A Polynomial Algorithm to Decide a Live SMA Nets

[1]Dong-Ik Lee, [2]Sadatoshi Kumagai, and [3]Jin-Ho Bae

1 Dept of Elec. Eng., Osaka Univ., Japan
2 Computation Center, Osaka Univ., Japan
3 Dept. of Ele. Eng., Yeong-Nam Univ., Korea

I Introduction

In the modelling and design of a large number of concurrent systems, live and safe free choice nets (LSFC nets) have been explored in their structural characteristics [3][4][6][7]. On the other hand, state machine decomposable nets (SMD nets) are a class of Petri nets which can be decomposed by a set of strongly connected state machines (S-decomposition). State machine allocatable nets (SMA nets) are a subclass of SMD nets, for which the well-behavedness such as liveness and safeness of state machine components is preserved in the composed net. Of particular interest is the relation between LSFC nets and SMA nets such that a free choice net has a live and safe marking if and only if the net is an SMA net. That is, the structure of an LSFC net is an SMA net [1]. Recently, the complete characterization of SMA net structure has been obtained by the authors based on an S-decomposition [6][7]. In Ref. [6], a necessary and sufficient condition for a net to be an SMA net is obtained in terms of the net structure where synchronization between strongly connected state machine components (S-components) has been clarified. Unfortunately, it requires tremendous amount of time and spaces to decide a given net to be an SMA net or not by applying those conditions directly. Moreover, there exist no efficient algorithm to decide the liveness of a given SMA net that lessens the usefulness of the decomposition techniques.

The aim of this paper is to propose an efficient polynomial order algorithm to decide whether a given net is a live SMA net or not. The problem can be divided into two sub-problems ; (1) to decide a given net is an SMA net or not, (2) to decide a given SMA net is live or not. In Section 3, we present a polynomial order algorithm for problem (1). And in Section 4, efficient algorithms for problem (2) is presented. The algorithms proposed here are based on the net decomposition techniques adopted in Ref. [13]. In the next section, basic terminologies and definitions are given. Section 5 is the conclusion.

2. Basic terminologies and definitions

We assume that the readers are familiar with Petri net theory. Formal definitions of Petri nets, firing rule, state machine, marked graph, and free choice net are omitted here and is found in Ref. [6].

Definition 1. Let $(N,M_0)=(P,T;F,M_0)$ be a marked net and $N_1=(P_1,T_1;F_1)$ be a subnet of N.

(a) N_1 is called a *T-component* of (N,M_0) if N_1 is a strongly connected marked graph and T_1-closed, i.e., $P_1={}^\bullet T_1 \cup T_1{}^\bullet$, $F_1= F\cap((P_1\times T_1)\cup(T_1\times P_1))$.

(b) N_1 is called an *S-component* of (N,M_0) if N_1 is an SCSM (strongly connected state machine) and P_1-closed, i.e., $T_1={}^\bullet P_1\cup P_1{}^\bullet$, $F_1=F\cap((P_1\times T_1)\cup(T_1\times P_1))$.

Definition 2. A net N=(P,T;F) is an SMD net (state machine decomposable net) if there exists a set of S-components $N_i=(P_i,T_i;F_i)$ such that $P=\cup P_i$, $T=\cup T_i$, $F=\cup F_i$. In this case we say N is *covered* by S-components. A minimal set of S-components (T-components) which covers N is called an *S-decomposition* (*T-decomposition*) of N. A transition which belongs to at least two S-components in an S-decomposition is called a *common transition*.

Definition 3. Let S_D is a set of S-components of a net N=(P,T;F). For any place $p\in P$, the number of S-components in S_D containing p is the weight of p.

Definition 4. An *SM-allocation* over a Petri net N=(P,T;F) is a mapping $B:T\rightarrow P$ such that $\forall t\in T$, $B(t)\in {}^\bullet t$.
For a given SM-allocation B, the reduction of a Petri net can be defined by the following procedures. And the resulting net is called an *SM-reduced net.*

Algorithm 1. Find an SM-reduced net
Delete all unallocated places.
Repeat{
  Delete every transition, whose output places are all deleted.
  Delete every place, at least one of whose output transitions is deleted.
  Delete all arcs incident with deleted nodes.
} until no more deletions applicable.

Computational complexity of algorithm 1 is $O(m^2n)$, where m and n represent the number of transitions and the number of places in N, i.e., $|T|=m$, $|P|=n$, respectively.

N=(P,T;F) is called an *SMA net (state machine allocatable net)* if every SM-reduced net of N is a non-empty set of SCSMs.

Definition 5. Let $N_1=(P_1,T_1;F_1)$ be a subnet of N=(P,T;F). An elementary directed path π, $\pi=x_1\rightarrow\cdots\rightarrow x_n$, $x_i\in P\cup T$, is a *handle* of N_1 iff $\pi\cap N_1=\{x_1, x_n\}$ (possibly $x_1=x_n$). If both x_1 and x_n are places (transitions), π is called a pp-handle (tt-handle). If x_1 is a place (transition) and x_n is a transition (place), π is called a pt-handle (tp-handle).

Definition 6. For two elementary directed paths l_1 and l_2, $l_1:p_1\rightarrow p_n$, $l_2:p_1\rightarrow p_n$ (possibly $p_1=p_n$), in an S-component S_i, if $l_1\cap l_2=\{p_1,p_n\}$ and there exists l_3, $l_3:p_n\rightarrow p_i$, such that $l_3\cap(l_1\cup l_2)=\{p_1,p_n\}$, then l_1 and l_2 are called parallel paths in S_i with respect to the initial place p_1 and the terminal place p_n, and l_3 is called a return path of parallel paths l_1 and l_2. Let l_i' be a subpath of l_i, i=1, 2, 3. If there exists a pp-handle π of l_1', $l_1'\neq l_1$, such that $(\pi\cap(l_2\cup l_3))-\{p_1,p_n\}=\phi$, then $\pi\cup l_1$ and l_2 are in parallel. That is, $l_1\cup\pi$ is seen as a member of parallel paths with respect to p_1 and p_n. Similarly, if there exists a pp-handle π of l_3', $l_3'\neq l_3$, such that $(\pi\cap(l_1\cup l_2))-\{p_1,p_n\}=\phi$, then $\pi\cup l_3$ is also a return path. Any transitions t_i and t_j on parallel paths l_1 and l_2, respectively, are called *parallel transitions* in S_i with respect to p_1 and p_n.

Definition 7. A *deadlock* is a non-empty subset of places $D\subset P$ such that ${}^\bullet D\subset D^\bullet$. A deadlock D, any proper subset of which is not a deadlock, is called a *minimal deadlock.*

3. A polynomial order algorithm to decide SMA nets

In this section we consider a polynomial order algorithm to decide whether a given net is an SMA net or not based on the SM-decomposability of SMA nets. First of all, we show the decomposition property of SMA nets which plays an importtant role to construct the algorithm. The property can be directly derived from the decomposability of LSFC nets and the relationship between an LSFC net and an SMA net given in [1].

Lemma 1. Let N=(P,T;F) be an SMA net, and x be an arbitrary element of $P\cup T$.
(1) There exists a T-component $N_1=(P_1,T_1;F_1)$ of N such that $x\in P_1\cup T_1$.
(2) There exists an S-component $N_2=(P_2,T_2;F_2)$ of N such that $x\in P_2\cup T_2$.♦

Lemma 1 implies the existence of a T-decomposition and an S-decomposition of an SMA net. And the structure of

SMA (LSFC) nets was completely characterized by the authors based on an S-decomposition [7]. To simplify our discussion, we define a new terminology.

Definition 8. Let $S_i=(P_i,T_i;F_i)$ be an S-component of a net N. For any set L of parallel paths and any return path l_r of L in S_i, a tt-handle $\pi:t_i \rightarrow t_j$ of S_i such that $t_i \in l_i \cup l_r$, $t_j \in l_j$, $\{l_i, l_j\} \subset L$, is called a *bad handle* of S_i.

Theorem 1. [7] Let N=(P,T;F) be an SMD net. N is an SMA net if and only if there exists an S-decomposition S_D of N such that for every S-component of S_D there exist no bad handles. Conversely, if N is an SMA net, then for any S-component of N, there exist no bad handle.♦

Theorem 1 is useful for modular synthesis and analysis of SMA nets. To examine the condition, all the sets of parallel paths and the return paths of an S-component have to be found. However, it is very difficult and takes too much time. Thus, we consider to examine the condition without finding all the set of parallel paths and the return paths of an S-component. At first, we define two binary relations, called permissible relation and forbidden relation, defined over the Cartesian product $T_i \times T_i$, where T_i is the set of transitions in an S-component S_i of an SMD net.

Definition 9. Let $S_i=(P_i,T_i;F_i)$ be an S-component of an SMD net N=(P,T;F), and R be the Cartesian product $T_i \times T_i$. $(t_a,t_b) \in R$ is said *permissible* in S_i, represented by $t_a \mathcal{P} t_b$, if $t_a=t_b$ or there does not exist an elementary directed circuit which contains t_a but does not contain t_b, in S_i. Conversely, if there exists an elementary directed circuit which contains t_a but does not contain t_b, in S_i, then $(t_a,t_b) \in R$ is said *forbidden* in S_i, and represented by $t_a \mathfrak{F} t_b$. Moreover, for a tt-handle $\pi:t_i \rightarrow t_j$ of S_i, if $t_i \mathcal{P} t_j$, then π is called *permissible handle* of S_i. Similarly, if $t_i \mathfrak{F} t_j$, then π is called *forbidden handle* of S_i.

Let the net in Fig. 1 be an S-component of an SMD net. There exist no directed circuits which contains t_1 but not t_2, thus $t_1 \mathcal{P} t_2$. On the other hand, for (t_5,t_6), there exists a directed circuit $C=p_1 \rightarrow t_5 \rightarrow p_5 \rightarrow t_7 \rightarrow p_4 \rightarrow t_4 \rightarrow p_1$ which contains t_5 but not t_6, then $t_5 \mathfrak{F} t_6$.

In fact, a forbidden handle of an S-component is equivalent to a bad handle of the S-component. Here, we establish the equivalence.

Theorem 2. Let S_i be an S-component of an SMD net. The followings are equivalent. (1) There exists a bad handle of S_i. (2) There exists a forbidden handle of S_i.♦

Now, the original problem can be divided into two sub-problems, as follows.

SP1. Find an S-decomposition of the given net.

SP2. Decide the existence of forbidden handles of an S-component.

For SP1, a polynomial order algorithm is already obtained by the authors [9][10]. The algorithm is based on an SM-allocation and the well known depth-first search algorithm [11].

Algorithm 2. Find an S-decomposition of a net N=(P,T;F, M_0**).**

Set $S_D=\phi$ and weight of every place in P to be zero.
Repeat{ /Find a set of S-components covering N/
For (there exists a place whose weight is zero)
  Select a place p_i whose weight is zero.
  Generate a depth-first search tree, tree-i, by selecting p_i as the root.
  For any transition in T, allocate the input place on tree-i.
  Do SM-reduction in Algorithm 1, according to the allocation.
  Let S_i be the SM-reduced net.
  If S_i is an SCSM, then
    $S_D:=S_D \cup \{S_i\}$ and add 1 to weight of every place in S_i.
  Else, then
    goto End 2
  }
Repeat { /guarantee minimality of an S-decomposition/
For (there exists an S-component, in S_D, whose each place is weighted more than 1)
  Find an S-component $S_k=(P_k,T_k;F_k)$ whose each place is weighted more than 1.
  $S_D:=S_D-\{S_k\}$.
  Subtract 1 from weight of every place in P_k.
  }
End 1 Output "S_D" and end.
End 2 Output "N can not be an SMA net" and end.

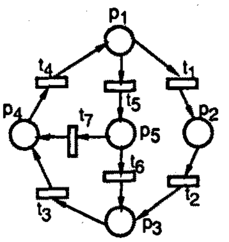

Figure 1 An S-component

TABLE 1 An Rtable of the net in Fig. 1

t_a \ t_b	1	2	3	4	5	6	7
1	O	O	O	O	X	X	X
2	O	O	O	O	X	X	X
3	X	X	O	O	X	X	X
4	X	X	X	O	X	X	X
5	X	X	X	O	O	X	X
6	X	X	O	O	O	O	X
7	X	X	X	O	O	X	O

The computational complexity of Algorithm 2 is $O(m^2n^2)$. The validity of Algorithm 2 is guaranteed by the following lemma.

Lemma 2. [9] For an SMA net N=(P,T;F), the set of S-components obtained by Algorithm 2 is an S-decomposition of N.♦

If an output of Algorithm 2 is a non-empty set of SCSMs, then N is an SMD net and may be an SMA net. For SP2, we consider a polynomial order algorithm to decide the relationship, permissible or forbidden, between each ordered pairs of transitions in an S-component. The output of the algorithm is a table, denoted by *Rtable*, which shows an ordered pair of transitions is either in the permissible or the forbidden relation. The procedure is described as follows.

Algorithm 3. Generate an Rtable of an S-component $S_i=(P_i,T_i;F_i)$**.**

Generate a $|T_i| \times |T_i|$ table RT_i, and mark each element with O.
Set $\tau_i=T_i$
Return {
For ($\tau_i \neq \phi$)
  Select a transition $t_b \in \tau_i$.
  Delete t_b and the incident arcs from S_i.
  Divide the reduced net of S_i into strongly connected components.
  Let T_{ia} be the set of transitions such that any transition in T_{ia} is isolated node in the reduced net.
  $\forall t_a \in T_{ia}$, (t_a,t_b) or (t_b,t_b) is permissible.
  Let $T_{ia}'=T_i-T_{ia}$.
  $\forall t_a' \in T_{ia}'$, (t_a',t_b) is forbidden.
  Change the mark of (t_a',t_b) element of RT_i into X
  $\tau_i:=\tau_i-t_b$
}

The computational complexity of Algorithm 3 is $O(m^2)$. We claim that, in an Rtable, O and X marks correctly an ordered pair in the permissible and the forbidden relations, respectively. For the validity of the algorithm, we have the following lemma.

Lemma 3. For an S-component of a net N, the output of Algorithm 3 agrees with the definitions of the permissible and the forbidden relations. ♦

For the net in Fig. 1, the output of Algorithm 3 is summarized in Table 1.

Finally, it remains to construct a polynomial order algorithm to decide the existence of forbidden handles of an S-component. We consider an algorithm to generate a table, denoted by *Htable*, which exhibits the existence of tt-handles of an S-component between ordered pair of transitions. The algorithm is based on a simple depth-first search algorithm.

Algorithm 4. Generate an Htable of an S-component $S_i=(P_i,T_i;F_i)$ **of N=(P,T;F).**

Delete all places in P_i and all the incident arcs from N.
Let N' be the reduced net.
Generate a $|T_i| \times |T_i|$ table HT_i, and mark each element with X.
Set $\tau_i=T_i$.

Return {
For ($\tau_i \neq \phi$)
Select a transition t_c in τ_i.
Do depth-first search from t_c.
If there exists a transition $t_d \in T_i$ during depth-first search, then change the mark of (t_c,t_d) element of HT_i to O and continue depth-first search.
$\tau_i := \tau_i - t_b$.
}

The validity of the algorithm is obvious, and the computational complexity of Algorithm 4 is $O(m^2n)$. If (t_i,t_j) element of the Htable of S_i is marked by O, there is a tt-handle $\pi: t_i \rightarrow t_j$ of S_i.

The existence of forbidden handles of an S-component S_i can be decided by comparison of Rtable with Htable of S_i. If (t_i,t_j) element is marked by X in Rtable and is marked by O in Htable, then there exists a forbidden handle from t_i to t_j of S_i. Otherwise, there exist no forbidden handle of S_i from t_i to t_j.

Above three algorithms can be summarized in the following algorithm.

Algorithm 5. Decide a given net N=(P,T;F) is an SMA net or not.

Find an S-decomposition S_D by Algorithm 2.
If there does not exist S_D, then
goto End 2.
Else, then
Repeat{
For (each S_i in S_D)
Generate the Rtable of S_i by Algorithm 3.
Generate the Htable of S_i by Algorithm 4.
Test by comparison the Rtable and the Htable.
If (t_k,t_l) element of the Rtable is X and that of the Htable is O, then
goto End 2.
}
End 1 Output "N is an SMA net", and end.
End 2 Output "N is not an SMA net", and end.

The computational complexity of the algorithm is $O(m^2n^2)$.

4. Liveness of an SMA nets

In this section, we consider a polynomial order algorithm to decide the liveness of an SMA net.

Lemma 4. [2][8] Let (N,M_0) be an SMA net. (N,M_0) is live if and only if any S-component in N has at least one token.♦

From Lemma 4, liveness of an SMA net (N,M_0) can be decided by finding all S-components of N and examining the existence of tokens in each S-component. Unfortunately, the computational complexity to find all S-components of N is $O(m^n)$. However, from Lemma 4 and the fact that the set of places in an S-component is a minimal deadlock of N, we can derive another necessary and sufficient condition for the liveness of SMA nets described as follows.

Theorem 3. An SMA net (N,M_0) is live if and only if there exists no token-free deadlocks.♦

Now, the problem is reduced to find a set of token-free deadlocks of a Petri net (N,M_0), which can be solved by the following algorithm.

Algorithm 6. Decide the existence of a token-free deadlock of a net (N,M_0)

Delete all marked places and the incident arcs.
Repeat { /Find a token-free deadlock of (N,M_0)/
For (the reduced net is to be empty)
Divide the reduced net into strongly connected components.
If there exists a strongly connected component in which all input transitions of the place are include, then
goto End 2.
Else, then
delete all the places and the incident arcs.
Delete all isolated transitions.
}
End 1 Output "there is no token-free deadlock" and end.
End 2 Output "there is a token-free deadlock", and end.

The computation complexity of Algorithm 6 is $O(mn^2)$. The termination of Algorithm 6 is obvious since N can be divided into at most n pieces of strongly connected components.Here, we show the correctness of Algorithm 6.

Theorem 4. For a Petri net (N,M_0), there does not exist any token-free deadlock if and only if Algorithm 6 terminates at End 1.♦

From Theorem 3 and 4, the liveness of SMA nets can be decided by Algorithm 6. If a given net (N,M_0) is decided to be an SMA net by Algorithm 5, and to be live by Algorithm 6, then (N,M_0) is a live and bounded. Now, we can decide whether a given net is a live and safe SMA net by the following algorithm. Thus, we can easily construct $O(m^2n^2)$ algorithm to decide whether a given net is a live SMA net or not, from Algorithm 5 and 6 as following.

Algorithm 7. Decide a given net (N,M_0) is a live and safe SMA net or not

Decide N is an SMA net or not by Algorithm 5.
If N is not an SMA net, then
goto End .
Else, then
decide (N,M_0) is live or not by Algorithm 6.
If (N,M_0) is not live, then
goto End.
Else, then
decide (N,M_0) is safe or not by Algorithm 8.
If (N,M_0) is not safe, then
goto End.
Else, then
output "(N,M_0) is a live and safe SMA net" and end.
End Output "(N,M_0) is not a live and safe SMA net" and end.

5. Conclusion

$O(m^2n^2)$ order algorithm to decide a live SMA net has been obtained. The algorithms obtained in the paper is useful to verification of concurrent systems.

REFERENCES

[1] M. Hack, "Analysis of Production Schemata by Petri Nets," M. S. thesis, TR-94, Project MAC, MIT, 1972.
[2] M. Hack, "Extended State-Machine Allocation Nets (ESMA), an extension of Free Choice Petri Net Results," Computation group memo 78-1, Project MAC, MIT, 1974.
[3] K. Hiraishi and A. Ichikawa, "A Class of Petri Nets That a Necessary Sufficient Condition for Reachability is Obtainable," Trans. of SICE Vol. 24, No. 6, June 1988, pp.635-640.
[4] T. Murata, "Circuit Theoretic Analysis and Synthesis of Marked Graphs," IEEE Trans. circuits and systems, Vol. CAS-24, No. 7, July 1977, pp. 400-405.
[5] P. S. Thiagarajan and K. Voss, "A Fresh Look At Free Choice Nets," Information and Control 62, pp.85-113, 1984.
[6] D. I. Lee, S. Kumagai and S. Kodama, "Complete Structural Characterization of State Machine Allocatable Nets," IEICE Trans., Vol. E 74, No. 10, 1991.
[7] D. I. Lee, S. Kumagai and S. Kodama, "A Basic Theorem for Modular Synthesis of State Machine Allocatable Nets", Proceedings of ISCAS'92, Vol. 5, pp2390-2393, 1992.
[8] M. Jantzen and R. Valk, "Formal Properties of Place/Transition Nets," Lecture Notes in Computer Science 84, pp166-pp210, Springer-Verlag, 1980.
[9] T. Nishimura, D. I. Lee, S. Kumagai and S. Kodama, "A decomposition Algorithm for LSFC nets," Proceedings of 7th SICE Symposium on Discrete Event Systems (in Japanese), 1991.
[10] T. Nishimura, D. I. Lee, S. Kumagai, and S. Kodama, "A Decomposition Algorithm for Live and Safe Free Choice Nets", Submitted to IEICE. (in Japanese)
[11] M. Iri, I. Shirakawa, Y. Kajitani, and S. Shinoda, "Graph Theory with Exercise", Corona Publishing, 1983. (in Japanese)