Maximal Models of Assertion Graph in GSTE

Guowu Yang[1], Jin Yang[2], Xiaoyu Song[3], and Fei Xie[1]

[1] Dept. of Computer Science, Portland State University, Portland, OR 97207, USA
[2] Intel Corporation, Strategic CAD Research Labs, Hillsboro, OR 97124, USA
[3] Dept. of Electrical and Computer Engineering,
Portland State University, Portland, OR 97207, USA

Abstract. Generalized symbolic trajectory evaluation (GSTE) is an extension of symbolic trajectory evaluation (STE). In GSTE, assertion graphs are used to specify properties in a special form of regular automata with antecedent and consequent pairs. This paper presents a new model characterization, called maximal models, for an assertion graph with important properties. Besides their own theoretical significance, maximal models are used to show the implication of two assertion graphs in GSTE. We show that, contrary to the general belief, an assertion graph may have more than one maximal model. We present a provable algorithm to find all maximal models of a linear assertion graph. We devise an algorithm for finding a maximal model for an arbitrary assertion graph.

1 Introduction

Generalized symbolic trajectory evaluation (GSTE) [1, 2] is an extension of symbolic trajectory evaluation (STE) [5]. STE can handle large, industrial design and has been actively used in HP, IBM, and Motorola [9, 10, 11, 12]. The STE theory consists of a simple specification language, a simulation-based model checking algorithm, and a mapping of the algorithm to a coarse abstract domain. The specification language of STE has the limited expressiveness where only properties over finite time intervals are allowed. GSTE was originally developed at Intel and has successfully demonstrated its powerful capacity in formal verification of digital systems [1, 2, 3, 4, 13, 14].

In GSTE, all Omega-regular properties can be expressed and verified with the same space efficiency and comparable time efficiency. Assertion graphs are introduced in GSTE as an extension of STE's specification language. Assertion graphs are the specification language in GSTE based on a special form of regular automata with assertion letters (antecedent and consequent pairs) [2]. GSTE specifications are expressed in the form of assertion graphs.

Many RTL designs are rather complicated, primarily because they model complex functional behavior while accommodating tight performance constraints. If we have already proved an assertion graph G_1 against the RTL, a desirable usage is to use G_1 to prove (imply) another assertion graph G_2. Having such an implication mechanism would enable us to achieve higher level abstractions and pursue assume-guarantee prove strategies. There is some work on the implication [3, 4]. In this paper, we present a new concept: maximal model of an assertion graph. Maximal models are used to show the implication of two assertion graphs in GSTE.

This paper is organized as follows. In Section 2, we introduce the basic definitions in GSTE. In Section 3, we introduce some concepts, such as sub-model, maximal model and related properties. In Section 4, we present a provable algorithm to find all maximal models for a linear assertion graph. The application of maximal models in the model-based implication is discussed. In Section 5, we present an algorithm to find a maximal model of an arbitrary assertion graph. We give a condition to determine if a model is a maximal model. In Section 6, we conclude the paper.

2 Preliminaries

We introduce some basic definitions on GSTE [1, 2]. We assume a non-empty set of finite states, denoted by S. A relation $T \subseteq S \times S$ is a transition relation if $\forall s \in S, \exists s' \in S, (s, s') \in T$, where S is a non-empty set of finite states. The model M induced by the transition relation T is the pair $(pre, post)$ where: (1) the pre-image transformer $pre : 2^S \rightarrow 2^S$ is defined as: $pre(Q) = \{s | s' \in Q, (s, s') \in T\}$ for all $Q \in 2^S$; and (2) the post-image transformer $post : 2^S \rightarrow 2^S$ is defined as: $post(Q) = \{s' | s \in Q, (s, s') \in T\}$ for all $Q \in 2^S$.

In fact, a model $M = (pre, post)$ is a directed graph $M = (S, T)$. We use pre and $post$ to represent two functions based on M. Note that $pre(s) = pre(s)$, $post(s) = post(s)$, for all $s \in S$. If for all $s \in S, post(s)$ is defined and nonempty, then M is well-defined. Namely, if we first define $post : S \rightarrow 2^S - \{\emptyset\}$, where $\emptyset$ is an empty set, then a transition relation T can be defined as $T = \{(s, s') | s \in S, s' \in post(s)\}$. A trace in $M = (pre, post)$ is a state sequence such that $\sigma[i+1] \in post(\sigma[i])$, for all $1 \leq i < |\sigma|$, i.e., $(\sigma[i], \sigma[i+1]) \in T$.

An assertion graph is a quintuple $G = (V, v_0, E, ant, cons)$ where V is a finite set of vertices, v_0 is the initial vertex, $E \subseteq V \times V$ is a set of edges, satisfying $\forall u \in V, \exists v \in V$, such that $(u, v) \in E$, ant is a mapping: $E \rightarrow 2^S$, $cons$ is a mapping: $E \rightarrow 2^S$. Let $G = (V, v_0, E, ant, cons)$ be an assertion graph, and let $M = (pre, post)$ be a model. We define an edge labeling γ as : $E \rightarrow 2^S$ where γ is either ant or $cons$. A trace in M satisfies a path ρ of the same length under γ , denoted by $(M, \sigma) \models_\gamma (G, \sigma)$, iff $\sigma[i] \in \gamma(\rho[i]), 1 \leq i \leq |\sigma|$. A trace satisfies a path, denoted by $(M, \sigma) \models (G, \rho)$, iff $[(M, \sigma) \models_{ant} (G, \rho)] \Rightarrow [(M, \sigma) \models_{cons} (G, \rho)]$.

Let $ban(e) = ant(e) - cons(e)$. For a trace σ and a path ρ with length k, if the trace with the first $k-1$ elements of σ ant satisfies the path with the first $k-1$ elements of ρ, then $(M, \sigma) \models (G, \rho)$ if and only if $\sigma[k]$ is not in $ban(\rho[k])$. A model M strongly satisfies an assertion graph G, denoted by $M \models G$ iff $(M, \sigma) \models (G, \rho)$ for all finite initial path ρ in G and all finite trace σ in M of the same length. Given two assertion graphs $G_1 = (V, v_0, E_1, ant_1, cons_1)$ and $G_2 = (U, u_0, E_2, ant_2, cons_2)$, G_1 model-based implies G_2, denoted by $G_1 \Rightarrow_{model} G_2$ (we simply denoted by $G_1 \Rightarrow G_2$), iff $\forall M, M \models G_1 \Rightarrow M \models G_2$.

We impose two restrictions on an assertion graph:

Assumption 1: for all initial edge e (i.e., $start(e) = v_0$), $ban(e) = \emptyset$, i.e., $ant(e) = cons(e)$;

Assumption 2: for all $e, ant(e) \neq \emptyset$.

First, if there is an initial edge e such that $ban(e) \neq \emptyset$, then the one-length trace s ($s \in ban(e)$) does not satisfy the path e, which means no model satisfies this an assertion graph. Second, if $ant(e) = \emptyset$ for some edge e, then all successor edges of e do not affect models.

3 Maximal Model

In this section, we define some concepts such as submodel and maximal model, and give some properties on them.

Definition 1 (Submodel or Contained)

i) Given two models: $M_1 = (S_1, T_1)$, $M_2 = (S_2, T_2)$, where $S_1 \subseteq S, S_2 \subseteq S$, if $S_1 \subseteq S_2$, and $T_1 \subseteq T_2$, then M_1 is called a submodel of M_2, or M_1 is contained by M_2, denoted by $M_1 \leq M_2$.

ii) If $S_1 \subseteq S_2$, and $T_1 \subset T_2$, then M_1 is called a proper submodel of M_2, or M_1 is properly contained by M_2, denoted by $M_1 < M_2$.

Theorem 1. *If $M_1 \leq M_2$, and $M_2 \models G$, then $M_1 \models G$.*

Definition 2 (Maximal Model). *A Maximal-Model of an assertion graph G is a model $M = (S, T) \models G$ and we can not find another model $M_1 = (S, T_1) \models G$ such that $T \subset T_1$. Denoting $M\_max\_G = \{M | M$ is a maximal-model of $G\}$.*

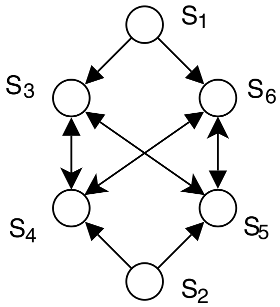

Fig. 1. A model

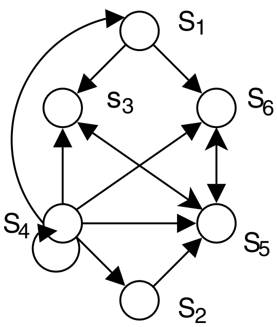

Fig. 1(b). A model

Theorem 2. $(G_1 \Rightarrow G_2) \Leftrightarrow (\forall M, M \in M\_max\_G_1 \Rightarrow M \models G_2)$.

Example 1. Models in Fig.1 and 1(b) are both the maximal models of G in Fig.2.

Theorem 3. *For any given model $M = (S, T)$, there exists an assertion graph G such that M is the unique maximal model of G.*

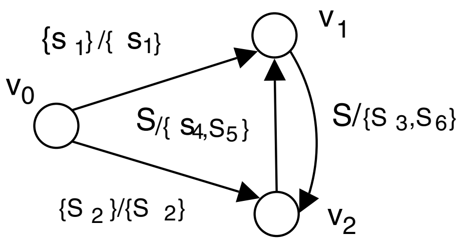

Fig. 2. An assertion graph

The maximal-model of an assertion graph G is usually not unique (see Example 1). Theorem 3 illustrates that only one maximal model of G_1 satisfying G_2 is not enough to derive $G_1 \Rightarrow G_2$ if G_1 has at least two maximal models. From Example 2 in Section 4, we can see how to use the maximal models to determine $G_1 \Rightarrow G_2$.

4 Finding all Maximal Models

In this section, we consider the problem of finding all maximal models of a linear assertion graph G. From Theorem 2, if we find all maximal models of an assertion graph G_1, then we can determine that G_1 model-based implicates G_2. We present the following algorithm: Computing All Maximal Models (CAMM) which can find all maximal models of G.

Definition 3 (Linear assertion graph). $G = (V, v_0, E, ant, cons)$: *Every edge has one and only one successor edge. In the following, without special announcement, if an assertion graph G is a linear assertion graph, we always assume that* $|E| = m, e[m] = (v_{m-1}, v_t), 0 \leq t \leq m-1, e[m+1] = e[t+1]$, *namely, from* $m+1$, *edges have a periodicity* $\tau = m - t$ *(Fig.3).*

Algorithm: CAMM (G)

```
1. $\forall s \in S, P_1(s) = S, Q_1 = S, A_1 = ant(e_1) \cap Q_1 = ant(e_1)$;
2. for $i$ from 1 to $t-1$ do
3.     If $s \in A_i$, then $PP_{i+1}(s) = P_i(s) - ban(e_{i+1})$;
4.     else, $PP_{i+1}(s) = P_i(s)$;
5.     $QQ_{i+1} = \cup_{s \in A_i} PP_{i+1}(s)$;
```

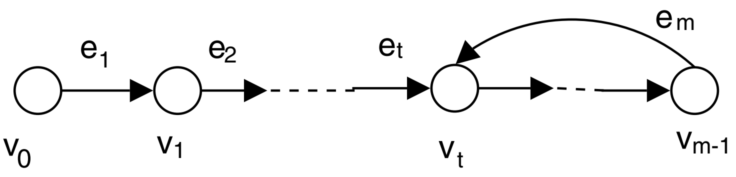

Fig. 3. Linear assertion graph

```
6.    $AA_{i+1} = ant(e_{i+1}) \cap QQ_{i+1}$;
7.    $C_{i+1} = subsetAA_{i+1}$;
8.    If $s \in AA_i$, then $P_{i+1}(s) = P_i(s) - ban(e_{i+1}) - C_{i+1}$;
9.    $Q_{i+1} = \cup_{s \in A_i} P_{i+1}(s)$;
10.   $A_{i+1} = ant(e_{i+1}) \cap Q_{i+1}$;
11. End for;
12. $k = t$;
13. for $j$ from 1 to $\tau - 1$ do
14.    $i = k - 1 + j$;
15.    If $s \in A_i$, then $PP_{i+1}(s) = P_i(s) - ban(e_{i+1})$;
16.    else, $PP_{i+1}(s) = P_i(s)$;
17.    $QQ_{i+1} = \cup_{s \in A_i} PP_{i+1}(s)$;
18.    $AA_{i+1} = ant(e_{i+1}) \cap QQ_{i+1}$;
19.    $C_{i+1} = subsetAA_{i+1}$;
20.    If $s \in AA_i$, then $P_{i+1}(s) = P_i(s) - ban(e_{i+1}) - C_{i+1}$;
21.    $Q_{i+1} = \cup_{s \in A_i} P_{i+1}(s)$;
22.    $A_{i+1} = ant(e_{i+1}) \cap Q_{i+1}$;
23. End for;
24. If $P_{k+\tau}(s) = P_k(s)$ for all $s \in S$ and $Q_{k+\tau} = Q_k$, goto 26;
25. Else $k = k + \tau$ , goto 13;
26. Return $P^*(s) = P_k(s)$ for all $s \in S$.
```

Starting with a trivial model $M_0 : post(s) = S$, for every state s, the algorithm reduces the set of reachable states $P_i(s)$ for each state s edge by edge until it finds a fix-point $P^*(s)$. Let $A_1 = ant(e_1)$ be the set of initial states which are constrained by the second edge. For $s \in A_1$, the set of reachable states $PP_2(s)$ from s is limited by the second edge e_2. Let QQ_1 be the union of $PP_2(s)$ for $s \in A_1$. Let AA_2 be the set of the states that are limited by the 3rd edge. The set of states $ban(e_2)$ are removed from $P_1(s)$. Let C_2 contain the states that are forced to reduce from $PP_2(s)$. C_2 is a subset of AA_2. Let $P_2(s)$ be the final set of reachable states of s after limitation by the 2nd edge. Let A_2 be the final set of states to be limited by the 3rd edge. Repeating the same process, we continue the computation of $P_i(s)$ until no states will be removed from $P_i(s)$. As a result, $P_i(s)$ monotonically decreases to a fix-point $P^*(s)$. We obtain a model M such that $post(s) = P^*(s)$. CAMM is devised to attain the models including all the maximal models. Example 4.1 shows the process.

Let $M(subsetAA_2, subsetAA_3, \ldots, subsetAA_h)$ be an output model produced by algorithm CAMM, where $C_j = \emptyset, j > h$.

Theorem 4. *For any given maximal model M of G, there is a model*

$$M(subsetAA_2, subsetAA_3, \ldots, subsetAA_l) = M.$$

Example 2. Given two assertion graphs G_1 and G_2, with a same directed graph: two vertices: v_0, v_1, and two edges: $e_1 = (v_0, v_1), e_2 = (v_1, v_1)$.

$G_1 : ant_2(e_1) = cons_2(e_1) = \{2\}, ant_2(e_2) = \{2, 3, 4, 5\}, cons_2(e_2) = \{2, 4, 5\}$, $ban_2(e_2) = \{3\}$,

$G_2 : ant_1(e_1) = cons_1(e_1) = \{2\}, ant_1(e_2) = \{3, 4, 5, 6, 7\}$, $cons_1(e_2) = \{5, 6\}, ban_1(e_2) = \{3, 4, 7\}$.

Using CAMM, ($S = \{1, 2, 3, 4, 5, 6, 7\}$), we have four models:

$M_1 : P(2) = P(5) = P(6) = \{1, 2, 5, 6\}, P(1) = P(3) = P(4) = P(7) = S$.

$M_2 : P(2) = P(5) = \{1, 2, 5\}, P(1) = P(3) = P(4) = P(6) = P(7) = S$.

$M_3 : P(2) = P(6) = \{1, 2, 6\}, P(1) = P(3) = P(4) = P(5) = P(7) = S$.

$M_4 : P(2) = \{1, 2\}, P(1) = P(3) = P(4) = P(5) = P(6) = P(7) = S$.

Using SMC in [1, 2], we know that $M_i \models G_2$ for $i = 1, 2, 3, 4$. Therefore, $G_1 \Rightarrow G_2$. In fact, these four models are maximal models of G_1 according to Theorem 8. For the model-based implication, we know in [15] if two assertion graphs G_1 and G_2 have the same graph structure and $(ant_2(e) \subseteq ant_1(e)) \wedge (cons_1(e) \cap ant_2(e) \subseteq cons_2(e))$, for all $e \in E$, namely, $(ant_1(e) \supseteq ant_2(e)) \wedge (ban_1(e) \supseteq ban_2(e))$, for all $e \in E$, then we have $G_1 \Rightarrow G_2$. In example 2, $ant_1(e) \supseteq ant_2(e)$ is not true, but $G_1 \Rightarrow G_2$, which means this sufficient condition for model-based implication is not necessary. For linear assertion graphs, [15] gave the sufficient and necessary conditions for language-based implication. But for model-based implication, the problem is more complicated. Example 2 shows that these conditions are not either sufficient or necessary for model-based implication.

5 Finding a Maximal Model of an Arbitrary Assertion Graph

Let $start(e)$ and $end(e)$ denote the start and end vertices of a directed edge e, respectively. Let $start(v)$ and $end(v)$ denote the directed edges in an assertion graph G with the starting vertex v and the ending vertex v, respectively. We define the following sets for an assertion graph $G = (V, v_0, E, ant, cons)$:

$V_0 = \{v_0\}, E_1 = \{e | start(e) = v_0\}$,

$V_1 = \{v | e \in E_1, end(e) = v\}$,

$E_i = \{e | start(e) \in V_{i-1}\}, V_i = \{v | e \in E_i, end(e) = v\}$, for $i = 2, 3, \ldots$.

Lemma 1. *There exist t and $\tau > 0$ such that $V_t = V_{t+\tau}$.*

Let t and τ be the minimum numbers satisfying $V_t = V_{t+\tau}$. We present an algorithm, Computing Satisfied Model (CSM), to find a maximal model of an arbitrary assertion graph. In the algorithm, we compute the set of reachable states $P_i(s)$ from state s after the restriction of the ith step for all $s \in S$.

The basic idea of the algorithm CSM is described as follows. We initialize $P_1(s) = post\_M(s)$, for all $s \in S$. Initially, the set of reachable states from any state s is the post function of s in an input model M. Along the path of the assertion graph from the initial edges E_1, we reduce the set of the states reachable from s. $A_1(v) = \cup_{e \in E_1} ant(e), v \in V_1$, is the initial states which will be constrained by the edges E_2. For $s \in A_1(v)$, the reachable states $P_2(s)$ from

s will be limited by the edges of $start(v)$. The states of $ban(v)$ will be removed from $P_1(s)$. $P_i(s)$ is the set of the i^{th} step reachable states from s via E_i. $A_i(v), v \in V_i$, is the set of states which will be limited by the $(i+1)^{th}$ step edges E_{i+1}. For $s \in A_i(v), ban(v)$ will be removed from $P_i(s)$. We continue our computation $P_i(s)$ until no states are removable from $P_i(s)$. As a result, $P_i(s)$ monotonically decreases to a fix-point $P^*(s)$. Thus, we obtain a model $M\_r, post\_M\_r(s) = P^*(s)$.

Algorithm: CSM(M,G)

```
1. $\forall s \in S, P_1(s) = post\_M(s), Q_1(v_0) = S$,
2. For $i$ from 1 to $t-1$ do
3.   for $v \in V_i$,
4.     $A_i(v) = \cup_{e \in E_i \cap end(v)}[ant(e) \cap Q_i(start(e))]$;
5.     $ban(v) = \cup_{e \in start(v)} ban(e)$;
6.     If $s \in A_i(v)$, then $P_{i+1}(s) = P_i(s) - ban(v)$;
7.     Else, $P_{i+1}(s) = P_i(s)$;
8.     $Q_{i+1}(v) = \cup_{s \in A_i(v)} P_{i+1}(s)$;
9. End For.
10. $k = t$;
11. For $j$ from 1 to $\tau - 1$ do
12.   $i = k - 1 + j$;
13.   for $v \in V_i$,
14.     $A_i(v) = \cup_{e \in E_i \cap end(v)}[ant(e) \cap Q_i(start(e))]$;
15.     $ban(v) = \cup_{e \in start(v)} ban(e)$;
16.     If $s \in A_i(v)$, then $P_{i+1}(s) = P_i(s) - ban(v)$;
17.     Else, $P_{i+1}(s) = P_i(s)$;
18.     $Q_{i+1}(v) = \cup_{s \in A_i(v)} P_{i+1}(s)$;
19. End For.
20. If $P_{k+\tau}(s) = P_k(s)$ for all $s \in S$ and $Q_{k+\tau}(v) = Q_k(v)$, for $v \in V_k$, goto 22;
21. Else $k = k + \tau$; goto 11;
22. Return a model $M\_r$, where $post\_M\_r(s) = P^*(s) = P_k(s)$ for all $s \in S$.
```

Lemma 2. *The algorithm CSM stops in a finite number of steps.*

Theorem 5. $M\_r \models G$.

We use CSM to find a maximal model of G. We start from a trivial model $M_0 : post\_M_0(s) = S$ for all $s \in S$. Using CSM, we get a satisfying model $M\_r = CSM(M_0)$. But in some cases, M_r may not be a maximal model. For instance, when we calculate $A_2(v)$ to do the third edge's limitation, $A_2(v) = \cup_{e \in E_2 \cap end(v)}[ant(e) \cap \cup_{s \in A_1(start(e))} P_2(s)]$. Because $P_2(s) \supseteq P^*(s)$, it is possible that $A_2(v) \supset \cup_{e \in E_2 \cap end(v)}[ant(e) \cap \cup_{s \in A_1(start(e))} P_2(s)]$. As a result, there are more states which are taken off from the set of reachable states set during the third step. To avoid this, we have to start from a refined model M which is smaller than M_0 but no more than a maximal model. The following algorithm, called Induced Model (IM), is used to find such an initial model.

Algorithm: IM(M,G)

```
1. $\forall s \in S, P_1(s) = S, R_1(v_0) = S$,
2. For $i$ from 1 to $t-1$ do
3.    for $v \in V_i$,
4.       $B_i(v) = \cup_{e \in E_i \cap end(v)}[ant(e) \cap R_i(start(e))]$;
5.       $ban(v) = \cup_{e \in start(v)} ban(e)$;
6.       If $s \in B_i(v)$, then $P_{i+1}(s) = P_i(s) - ban(v)$;
7.       Else, $P_{i+1}(s) = P_i(s)$;
8.       $R_{i+1}(v) = \cup_{s \in B_i(v)} post\_M(s)$;
9. End For.
10. $k = t$;
11. For $j$ from 1 to $\tau - 1$ do
12.    $i = k - 1 + j$;
13.    for $v \in V_i$,
14.       $B_i(v) = \cup_{e \in E_i \cap end(v)}[ant(e) \cap R_i(start(e))]$;
15.       $ban(v) = \cup_{e \in start(v)} ban(e)$;
16.       If $s \in B_i(v)$, then $P_{i+1}(s) = P_i(s) - ban(v)$;
17.       Else, $P_{i+1}(s) = P_i(s)$;
18.       $R_{i+1}(v) = \cup_{s \in A_i(v)} post\_M(s)$;
19. End For.
20. If $R_{k+\tau}(v) = R_k(v)$, for $v \in V_k$, goto 22;
21. Else $k = k + \tau$; goto 11;
22. Return a model $M\_r$, where $post\_M\_r(s) = P^*(s) = P_k(s)$ for all $s \in S$.
```

Theorem 6. *If $M \models G$, then the output model in IM $IM(M,G) \geq M$.*

Theorem 7. *If $M_1 \geq M_2$, and $M_1 \models G$, then $IM(M_2,G) \geq IM(M_1,G) \geq M_1 \geq M_2$.*

Theorem 8. *If $M \models G$ and $IM(M,G) = M$, then M is a maximal model of G.*

Algorithm: RCSM(G)

```
1. $Flag1 = Flag2 = 0$; $k = 1$;
2. $M\_r[1] = CSM(M_0, G)$;
3. While $Flag1 = 0$ do
4.    $M_k = IM(M\_r[k], G)$;
5.    $M\_r[k+1] = CRM(M_k, G)$;
6.    If $M\_r[k+1] > M\_r[k]$, then $k = k + 1$;
7.    Else $Flag1 = 1$; If $M_k = M\_r[k]$, then $Flag2 = 1$;
8. Return: $M\_rr = M\_r[k], Flag2$.
```

Theorem 9. *(1) Algorithm RCSM will stop in finite steps.*
(2) $M\_rr \models G$.
(3) If $Flag2 = 1$, then M_rr is a maximal model of G.

It is still open if $M\_r[k]$ monotonically increases. Therefore we cannot guarantee $M_r r$ be a maximal model of G. But, anyway, even M_rr is not a maximal

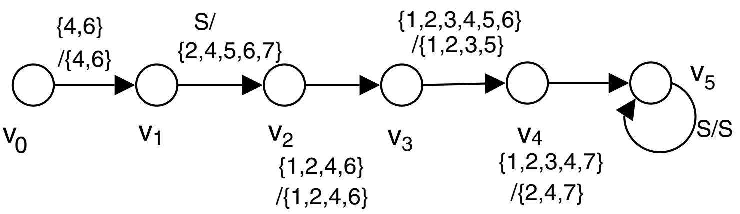

Fig. 4. An assertion graph

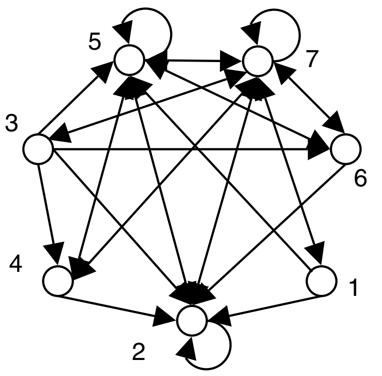

Fig. 5. Model $M\_r[1]$

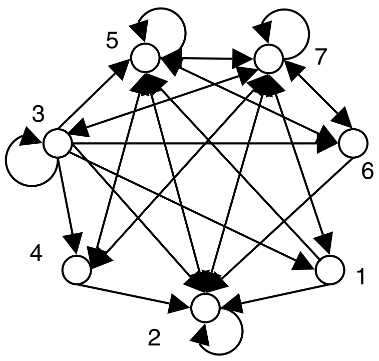

Fig.5(b). Model $M\_rr = M\_r[2]$

model, there is a maximal model M between M_rr and $IM(M\_rr, G)$ according to Theorem 7. The following example shows an application of RCSM to find a maximal model of an assertion graph G.

Example 3. Given an assertion graph G as Fig.4.

According to the algorithm RCRM, we get $M\_r[1]$ (Fig.5) as:

$Post(1) = \{2,5,7\}, Post(2) = \{2,5,7\}, Post(3) = \{2,4,5,6,7\}$, $Post(4) = \{2,5,7\}, Post(5) = \{2,4,5,6,7\}, Post(6) = \{2,5,7\}, Post(7) = \{1,2,3,4,5,6,7\}$.

$M\_r[2]$ (Fig.5(b)) is: $Post(1) = \{2,5,7\}, Post(2) = \{2,5,7\}$, $Post(3) = \{1,2,3,4,5,6,7\}, Post(4) = \{2,5,7\}, Post(5) = \{2,4,5,6,7\}, Post(6) = \{2,5,7\}$, $Post(7) = \{1,2,3,4,5,6,7\}$. $M\_r[3] = M\_[2]$, so stop. $M_2 = IM(M\_r[2], G) = M\_r[2]$, so $Flag2 = 1$, thus $M\_rr = M\_r[2]$ is a maximal model of G (Theorem 5.4). $M\_r[2] > M\_r[1]$ (There are two more edges: (3,3), (3,1) in $M\_r[2]$ than in $M\_r[1]$).

6 Conclusions

We presented a new model characterization called maximal models for an assertion graph with important properties. We showed that an assertion graph may have more than one maximal model. We presented a provable algorithm to find all maximal models of a linear assertion graph. We devised an algorithm for finding a maximal model for an arbitrary assertion graph.

References

[1] Yang, J. and Seger, C.: Generalized Symbolic Trajectory Evaluation. Technical Report, 2002, Intel.

[2] Yang, J. and Seger, C.: Introduction to Generalized Symbolic Trajectory Evaluation. IEEE Trans on VLSI Systems, 11(3) (2003), 345-353

[3] Hu, A. J., Casas, J., and Yang, J.: Efficient Generation of Monitor Circuits for GSTE Assertion Graphs. IEEE/ACM International Conference on Computer-Aided Design (2003), 154-159

[4] Hu, A. J., Casas, J., and Yang, J.: Reasoning about GSTE Assertion Graphs. 12th Ad-vanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME) (2003), 170-184

[5] Seger, C. and Bryant, R. E.: Formal verification by symbolic evaluation of partially-ordered trajectories. Formal Methods in System Design, 6(2) (1995): 147-190

[6] Yang, G. , Yang, J. and Song, X.: Formal Verification by Generalized Symbolic Trajectory Evaluation. Technical Report, Portland State University, 2004

[7] Chou, C.-T.: The mathematical foundation of synmbolic trajectory evaluation, in Proc. Computer-aided Verifucation(CAV) (1999), 196-207

[8] Bentley, B.: High level validation of next generation microprocessors. Proc. IEEE High Level Design Validation and Test Workshop (HLDVT) (2002), 31-35

[9] Aagaard, M., Jones, R. B., and Seger, C.: Combining thereom proving and trajectoy evaluaton in an industrial environment, in 35th Design Automation Conferece, ACM/IEEE (1998), 538-541

[10] Bjesse, P., Leonard, T., and Mokkedem, A.: Finding bugs in an Alpha microprocessor using satisfiability solvers, in Computer-Aided Verification: 13th International Conferece, pages 454-464

[11] Kyle L. Nelson Alok Jain, and Bryant, R. E.: Formal verification of a superscalar execution unit, in 34th Design Automation Conferece, ACM/IEEE (1997), 161-167

[12] Pandey, M., Raimi, R., Beatty, D. L., and Bryant, R. E.: Formal verification of PowerPC arrays using symbolic trajectory evluation, in 33rd Design Automation Confer-ence, ACM/IEEE (1996), 649-654

[13] Edmund M. Clarke and E. Allen Emerson: Design and synthesis of synchronization skele-tons using branching time temporal logic, in Dexter Kozen, editor, Workshop, on Logics of Programs (May 1981), 52-71

[14] Kurshan, R. P.: Computer-Aided Verification of Coordinating Processes: The automata-Theorotic Approach, Princeton University Press, 1994

[15] Yang, G., Yang, J. Hung, W.N.N, and Song, X.: Implication of assertion graphs in GSTE, ASPDAC (2005), 1060-1063