LETTER *Special Section on Concurrent Systems Technology*

An Algorithm for Legal Firing Sequence Problem of Petri Nets Based on Partial Order Method

Kunihiko HIRAISHI[†a)], ***Regular Member and*** **Hirohide TANAKA**[†b)], ***Nonmember***

SUMMARY The legal firing sequence problem of Petri nets (LFS) is one of fundamental problems in the analysis of Petri nets, because it appears as a subproblem of various basic problems. Since LFS is shown to be NP-hard, various heuristics has been proposed to solve the problem of practical size in a reasonable time. In this paper, we propose a new algorithm for this problem. It is based on the partial order verification technique, and reduces redundant branches in the search tree. Moreover, the proposed algorithm can be combined with various types of heuristics.

key words: *Petri nets, legal firing sequence problem, partial order methods, stubborn sets, state space explosion*

1. Introduction

The legal firing sequence problem of Petri nets (LFS, for short) is one of fundamental problems in the analysis of Petri nets, because it appears as a subproblem of various basic problems [6]. Since LFS is shown to be NP-hard [5], various heuristics has been proposed to solve the problem of practical size in a reasonable time [7]–[9]. In this paper, we propose a new algorithm for LFS. It is based on the partial order verification technique for concurrent programs [1], [3], [4]. In this technique, a partial order defined on the set of transitions is used to compute a reduced state space that preserves properties to be verified, such as existence of deadlocks and properties written in the form of next-free linear time temporal logic. Interleaving unnecessary for the verification is excluded from the state space.

The proposed algorithm is orthogonal to existing heuristic algorithms, i.e., it can be combined with existing algorithms and can increase the performance.

2. Preliminaries

Let $\mathbb{N}$ denote the set of nonnegative integers. For a finite set S, we identify a function $f : S \to \mathbb{N}$ with a $|S|$ dimensional vector of nonnegative integers and write $f \in \mathbb{N}^S$.

A Petri net is a triple $N = (P, T, A)$, where P and T are disjoint finite sets, and $A : (P \times T) \cup (T \times P) \to \mathbb{N}$ is a function. Each element in P is called *a place*, and each element in T *a transition*. A Petri net is usually drawn as a directed bipartite graph, with two set of nodes P and T, and arcs represented by the function A.

As usual for each $v \in P \cup T$, let ${}^\bullet v = \{w \mid (w, v) \in A\}$ and $v^\bullet = \{w \mid (v, w) \in A\}$, and for $V \subseteq P \cup T$, let ${}^\bullet V = \cup_{v \in V} {}^\bullet v$ and $V^\bullet = \cup_{v \in V} v^\bullet$. A function $m : P \to \mathbb{N}$ is called *a marking*, which is a state of the Petri net.

The behavior of a Petri net $N = (P, T, A)$ is represented by a transition system $TS_N = (\mathbb{N}^P, T, \to)$, where $\to \subseteq \mathbb{N}^P \times T \times \mathbb{N}^P$ is defined by $(m, t, m') \in \to$ if and only if

$$\forall p \in P : \quad [m(p) \geq A(p,t) \wedge \\ m'(p) = m(p) + A(t,p) - A(p,t)].$$

We write $m \xrightarrow{t} m'$ instead of $(m, t, m') \in \to$. As usual, the transition relation $\to$ is extended to finite sequences of transitions.

Let T^* denote the set of all finite sequences (including the empty sequence) over T. Let $\psi : T^* \to \mathbb{N}^T$ be a function such that $\psi(\sigma)(t)$ denotes the number of occurrences of transition t in σ, e.g., $\psi(abaabbc)(a) = 3$. We write $m \xrightarrow{\sigma}$ to denote $\exists m' : m \xrightarrow{\sigma} m'$. When $m \xrightarrow{t}$, t is called *enabled* in marking m. Let $en(m)$ denote the set of enabled transitions in m.

3. New Algorithm for LFS

3.1 Problem Description

Let $N = (P, T, A)$ be a Petri net. A non-negative integer vector $x \in \mathbb{N}^T$ is called a firing count vector. Then the problem LFS is formulated as follows:
Given: A Petri net $N = (P, T, A)$, an initial marking m_0, and a firing count vector $x \in \mathbb{N}^T$;
Find: a firing sequence $\sigma \in T^*$ such that $m_0 \xrightarrow{\sigma}$ and $\psi(\sigma) = x$.

3.2 Persistent Sets

We introduce the notion of *persistent sets*, which will be used in the algorithm.

Let $N = (P, T, A)$ be a Petri net. A subset W of T is called a persistent set at marking m if for any $t \in W$ and any $\sigma \in (T - W)^* : m \xrightarrow{\sigma t} m' \wedge m \xrightarrow{t} \Rightarrow m \xrightarrow{t\sigma} m'$.

Manuscript received March 19, 2001.
Manuscript revised June 5, 2001.

[†]The authors are with the School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923-1292 Japan.
a) E-mail: hira@jaist.ac.jp
b) E-mail: hirohide@jaist.ac.jp

A sufficient condition for $W \subseteq T$ to be a persistent set can be obtained from the results on *stubborn set theory* [4].

Proposition 1: Let $N = (P, T, A)$ be a Petri net. A subset W of T is a persistent set at marking m if for each $t \in W \cap en(m)$ and $p \in P$, one of the following (i), (ii) holds:

(i) $\forall t' \in T - W$:
$min(A(t,p), A(t',p)) \geq min(A(p,t), A(p,t'))$;
(ii) $\forall t' \in T - W$:
$min(A(t,p), A(p,t')) \geq min(A(p,t), A(t',p))$.

Proof: Suppose that $m \xrightarrow{\sigma t}$ and $m \xrightarrow{t}$ for $\sigma = t_1 t_2 \cdots t_n \in (T-W)^*$ and $t \in W$. Let $m \xrightarrow{t_1 \cdots t_k} m_k$ $(k = 1, \cdots, n)$.

For a place $p \notin {}^\bullet t$, the right-hand side of each inequality is zero, and therefore (i) and (ii) always hold. For a place $p \in {}^\bullet t$, the conditions (i), (ii) can be interpreted as follows.

(i) When $A(p,t) < A(p,t')$, $A(p,t) \leq A(t,p)$ holds and therefore a firing of t does not decrease the number of tokens in p. When $A(p,t) \geq A(p,t')$, $A(p,t') \leq A(t,p)$ holds and therefore p has sufficient tokens to enable t' after a firing of t. Moreover, when $A(p,t) \geq A(t',p)$, $A(t',p) \geq A(p,t')$ holds, and therefore a firing of t' does not decrease the number of tokens in p.

(ii) When $A(p,t) < A(t',p)$, $A(p,t) \leq A(t,p)$ holds and therefore a firing of t does not decrease the number of tokens in p. When $A(p,t) \geq A(t',p)$, $A(t',p) \leq A(t,p)$ holds and therefore p has sufficient tokens to enable t' after a firing of t. Moreover, when $A(p,t) \geq A(t',p)$, $A(t',p) \leq A(p,t')$ holds and therefore a firing of t' does not increase the number of tokens in p.

We first show that $m_k \xrightarrow{t}$ $(k = 1, \cdots, n)$. Assume that t is not enabled in m_k. Then there exists a place p from which the firing of $t_1 \cdots t_k$ removes tokens necessary for t to be enabled. Hence, $A(p,t) > A(t_i,p)$ holds for some $i \in \{1, \cdots, k\}$ (otherwise, p keeps sufficient tokens). If condition (i) holds for t and p, then any transition $t' \in T - W$ does not decrease the number of tokens in p. This contradicts the fact that $m \xrightarrow{t}$. If condition (ii) holds, then any transition $t' \in T - W$ does not increase the number of tokens in p. This contradicts the fact that $m_k \xrightarrow{t_{k+1} \cdots t_n t}$.

In addition, since the firing of t does not make t_i disabled, we have the following diagram.

$$\begin{array}{ccccccc} m - t_1 \rightarrow & m_1 - t_2 \rightarrow & \cdots & m_{n-1} - t_n \rightarrow & m_n \\ | & | & & | & | \\ t & t & & t & t \\ \downarrow & \downarrow & & \downarrow & \downarrow \\ m' - t_1 \rightarrow & m'_1 - t_2 \rightarrow & \cdots & m'_{n-1} - t_n \rightarrow & m'_n \end{array}$$

□

Table 1 Algorithm LFSPOM.

Algorithm LFSPOM:
Stack is empty;
push $\langle m_0, x, \lambda \rangle$ onto *Stack*;
while *Stack* is not empty **do**
  pop $\langle m, y, \sigma \rangle$ from *Stack*;
  if $y = \mathbf{0}$ **then** *output* σ and *halt*;
  compute $W(m, y)$;
  for all t in $W(m,y) \cap en(m)$ **do**
    push $\langle m', y - \psi(t), \sigma t \rangle$ onto *Stack*, where $m \xrightarrow{t} m'$;
output "no".

3.3 Algorithm LFSPOM

We show the proposed algorithm LFSPOM (the algorithm for the Legal Firing Sequence problem based on Partial Order Methods) in Table 1. It is a depth-first search algorithm except that at each step only transitions in the set $W(m,y)$ are considered for firing, where m is the current marking and y is the remaining firing count vector. If we choose the set $T_y := \{t \in T \mid y(t) > 0\}$ as $W(m,y)$, then the algorithm behaves like an exhaustive search algorithm using the depth-first search.

Remark: In the algorithm LFSPOM, a vector $y - \psi(t)$ and a sequence σt is added to the data in the stack. However, this is just for simplifying the description. We can implement the algorithm in which only a vector m is push onto the stack. When backtrack occurs, the remaining firing count is recoverd and the last transition of the current firing sequence is removed.

The set $W(m,y)$ is defined as follows. If $en(m) \cap T_y = \emptyset$, then $W(m,y) = \emptyset$. Otherwise, $W(m,y)$ is a nonempty subset of T_y that satisfies the following conditions:

1. $W(m,y) \cap en(m) \neq \emptyset$.
2. In the subnet $N|T_y$, $W(m,y)$ is a persistent set at marking m, where $N|T_y$ denotes the subnet of N consisting of transitions T_y and all places adjacent to T_y with arcs connecting these nodes.
3. If $t \in W(m,y) - en(m)$, then for any $\sigma \in (T_y - W(m,y))^* : m \xrightarrow{\sigma} m' \Rightarrow t \notin en(m')$.

We can easily obtain the following sufficient condition for the requirement 3.

Lemma 2: Let $t \in W(m,y) - en(m)$. Suppose that for all $t' \in T_y - W(m,y)$: $m(p) < A(p,t) \Rightarrow A(t',p) = 0$. Then for any $\sigma \in (T_y - W(m,y))^* : m \xrightarrow{\sigma} m' \Rightarrow t \notin en(m')$.

Using the results in Proposition 1 and Lemma 2, the set $W(s,y)$ can be computed by the following procedure.

1. Select a transition $t \in en(m) \cap T_y$ and let $W(m,y) := \{t\}$.
2. Repeat the following until no more transition is

added.

(i) If for $t \in W(m,y) \cap en(m)$ there exists a transition of $t' \in T_y - W(m,y)$ that does not satisfy the condition (i) and (ii) of Proposition 1, then add t' to $W(m,y)$.
(ii) If for $t \in W(m,y) - en(m)$ there exists a transition $t' \in T_y - W(m,y)$ that does not satisfy the condition of Lemma 2, then add t' to $W(m,y)$.

This procedure eventually terminates since $W(s,y) = T_y$ always satisfies the conditions.

We now prove the correctness of the algorithm.

Lemma 3: Suppose that $m \xrightarrow{\sigma} m'$, $\psi(\sigma) = y$, and that m is visited by LFSPOM. Then m' is also visited by a firing sequence σ' such that $\psi(\sigma) = \psi(\sigma')$ in the algorithm.

Proof: We use induction on the length of σ. It is clear when the length is 0. We consider the case that the length is greater than 0. Every transition in $W(m,y)$ is contained in σ since $W(m,y) \subseteq T_y$. Suppose that $t \in W(m,y)$ occurs first in σ, i.e., $\sigma = \sigma_1 t \sigma_2$ and σ_1 has no transitions of $W(m,y)$. By the requirement 3 of $W(m,y)$, t is enabled in m, and by the requirement 2, $m \xrightarrow{t\sigma_1\sigma_2} m'$ holds. By the induction hypothesis, our result follows. □

Theorem 4: Algorithm LFSPOM correctly solves LFS.

Proof: If LFSPOM outputs a firing sequence σ, then it is legal and $\psi(\sigma) = x$. Assume that LFS has a solution σ such that $m_0 \xrightarrow{\sigma} m$ and $\psi(\sigma) = x$. By Lemma 3, m is visited by a firing sequence σ' such that $\psi(\sigma) = \psi(\sigma')$ in the algorithm, and σ' is also a solution for the LFS. □

We now discuss computational complexity of LFSPOM. We first need to say that the partial order methods does not improve the worst case complexity. There exists a case that at every step $W(m,y)$ contains all enabled transitions of T_y. Then the execution of the algorithm requires exponential time. However, if we apply the algorithm to conflict-free nets, then we can always find the set $W(m,y)$ containing exactly one transition. For conflict-free nets, there exists a greedy algorithm that solves LFS, i.e., at each step selecting just one enabled transition within the remaining firing count [6]. Algorithm LFSPOM behaves like the greedy algorithm when it is applied to conflict-free nets. When the net has a conflict-free substructure, LFSPOM eliminates redundant search corresponding to the structure.

3.4 Experimental Results

We have applied the proposed algorithm to LFS for Petri nets describing well-known dining philosophers problem (Fig. 1). LFSPOM is compared with the exhaustive search algorithm (Algorith EXAUSTIVE in the table) which is the same algorithm as LFSPOM except that $W(m,y)$ is always set to T_y.

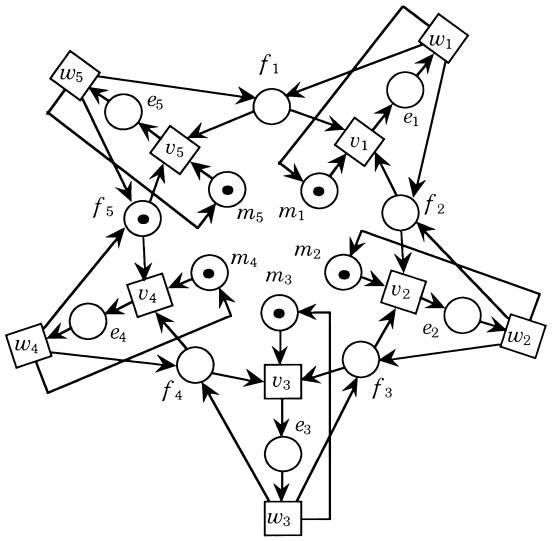

Fig. 1 A Petri net describing DPP.

Table 2 Experimental results.

Ex.	#FC	EXHAUSTIVE		LFSPOM	
		Time (sec.)	#BT	Time (sec.)	#BT
(1)	15	1.23	38,940	0.67	3,210
(2)	17	6.45	202,816	2.08	10,142
(3)	19	24.32	754,436	5.18	25,416
(4)	21	84.29	2,266,492	11.28	55,044
(5)	23	194.62	5,861,332	21.94	107,620

Algorithms are implemented by C language, and are executed on SUN Ultra5 (CPU: UltraSparc-IIi 270 MHz, Main Memory: 128 MB).

First we need to say that LFSPOM does not show better performance than EXAUSTIVE when the number of backtracks occurred in the execution is small. All examples shown in Table 2 have no legal firing sequences satisfying the requirements, and therefore many backtracks occur. Each example "(k)" is an instance of LFS having the following firing count:

$$X(w_1) = k,\ X(v_1) = k+1;$$
$$X(w_i) = 1,\ X(v_i) = 2\ (i = 2, \cdots, 5).$$

In the table, #FC denotes the total number of firing counts, Time (sec.) the CPU time, and #BT the number of backtracks. We can observe that the number of backtracks is reduced drastically in the execution of LFSPOM.

The proposed algorithm is essentially the same as the stubborn set method [4]. The stubborn set method generates a reduced state space that preserves all dead markings, while LFSPOM explores a reduced state space that preserves every marking in which the remaining firing count is 0. In both methods, a selective

set of transitions W is computed at each marking m so that it satisfies the following: If $m \xrightarrow{\sigma} m'$ and m' is a marking to be preserved, then there exists some $t \in W$ such that $\sigma = \sigma_1 t \sigma_2$ and $m \xrightarrow{t\sigma_1\sigma_2} m'$. The stubborn set method and similar methods were shown to be effective through many experiments (see [2], [4]). Therefore, we can expect that the proposed method will also work in many instances.

4. Conclusion

As written in the introduction, there are several heuristics proposed for LFS. Algorithm LFSPOM can be combined with these heuristics, i.e., LFSPOM will work as an accelerator of existing heuristic algorithms based on search algorithms.

References

[1] P. Godefroid, "Using partial orders to improve automatic verification methods," Lecture Notes in Computer Science, vol.531, pp.176–185, Springer, 1990.
[2] D.A. Peled, V.R. Pratt, and G.J. Holzmann (eds.), "Partial order methods in verifications," DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol.29, 1996.
[3] D.K. Probst and H.F. Li, "Using partial-order semantics to avoid the state explosion problem in asynchronous systems," Lecture Notes in Computer Ssience, vol.531, pp.147–155, Springer, 1990.
[4] A. Valmari, "Stubborn sets for reduced state space generation," Lecture Notes in Computer Science, vol.483, pp.491–515, Springer, 1990.
[5] T. Watanabe, Y. Mizobata, and K. Onaga, "Time complexity of legal firing sequence and related problems of Petri nets," IEICE Trans., vol.E71, no.12, pp.1400–1409, 1989.
[6] T. Watanabe and M. Yamauchi, "A survey on the legal firing sequence problem of Petri nets," IEICE Technical Report, CST97-33, 1997.
[7] M. Yamauchi, S. Nakamura, and T. Watanabe, "An approximation algorithm for the legal firing sequence problem of Petri nets," IPSJ SIG Notes, 93-AL-33, pp.17–24, 1993.
[8] M. Yamauchi and T. Watanabe, "A new heuristic algorithm for the legal firing sequence problem of Petri nets," IEICE Technical Report, CST97-32, 1997.
[9] M. Yamauchi, M. Hashimoto, and T. Watanabe, "A heuristic algorithm FSD for the legal firing sequence problem of Petri nets," IEICE Technical Report, CST98-6, 1998.