = (s, P) \) specifies the orbit which is the set of states \( \text{orbit}(\hat{s}) = \{t \in S : \exists \pi (\pi \in \langle P \rangle \land \pi(s) = t\}\) . Thus, for example, \( \text{orbit}((1, 2, 2), |1, 2| 3) = \{(1, 2, 2), (2, 1, 2)\} \). A meta-state \( \hat{s} \in \hat{S} \) is subsumed by a meta-state \( \hat{t} \in \hat{S} \) (\( \hat{s} \triangleright \hat{t} \) in symbols) iff \( \text{orbit}(\hat{s}) \subseteq \text{orbit}(\hat{t}) \).

Indexed LTL. We consider a variant of Indexed Linear Time Logic (ILTL) as introduced in [1] for the specification language. Suppose that we deal with \( n \) concurrent processes and each process has \( \ell \) local states \{1, 2, \ldots, \ell\} . By an atomic proposition we mean any pair \((i, j)\), \( i \in \mathcal{N}_n \), \( j \in \mathcal{L} \). In what follows we write \( AP \) for the set of all atomic propositions corresponding to the given distributed system. The set of ILTL formulae is the minimal set which includes all atomic propositions and all expressions of the form \( \neg \psi_1, \psi_1 \lor \psi_2, \psi_1 \land \psi_2, X \psi_1, G \psi_1, F \psi_1, \psi_1 U \psi_2 \), where \( \psi_1, \psi_2 \) are ILTL formulae. A formula \( \psi(i) \) is called a generic formula if \( i \) is the first component of all its atomic propositions. An ILTL formula \( \psi \) is closed iff every atomic proposition \((i, j)\) in \( \psi \) occurs in a scope of \( \bigvee_{i=1}^n \psi(i) \) or \( \bigwedge_{i=1}^n \psi(i) \). The satisfiability relation for atomic propositions is defined as follows. Let \( M \) be a Kripke structure and \( s = (s_1, s_2, \ldots, s_n) \) be a global state. Then \( M, s \models (i, j) \iff s_i = j \). The semantics of ILTL temporal operators is defined as usually (see [2]).

Model Checking Problem: Given a high-level description \( D \) of \( n \) concurrent processes and a closed ILTL formula \( \psi \) check whether \( M_D, s_0 \models \psi \) holds for the Kripke structure \( M_D \) induced by the high-level description \( D \).

3 Adaptive Symmetry Reduction for LTL

The basic idea of the adaptive symmetry reduction technique (ASR) for reachable state exploration proposed in [4] is as follows. Given a high-level description of a model \( M \) an ASR search runs over the extended state space \( \hat{S} \) of the model. The search starts with the initial perfectly symmetric meta-state \((s_0, |1, \ldots, n|)\). For every meta-state \((s, P)\) explored along the search a “small set” \( V \) of meta-states is constructed so that they exactly cover all successors of the states from \( \text{orbit}(s, P) \), i.e. \( \{t \mid t \in S \land \exists s'(s' \in \text{orbit}(s, P) \land s' \rightarrow t)\} = \bigcup_{(v, R) \in V} \text{orbit}(v, R) \). If a current partition does not fit a meta-transition then the meta-state is split (unwined) into several ones. Thus, in the course of the search meta-states and their partitions are refined. ASR search backtracks as soon as it reaches a meta-state.
which is subsumed by an already explored meta-state  \( \hat{w} \in \hat{S} \).

It should be noticed that meta-states are constructed on-the-fly and ASR search technique is similar to that in LTL model checking. This observation inspired us on an attempt to incorporate ASR into a double depth-first search (DDFS) algorithm developed in the framework of automata-theoretic approach to LTL model checking (see [2]). Both a Kripke structure \( M \) and the negation of a formula \( \psi \) under consideration are translated to Büchi automata \( B_M \) and \( B_{\neg \psi} \). Then the model checking problem \( M \models \psi \) is reduced to the language emptiness problem for Büchi automaton \( B_M \otimes B_{\neg \psi} \) which is a synchronous product of \( B_M \) and \( B_{\neg \psi} \). The latter is solved by means of DDFS algorithm. Its first search tries to reach an accepting state from an initial state of \( B_M \otimes B_{\neg \psi} \) and launches the nested search as soon as such state is found. The nested search seeks for a cycle via the accepting state. Our ASR modification of DDFS is as follows.

1. To capture possible symmetry in \( M \) ASR-DDFS operates with meta-states and meta-transition from the high-level description of \( M \).
2. ASR-DDFS backtracks from a meta-state \( \hat{v} \) if this meta-state is subsumed by a meta-state \( \hat{w} \) that has been visited before.
3. The nested search looks for a feasible pseudo-cycle instead of a cycle; a sequence of meta-states and meta-transitions \( \hat{s}_1, \alpha_1, \ldots, \alpha_{m-1}, \hat{s}_m \) forms a pseudo-cycle if \( \hat{s}_m \triangleleft \hat{s}_1 \). A pseudo-cycle is feasible if there exists a path \( s'_1, \alpha_1, \ldots, \alpha_{m-1}, s'_m \) such that \( s'_1 = s'_m \) and \( \hat{s}_m = (s'_m, P_m) \).

Some details of our techniques are explained below.

**Translation of an ILTL property to Büchi automaton.** We translate the negation of an ILTL formula \( \psi \) to a Büchi automaton by the well-known tableau algorithm [3]. However, the symmetry issues should be taken into account. To this end we introduce a specific modification of Büchi automaton. An annotated Büchi automaton is a sextuple \( B = \langle Q, Q_0, \delta, F, \lambda, \xi \rangle \), where \( Q \) is a set of automaton states, \( Q_0 \subseteq Q \) is a set of initial states, \( \delta \subseteq Q \times Q \) is a transition relation, \( F \subseteq Q \) is a set of accepting states, \( \lambda : Q \to 2^{AP} \) is a labelling function, and \( \xi : Q \to \mathcal{P}_n \) is a symmetry constraint. An automaton \( B \) accepts the word \( \sigma = w_1 w_2 \ldots \) from \( (2^{AP})^\omega \) iff there exists an accepting run \( \sigma = q_1, q_2, \ldots \) such that \( q_1 \in Q_0 \), \( (q_i, q_{i+1}) \in \delta \) and \( w_i \in \lambda(q_i) \) hold for every \( i \geq 1 \), and at least one state \( q \in F \) appears infinitely often in \( \sigma \).

The symmetry constraint \( \xi \) is used only in the synchronous product of \( B_M \) and \( B_{\neg \psi} \) when it is necessary to split the partitions of meta-states. This kind of splitting may be required by formulas \( \bigwedge_{i=1}^{n} \psi(i) \). When this formula is processed all the automaton’s states
that were constructed recursively from the formula are annotated with the finest partition $|1| \ldots |n|$. The symmetry constraint $\xi$ should be consistent, i.e. it imposes $(i, j) \in \lambda(q)$ iff $(\pi(i), j) \in \lambda(q)$ for any $q \in Q$, $\pi \in \langle \xi(q) \rangle$, $i \in \mathcal{N}$, and $j \in \mathcal{L}$.

**Depth-first search.** The ASR-DDFS algorithm operates on the state space $\hat{S} \times Q$, where $\hat{S}$ is the extended state of model $M$ and $Q$ is the set of states of $B_{\neg \psi}$. Its goal is to find a feasible pseudo-cycle, i.e. a path $(\hat{s}_1, q_1, \alpha_1, \ldots, \alpha_{m-1}, (\hat{s}_m, q_m))$ such that

- $\text{orbit}(\hat{s}_{i+1}) \subseteq \{t \mid \exists s \in \text{orbit}(\hat{s}_i) \land s \rightarrow t\}$ and $(q_i, q_{i+1}) \in \delta$ hold for any $i$, $1 \leq i < m$,

- $q_1 = q_m$ is an accepting state and $\hat{s}_m \triangleleft \hat{s}_1$,

- a representative state $s'_m = s'_i$ such that $\hat{s}_m = (s'_m, P_m)$ makes it possible to traverse a path $(s'_1, q_1), \ldots, (s'_m, q_m)$ in the model by applying meta-transitions $\alpha_1, \ldots, \alpha_{m-1}$.

The search starts with the pair $((s_0, |1, \ldots, n|), q_0)$, where $q_0$ is an initial state of $B_{\neg \psi}$. The dfs procedure is intended for searching for an accepting state of $B_M \otimes B_{\neg \psi}$. The ndfs procedure is used to find a feasible pseudo-cycle via an accepting state $(\hat{s}^*, q^*)$ passed from dfs. The successors function computes a set of meta-states that are successors of a given meta-state $(s, P)$ composed with the automaton’s states.

<table>
<thead>
<tr>
<th>Listing 1: DFS Procedure</th>
<th>Listing 2: Nested DFS Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>procedure</strong> dfs($\hat{s}$, $q$)</td>
<td><strong>procedure</strong> ndfs($\hat{s}$, $q$, $\hat{s}^<em>$, $q^</em>$)</td>
</tr>
<tr>
<td><strong>if</strong> $\exists \hat{\omega}((\hat{\omega}, q) \in \text{Visited} \land \hat{s} \triangleleft \hat{\omega})$</td>
<td><strong>if</strong> $\exists \hat{\omega}((\hat{\omega}, q) \in \text{Visited2} \land \hat{s} \triangleleft \hat{\omega})$</td>
</tr>
<tr>
<td><strong>return</strong> // backtrack</td>
<td><strong>return</strong> // backtrack</td>
</tr>
<tr>
<td>Visited := Visited $\cup {(\hat{s}, q)}$</td>
<td>Visited2 := Visited2 $\cup {(\hat{s}, q)}$</td>
</tr>
<tr>
<td><strong>for all</strong> $(\hat{s}', q') \in \text{successors}(\hat{s}, q)$</td>
<td><strong>for all</strong> $(\hat{s}', q') \in \text{successors}(\hat{s}, q)$</td>
</tr>
<tr>
<td><strong>if</strong> $q \in F$</td>
<td><strong>if</strong> $q' = q^* \land \hat{s}' \triangleleft \hat{s}^*$ $\land$ feasible($\hat{s}'$, ndfs_stack)</td>
</tr>
<tr>
<td>Visited2 := $\emptyset$</td>
<td>report pseudo-cycle &amp; exit</td>
</tr>
<tr>
<td>ndfs($\hat{s}$, $q$, $\hat{s}$, $q$)</td>
<td>ndfs($\hat{s}'$, $q'$, $\hat{s}^<em>$, $q^</em>$)</td>
</tr>
</tbody>
</table>

The soundness and completeness of ASR-DDFS is guaranteed by the theorems below.

**Theorem 1.** Let $(\hat{s}_1, q_1), \ldots, (\hat{s}_m, q_m)$ be a feasible pseudo-cycle found by ASR-DDFS. Then there exists an accepting run $\sigma$ of $B_M \otimes B_{\neg \psi}$ with $q_1$ appearing infinitely often in it.

**Theorem 2.** If $B_M \otimes B_{\neg \psi}$ has an accepting run then ASR-DDFS eventually finds a feasible pseudo-cycle.

Thanks to the well-known properties of DDFS for LTL [2] and Theorems 1, 2 we immediately arrive at:
Theorem 3. Let $M = \langle S, s_0, \rightarrow, L \rangle$ be a Kripke structure induced by a high-level $D$ description of $n$ concurrent processes and $\psi$ be any closed ILTL formula. Then $M_D, s_0 \models \psi$ iff the algorithm ASR-DDFS detects a feasible pseudo-cycle when applied to $B_M \otimes B_{\neg \psi}$.

Listing 3: Successors Function

```plaintext
function successors((s, P), q)
    succ := Ø
    for all meta-transitions $A \xrightarrow{\varphi,P'} B$
        for all $q' \in \delta(q)$
            $R := P \sqcup P' \sqcup \xi(q')$
            $U := \text{unwind}(s, P, R) // \text{find } U : \bigcup_{u \in U} \text{orbit}(u, R) = \text{orbit}(s, P)$
            for all states $u \in U$
                for all cells $c \in R$
                    if $\exists i \in c: u_i = A \land \varphi(u, i)$
                        $v := (u_1, \ldots, u_{i-1}, B, u_{i+1}, \ldots, u_n)$
                        if $L(v) \in \lambda(q') // \text{agreed with } B_{\neg \psi}$
                            canonicalize($v, R$) // sort states in cells
                            succ := succ $\cup \{(v, R, q')\}$
            return succ
```

References


A Proofs of the theorems

Theorem 1. Let \((\hat{s}_1, q_1), \ldots, (\hat{s}_m, q_m)\) be a feasible pseudo-cycle found by ASR-DDFS. Then there exists an accepting run \(\sigma\) of \(\mathcal{B}_M \otimes \mathcal{B}_{\neg \psi}\) with \(q_1\) appearing infinitely often in it.

Proof. Let \((\hat{s}_1, q_1), \alpha_1, \ldots, \alpha_{k-1}, (\hat{s}_k, q_k)\) be a meta-path formed by dfs procedure and \((\hat{s}_k, q_k), \alpha_k, \ldots, \alpha_{m-1}, (\hat{s}_m, q_m)\) be a meta-path reported by ndfs procedure after applying it to the accepting state \((\hat{s}_k, q_k)\).

The condition at line 15 guarantees that \(q_m = q_k\) and \(\hat{s}_m \triangleleft \hat{s}_k\). Since \(\hat{s}_m \triangleleft \hat{s}_k\), the canonical representative state \(s_m : \hat{s}_m = (s_m, P_m)\) belongs to orbit(\(\hat{s}_k\)). The feasibility of the pseudo-cycle allows one to apply meta-transitions \(\alpha_k, \ldots, \alpha_m\) starting from \(s_m\) and thus form a path \(\sigma = (s'_k, q_k), \ldots, (s'_m, q_m)\) in \(\mathcal{B}_M \otimes \mathcal{B}_{\neg \psi}\) with \(s'_k = s'_m = s_m\). Since \(q_m = q_k\), the path \(\sigma\) is a cycle via the accepting state \(q_k\).

It remains to prove that \((s_m, q_k)\) is reachable from some initial state. To this end one has to find permutations \(\pi_0, \pi_1, \ldots, \pi_k \in \mathcal{P}_n\) such that the sequence \((\pi_0(s_0), q_0), \ldots, (\pi_k(s_k), q_k)\) forms a path in \(\mathcal{B}_M \otimes \mathcal{B}_{\neg \psi}\), where each \(s_i\) is a canonical representative of orbit(\(\hat{s}_i\)), i.e. \(\hat{s}_i = (s_i, P_i)\). We prove this claim by induction on index \(i\), \(0 \leq i \leq k\).

If \(i = k\) then \(s_m \in \text{orbit}(\hat{s}_k)\) implies that there exists a permutation \(\pi_k \in \langle P_k \rangle\) such that \(\pi_k(s_k) = s_m\).

Suppose that \(i = p, 0 \leq p < k\), and the required permutations \(\pi_{p+1}, \ldots, \pi_k\) are constructed. Since dfs procedure performs a recursive call from \((\hat{s}_p, q_p)\) to \((\hat{s}_{p+1}, q_{p+1})\) then \((\hat{s}_{p+1}, q_{p+1})\) \(\in \text{successors}(\hat{s}_p, q_p)\). As it can be seen from the lines 21 – 30, there exists a state \(u \in \text{orbit}(\hat{s}_p)\) and an index \(j, 1 \leq j \leq n\), such that:

- a meta-transition \(\alpha_p : A \xrightarrow{\varphi, P} B\) makes a step from the state \(u\) to state \(v = (u_1, \ldots, u_{j-1}, B, u_{j+1}, \ldots, u_n) \in \text{orbit}(\hat{s}_{p+1})\),
- the state \(v\) satisfies constraint \(L(v) \in \lambda(q_{p+1})\),
- state \(v\) is transformed to \(s_{p+1}\) via call to canonicalize\((v, P_{p+1})\) at line 30, i.e. there exists a permutation \(\rho \in \langle P_{p+1} \rangle\) such that \(\rho(v) = s_{p+1}\).

As far as \(P_p \subseteq P, P_{p+1} \subseteq P_p\) and \(P_{p+1} \subseteq \xi(q_{p+1})\), the consistency of \(\alpha\) and the consistency of symmetry constraint \(\xi\) bring us to the conclusion that \(\alpha\) is enabled in \(\rho(v)\) and \(L(\rho(v)) \in \lambda(q_{p+1})\). Thus the transition from state \((\rho(u), q_p)\) to state \((\rho(v), q_{p+1})\) is enabled in \(\mathcal{B}_M \otimes \mathcal{B}_{\neg \psi}\). By the same argument one may ascertain that \(\alpha\) is applicable to
\( \pi_{p+1}(\rho(u)) \) and, therefore, it is possible to make a step from \( \pi_{p+1}(\rho(u)) \) to \( \pi_{p+1}(\rho(v)) \) such that \( L(\pi_{p+1}(\rho(v))) \in \lambda(q_{p+1}) \). Thus we obtain the required permutation \( \pi_p = \rho \circ \pi_{p+1} \).

As soon as \( \pi_0, \ldots, \pi_k \) are constructed, the path \((\pi_0(s_0), q_0), (\pi_1(s_1), q_1), \ldots, (\pi_k(s_k), q_k)\) is built. This path leads to the cycle \((s'_k, q_k), \ldots, (s'_m, q_m)\) via the accepting state \((s_m, q_k)\) in \( B_M \otimes B_{-\psi} \), where \( s'_k = s'_m = s_m, q_m = q_k \).

This finishes the proof.

Note that the condition \( \hat{s}_m \prec \hat{s}_k \) only is not sufficient to provide the feasibility of a pseudo-cycle. If the additional requirement \( \text{feasible}(\hat{s}', \text{ndfs}, \text{stack}) \) is missed then \( \text{ndfs} \) may report a pseudo-cycle which does not correspond to any accepting run of \( B_M \otimes B_{-\psi} \). This happens when states in \( \text{orbit}(\hat{s}_m) \) cause a deadlock. We missed the feasibility requirement in the previous version of the paper.

To prove Theorem 2 we need two basic claims on the properties of meta-states. In what follows we write \( \text{orbit}(\hat{s}, q) \) to denote the set \( \{ (s, q) \mid s \in \text{orbit}(\hat{s}) \} \) for \( \hat{s} \in \hat{S} \) and \( q \in Q \).

**Claim 1.** If successors function is applied to \((\hat{s}, q)\) then it returns a set of meta-states that cover exactly the set

\[
\text{post}(\text{orbit}(\hat{s}, q)) = \{ (t, q') \mid s \in \text{orbit}(\hat{s}) \land \exists A. A \xrightarrow{t \in P} B : \exists j, 1 \leq j \leq n : s_j = A \land t_j = B \land q' \in \delta(q) \land L(t) \in \lambda(q') \}
\]

**Proof.** Follows from the definition of the function.

**Claim 2.** If a meta-state \((\hat{t}, q)\) is reached from a meta-state \((\hat{s}, q)\) in the course of the same nested search and \( \text{orbit}(\hat{s}) \cap \text{orbit}(\hat{t}) \neq \emptyset \) then \( t \triangleleft s \).

**Proof.** Let \((\hat{s}_1, q_1), \ldots, (\hat{s}_m, q_m)\) be a sequence of meta-states formed in the course of the search such that \( \hat{s}_i = (s_i, P_i) \) for all \( i, 1 \leq i \leq m \). Then \( P_1 \supseteq \cdots \supseteq P_m \). Suppose that \( \text{orbit}(\hat{s}_1) \cap \text{orbit}(\hat{s}_m) \neq \emptyset \). Then for any \( s \in \text{orbit}(\hat{s}_1) \cap \text{orbit}(\hat{s}_m) \) and any permutation \( \pi \in \langle P_m \rangle \) we have \( \pi \in \langle P_1 \rangle \) and, hence, \( \pi(s) \in \text{orbit}(\hat{s}_1) \). Thus, \( \text{orbit}(\hat{s}_m) \subseteq \text{orbit}(\hat{s}_1) \) and \( \hat{s}_m \prec \hat{s}_1 \).
Theorem 2. If $B_M \otimes B_\neg \psi$ has an accepting run then ASR-DDFS eventually finds a feasible pseudo-cycle.

Proof. Suppose that the automaton $B_M \otimes B_\neg \psi$ reaches a cycle via an accepting state.

As it follows from Claim 1 at each step of ASR-DDFS computation $dfs$ procedure being called with a meta-state $(\hat{s}, q)$ as an input explores the set of successors of $(\hat{s}, q)$ that exactly cover the set of states $post(orbit(\hat{s}, q))$. Thus, $dfs$ either terminates and reports a feasible pseudo-cycle or reaches some meta-state that covers an accepting state of the cycle. Let $(\hat{s}^*, q^*)$ be the first such a state that $ndfs$ procedure is called for.

By Claim 1 the nested search explores the set of successors of $(\hat{s}, q)$ that exactly covers $post(orbit(\hat{s}, q))$ as well. For a cycle $(u_1, r_1), \ldots, (u_m, r_m)$ with $u_m = u_1, r_m = r_1 = q^*$ the nested search eventually visits a meta-state $(\hat{t}, q)$ such that $u_m \in orbit(\hat{t})$. As $u_m \in orbit(\hat{t})$ and $u_1 \in orbit(\hat{s})$, then by Claim 2, we have that $\hat{t} \triangleleft \hat{s}$.

So the nested search detects a pseudo-cycle $(\hat{s}_1, q_1), \ldots, (\hat{s}_k, q_k)$ such that $q = q_1 = q_k$ and $s_k \triangleleft s_1$. It remains to prove that the pseudo-cycle is a feasible one. If all the states of the cycle $(u_1, r_1), \ldots, (u_m, r_m)$ belong to the orbits of the same pseudo-cycle then the feasibility follows immediately. Suppose that for some $j, 1 \leq j < k$, the states $(u_j, r_j)$ and $(u_{j+1}, r_{j+1})$ belong to the orbits of states $(\hat{s}, q)$ and $(\hat{t}, q')$ of two different pseudo-cycles $C_1$ and $C_2$ correspondingly. Let $\hat{s} = (s, P_s)$ and $\hat{t} = (t, P_t)$. Three cases are possible:

- If $P_t \subseteq P_s$ then the suffix $(u_j, r_j), \ldots, (u_k, r_k)$ belongs to orbits of $C_1$ as well and $\hat{s} \triangleleft \hat{t}$ does not hold. Otherwise, not all the successors of $orbit(\hat{s}, q)$ are visited, contrary to the Claim 1.

- If $P_t \not\subseteq P_s$ and $P_s \not\subseteq P_t$ then by Claim 1 there must be a meta-transition from $(\hat{s}, q)$ to a meta-state $(\hat{s}', q'')$ such that $u_{j+1} \in orbit(\hat{s})$. It should be noticed that in this case $\hat{s}'$ is not subsumed by $\hat{t}$. Therefore, $ndfs$ builds a meta-path, which orbits cover a suffix $(u_j, r_j), \ldots, (u_k, r_k)$.

- If $P_s \subseteq P_t$ then any successor meta-state $(\hat{s}', q')$ of $(\hat{s}, q)$ such that $u_{j+1} \in orbit(\hat{s})$ is subsumed by $(\hat{t}, q')$. This contradicts to the assumption that $C_1$ is a pseudo-cycle.

Thus, if $B_M \otimes B_{\neg \psi}$ has a reachable cycle via an accepting state then $ndfs$ procedure eventually reports a feasible pseudo-cycle. \qed
Remark (on the nested search). Note, that \textit{dfs} makes the set \textit{Visited2} empty (see line 8) every time before running the nested search \textit{ndfs}. This prevents the nested search from reusing the results of the previous searches and causes some inefficiency. The classic DDFS does not clear \textit{Visited2} and thus performs more efficiently. But the direct application of this trick in the case of ASR-DDFS is not correct, since we found that it might lead to a missed feasible pseudo-cycle due to the subsumption effects.

In fact, this is a major drawback in the performance of our algorithm. Both the theoretical improvement of ASR-DDFS and its practical applicability are the topics of our future research.