Backward Multiple Time-frame Expansion
for Accelerating Sequential SAT

Kousuke Torii† Kazuhiro Nakamura‡ Kazuyoshi Takagi† Naofumi Takagi‡
†Graduate School of Information Science, Nagoya University, Japan
‡Graduate School of Informatics, Kyoto University, Japan
{kutorii,nakamura}@takagi.is.nagoya-u.ac.jp
{ktakagi,takagi}@i.kyoto-u.ac.jp

Abstract—Sequential SAT is a formal verification problem, which finds an ordered sequence of input assignments to a sequential circuit, such that a desired objective is satisfied, or proving that no such sequence exists. Efficient algorithm for sequential SAT solver is required to deal with sequential circuits which have large state space. In this paper, we demonstrate backward multiple time-frame expansion (BMTE) a circuit expansion method for sequential SAT solver that supports it. Our algorithm improves the efficiency of state reduction and accelerates sequential SAT. Experimental results show that our method can speed up sequential SAT solving.

I. INTRODUCTION

As VLSI technology advances, it becomes possible to design larger and more complex logic circuits. Verification of logic circuits becomes increasingly important as circuit designs are getting larger and more complex. Formal verification is a remarkable method that mathematically proves the correctness of logic circuits.

Sequential SAT is a formal verification problem[1, 2, 3]. Sequential SAT is the problem of finding an ordered sequence of input assignments to a sequential circuit, such that a desired objective is satisfied, or proving that no such sequence exists.

An algorithm for sequential SAT solvers with efficient state reductions has been proposed[1, 3]. The algorithm traces state transitions from the states satisfying a desired objective toward the initial state of sequential circuit. State reductions speed up the solving process by merging states and improve the efficiency of sequential search. Efficient high speed sequential SAT solver is required to deal with sequential circuits which have large state space.

In this paper, we demonstrate backward multiple time-frame expansion (BMTE) and present a new circuit expansion method for sequential SAT solver that supports it. In our approach, first, given sequential circuit is expanded with our circuit-expansion method for BMTE-based sequential SAT. Second, the expanded sequential circuit is fed to a sequential SAT solver with the conventional backward single time-frame expansion (BSTE). By the two-step process, we solve the sequential SAT problem for given circuit using BMTE.

When the initial state is far from the state satisfying desired objective, compared with BSTE, the proposed BMTE is suitable for merging states in different timeframes and pruning the state space for search. We show our promising experimental result.

The remainder of this paper is organized as follows: Sequential SAT is described in Sect. A.; backward multiple time-frame expansion for accelerating sequential SAT is described in Sect. III.; the proposed method is evaluated in Sect. IV.; and conclusions are presented in Sect. V.

II. PRELIMINARIES

Definition 2.1 An finite state machine (FSM) M is a 6-tuple ($S, \Sigma, \Gamma, \delta, \lambda, q_0$) where

- $S$ is a finite set of states,
- $\Sigma$ is an input alphabet,
- $\Gamma$ is an output alphabet,
- $\delta : S \times \Sigma \rightarrow S$ is a state transition function,
- $\lambda : S \times \Sigma \rightarrow \Gamma$ is an output function,
- $q_0 (\in S)$ is the initial state.

The behavior of $M = (S, \Sigma, \Gamma, \delta, \lambda, q_0)$ with respect to an input sequence $a_1a_2 \ldots a_n(a_i \in \Sigma)$ is a sequence of states $q_0q_1 \ldots q_n(q_i \in S)$ and a sequence of outputs $a_1a_2 \ldots a_n(a_i \in \Gamma)$ satisfying $q_i = \delta(q_{i-1}, a_i)$.

Let $\Sigma^*$ be a set of all input sequences over $\Sigma$, and $\Sigma^k$ be a set of input sequences with length $k$. We use $\varepsilon$ as the sequence of length 0.

To represent the behavior of $M$, the domain of $\delta$ and $\lambda$ are extended, and $\delta^* : S \times \Sigma^* \rightarrow S$ and $\lambda^* : S \times \Sigma^* \rightarrow \Gamma^*$ are introduced.

Definition 2.2 $\delta^* : S \times \Sigma^* \rightarrow S$ is defined as follows.

- $\delta^*(q, \varepsilon) = q$
- $\delta^*(q, xa) = \delta(\delta^*(q, x), a), (a \in \Sigma, x \in \Sigma^*)$
Definition 2.3 A set of reachable states (RS) from the initial state \( q_0 \) of an FSM \( M \) is defined as follows.

\[ RS(M) = \{ q | \exists \omega \in \Sigma^*, q = \delta^*(q_0, \omega) \}, (\omega \in \Sigma^*) \]

A. Sequential SAT

Sequential SAT is the problem of finding an ordered sequence of input assignments to a sequential circuit, such that a desired objective is satisfied, or proving that no such sequence exists[1, 2, 3]. By setting the objectives as not to satisfy the property, e.g., \( G(\neg p) \), sequential SAT solvers can be used for reachability analysis to the state with unbounded cycles.

Relevant part of the sequential SAT algorithm[3] is described in Algorithm 1. The procedure select-a-frame-objective() selects a state named frame objective\( f_o \) to be solved next from objective lists\( (OL) \) which is storing reached states in search. The procedure SAT-solve() performs backward time-frame expansion[1, 3] by solving the SAT problem to find a new state named frame solution \( f_s \) from \( f_o \) on combinational part \( C \) of given sequential circuit. In the algorithm, it is assumed that each frame objective produces at most one frame solution \( f_s \), where a direct single state transition from \( f_s \) to \( f_o \) exists.

The procedure solution-state-reduction() performs solution state reduction[1, 3]. It can merge states in the same time-frame, where time-frame is the set of the states that satisfy a desired objective within the same number of transitions. The state reduction speed up the algorithm by merging states to prune the state space for search. Fig. 1 shows solution state reduction. In Fig. 1, state transitions of a sequential circuit and backward traversal of sequential SAT are denoted with thin and bold arrows, respectively. The state 1111 is the desired initial objective state. In Fig. 1, time-frame 0, 1 and 2 are describes. The state 1011 is a \( f_s \) obtained from \( f_o \), 1111. 0011 has also direct state transition to 1111, hence, it is merged with 1011 and, subsequently, the merged state -011 is obtained. Here, “-” means unnecessary value assignment. The procedure convert-to-clause() converts \( f_s \) into state constraints for avoiding a state included in \( f_s \) is obtained again by backward time-frame expansion. The solving process continues until either one of the following two cases is occurred. One case is when \( f_s \) includes the initial state \( q_0 \) of given sequential circuit. In this case, the solution is found(SAT). The other case is when objective list becomes empty. In this case, the solution has never found. So, the answer of the original problem is also UNSAT.

Algorithm 1 SequentialSAT-solver\( (C, obj, q_0) \)

\( C \): combinational part of given sequential circuit
\( obj \): desired initial objective states
\( q_0 \): initial states
\( f_o, f_s \): frame objective, frame solution
\( SC \): state constraints(initial constraint \( SC = \emptyset \))
\( OL \): objective list (initially \( OL = \{ obj \} \))

1: \( \text{while } OL \neq \emptyset \) do
2: \( f_o \leftarrow \text{select-a-frame-objective}(OL) \);
3: \( f_s \leftarrow \text{SAT-solve}(C, f_o, SC) \);
4: \( \text{if } f_s \neq \text{NULL} \) then
5: \( f_s' \leftarrow \text{solution-state-reduction}(C, f_s) \);
6: \( \text{if } q_0 \in f_s' \) then
7: \( \text{return SAT} \)
8: \( \text{else} \)
9: \( sc \leftarrow \text{convert-to-sc}(f_s') \);
10: \( SC \leftarrow sc + \{ sc \} \);
11: \( OL \leftarrow OL + \{ f_s' \} \);
12: \( \text{end if} \)
13: \( \text{end if} \)
14: \( \text{end while} \)
15: \( \text{return UNSAT} \)

III. Backward Multiple Time-frame Expansion for Accelerating Sequential SAT

A. Backward Multiple Time-frame Expansion

In this paper, we classify two types of backward time-frame expansion in sequential SAT algorithm according to the number of state transitions between \( f_s \) and \( f_o \): backward single time-frame expansion(BSTE) and backward multiple time-frame expansion(BMTE).

BSTE-based sequential SAT solver traces state transitions from the states satisfying a desired objective toward the initial state using backward single time-frame expansion, which back traces one state transition. BMTE-based sequential SAT solver traces state transitions from the states satisfying a desired objective toward the initial state using backward multiple time-frame expansion, which back-traces more then one state transitions.

The backward time-frame expansion used in sequential SAT solver[1, 3] based on Algorithm 1 is classified as a BSTE, where the number of state transitions between \( f_s \) and \( f_o \) is one. In this paper, we propose a new algorithm that supports BSTE.

B. A New Algorithm for Accelerating Sequential SAT

In BSTE-based sequential SAT solver, a frame solution \( f_s \) is computed from the frame objective \( f_o \) by using SAT-solve() as shown in Algorithm 1. In the while-loop in Algorithm 1, SAT solver is used to obtain \( f_s \) from \( f_o \) repeatedly. When the initial state is far from the state satisfying desired objective, a number of SAT-solve() are
called, it spends much time. Furthermore, BSTE-based sequential SAT solver can not merge states in different time-frames.

We speed up solving process by using BMTE which traces multiple state transitions at once. Furthermore, we improve solution state reduction by using BMTE which can deal with states in different time-frames. For achieving these two effects, we propose a circuit expansion method. In our approach, first, given sequential circuit is transformed. Second, the transformed sequential circuit is fed to BSTE-based sequential SAT solver. By this way, BMTE-based sequential SAT is performed to original given circuit.

BMTE-based sequential SAT solver traces state transitions from the states satisfying a desired objective toward the initial state using backward multiple time-frame expansion which back traces more than one state transitions.

Our BMTE-based sequential SAT algorithm is described in Algorithm 2. The procedure circuit-expansion() performs the circuit expansion which enables backward multiple time-frame expansion(BMTE) to accelerate solving process, and transformed circuit $CC$ is obtained. The procedure select-a-frame-objective() and convert-to-sc() are the same in Algorithm 1. The procedure SAT-solve() solves the SAT problem to find a new $f_s$ from $f_o$ on $CC$. The procedure solution-state-reduction() performs solution state reduction on $CC$. It merges states in single time-frame on $CC$, in the multiple time frames on $C$. The state reduction speed up the algorithm by merging states to prune the state space for search. The pruned space in Algorithm 2 is larger than that of Algorithm 1. In the algorithm, it is assumed that each frame objective produces at most one frame solution, where a direct single state transition on $CC$, a direct multiple state transitions on $C$, from $f_s$ to $f_o$ exists. The solving process continues until either one of the following two case is occurred. One case is when $f_s'$ includes the initial state $q_0$ of given sequential circuit. In this case, the solution is found(SAT).

**Algorithm 2** BMTE-SequentialSAT-solver($C, obj, q_0, n$)

$C$: combinational part of given sequential circuit $n$: the number of BMTE $CC$: combinational part of the transformed circuit $obj$: desired initial objective states $q_0$: initial states $f_o, f_s$: frame objective, frame solution $SC$: state constraints(initial constraint $SC = \emptyset$) $OL$: objective list (initially $OL = \{ obj \}$)

1: $CC \leftarrow$ circuit-expansion($C, n$)
2: while $OL \neq \emptyset$ do
3: $f_o \leftarrow$ select-a-frame-objective($OL$);
4: $f_s \leftarrow$ SAT-solve($CC, f_o, SC$);
5: if $f_s \neq$ NULL then
6: $f'_s \leftarrow$ solution-state-reduction($CC, f_s$);
7: if $q_0 \in f'_s$ then
8: return SAT
9: else
10: $sc \leftarrow$ convert-to-sc($f'_s$);
11: $SC \leftarrow sc + \{ sc \}$;
12: $OL \leftarrow OL + \{ f'_s \}$; the n copies of circuit C
13: end if
14: end if
15: end while
16: return UNSAT

The other case is when objective list becomes empty. In this case, the solution has never found(UNSAT). So, the answer of the original problem is UNSAT, too.

**C. Circuit Expansion for BMTE**

We propose a circuit-expansion method which enables BMTE to accelerate sequential SAT solving process. For a given number of cycles $n$, the circuit-expansion method generates a sequential circuit whose 1-cycle behavior is at most n-cycle behavior, one of 1-cycle, 2-cycle, \ldots, n-cycle behavior, of the original sequential circuit. It is a reachability-preserving circuit-expansion method.

Fig. 2 shows the circuit-expansion for BMTE. In Fig. 2, C, FFs, PI, PO, PPI and PPO represent combinational part, flip-flops, primary inputs, primary outputs, pseudo primary inputs and pseudo primary outputs of transformed sequential circuit, respectively. FFs in the transformed and the original circuit are identical. Combinational part of transformed circuit $CC$ consists of n Cs, where C is combinational part of the original circuit. In Fig. 2, PIi, PPIi, POi, and PPOi represent PI, PPI, PO and PPO of i-th C, respectively, where $i = 1, 2, \ldots, n$. PI consists of PIi's and a new $\lceil \log n \rceil$-bit input newin, where $i = 1, 2, \ldots, n$. The value of PPO' is decided from PPO1, PPO2, \ldots, PPOn by the selector, where PPO' = PPOn when the value of newin is n. The value of PO' is decided.
from PO1, PO2, ..., PO\textsubscript{n} by the selector, where PO' = PO\textsubscript{n} when the value of newin is n.

In our approach, sequential SAT solving process is performed to the circuit obtained by the circuit-expansion in place of the original one. The transformed circuit and the original circuit have the same result of sequential SAT. When the result of sequential SAT for the original circuit is UNSAT(SAT), then that for the transformed circuit is UNSAT(SAT), and vice versa.

D. Solution State reduction in BMTE-based Sequential SAT

Solution state reduction, which is performed in our BMTE-based sequential SAT algorithm (line 6 in Algorithm 2), is shown in Fig. 3. In Fig. 3, state transitions of a sequential circuit and backward traversal of sequential SAT are denoted with thin and bold arrows, respectively. State transitions in Fig. 3 are obtained from state transitions in Fig. 1 by our circuit-expansion. The state 1111 is the desired initial objective state. Because of existence of new two state transitions, from the state 1001 to the state 1111 and from the state 0001 to the state 1111, there are two time-frames in Fig. 3, which are 0 and 1.

The state 1011 is a frame solution, \( f_s \), obtained from the state 1111, \( f_o \), by the procedure SAT-solve() (Algorithm 2). Here, states 0011, 1001 and 0001 are in the same time-frame 1 with \( f_s \), and they have direct state transition to \( f_o \). Hence, they are effectively merged with \( f_s \), by the procedure solution-state-reduction() (Algorithm 2) and the merged state -0-1, \( f'_s \), is obtained. Here, -0-1 is a cube representing the four states. The state space of merged state in BMTE is larger than that in BSTE (Fig. 1). It prunes a larger state space needed to be search.

Theorem 1 Given an FSM \( M \) having an initial state \( q_0 \). Let \( M' \) be a new FSM which is derived from \( M \) by adding transitions from every state \( q_i \) to the states which are reachable from \( q_i \) within \( n \) state transitions. Then \( RS(M) = RS(M') \).

The above theorem ensures that by adding transitions from every state \( q_i \) to the states which are reachable from \( q_i \) within \( n \) state transitions, the set of reachable states from the initial state of the machine and reachability from the initial state to the states which satisfying a desired objective are not altered.

IV. Experimental Results

We have compared our BMTE-based sequential SAT solver and BSTE-based sequential SAT solver which is Seq-SAT[3]. In our BMTE-based sequential SAT solver, given circuit is firstly transformed to the circuit for BMTE-based sequential SAT. Secondly, the transformed circuit is fed to BSTE-based sequential SAT solver which Seq-SAT[3].

Table I shows the experimental results, where BMTE-based and BSTE-based sequential SAT solvers are applied to 20-bit binary counter. In Table I, “org.” is elapsed CPU seconds of MSTE-based sequential SAT solver, Seq-SAT[3]. “ours” is elapsed CPU seconds of our BMTE-
TABLE II
Comparison of processing time using s13207 (time limit is 140,000s).

<table>
<thead>
<tr>
<th>Property</th>
<th>CPU time [sec]</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>org. [3]</td>
<td>ours (2 steps)</td>
</tr>
<tr>
<td>F(g4267 = 1)</td>
<td>time-out</td>
<td>79297</td>
</tr>
<tr>
<td>F(g4316 = 1)</td>
<td>1293</td>
<td>628</td>
</tr>
<tr>
<td>F(g4370 = 1)</td>
<td>time-out</td>
<td>28943</td>
</tr>
<tr>
<td>F(g4371 = 1)</td>
<td>time-out</td>
<td>76909</td>
</tr>
<tr>
<td>F(g4372 = 1)</td>
<td>time-out</td>
<td>78618</td>
</tr>
<tr>
<td>F(g4373 = 1)</td>
<td>time-out</td>
<td>64461</td>
</tr>
<tr>
<td>F(g4655 = 1)</td>
<td>857</td>
<td>102</td>
</tr>
<tr>
<td>F(g4661 = 1)</td>
<td>time-out</td>
<td>time-out</td>
</tr>
</tbody>
</table>

TABLE III
Comparison of processing time using s13207.1 (time limit is 140,000s).

<table>
<thead>
<tr>
<th>Property</th>
<th>CPU time [sec]</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>org. [3]</td>
<td>ours (2 steps)</td>
</tr>
<tr>
<td>F(g4267 = 1)</td>
<td>time-out</td>
<td>13742</td>
</tr>
<tr>
<td>F(g4316 = 1)</td>
<td>958</td>
<td>52</td>
</tr>
<tr>
<td>F(g4370 = 1)</td>
<td>time-out</td>
<td>9229</td>
</tr>
<tr>
<td>F(g4371 = 1)</td>
<td>time-out</td>
<td>22485</td>
</tr>
<tr>
<td>F(g4372 = 1)</td>
<td>time-out</td>
<td>14938</td>
</tr>
<tr>
<td>F(g4373 = 1)</td>
<td>time-out</td>
<td>44655</td>
</tr>
</tbody>
</table>

The number of steps of our circuit-expansion is 2. Experiments with our BMTE were run on Xeon X5270 X 2 with 4GB memory, while experiments with BSTE were run on Xeon X5272 X 2 with 16GB memory, where higher spec WS was used for BMTE. From this experimental results, we can see that results of sequential SAT were obtained in 12 of 14 benchmarks with BMTE within time limit, 140,000 seconds.

These experimental results show that the proposed sequential SAT algorithm with BMTE is an improvement in sequential SAT algorithms through efficient state reduction and tracing multiple state transitions at once.

V. Conclusions

We have presented circuit-expansion method for BMTE and proposed sequential SAT algorithm that supports it. Experimental results show that the proposed method may speed up solving process of sequential SAT. Our future work includes sophisticated circuit-expansion method to make sequential SAT solver even faster.

References

