Information Sciences 363 (2016) 190–197

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Complexity of the deadlock problem for Petri nets modeling resource allocation systems

Guanjun Liu*

*Department of Computer Science and Key Lab of Ministry of Education for Embedded Systems and Services Computing, Tongji University, Shanghai 201804, China*

ARTICLE INFO

*Article history:*
Received 24 December 2014
Revised 8 September 2015
Accepted 12 November 2015
Available online 27 November 2015

*Keywords:*
Petri nets
Deadlock
Resource allocation systems
Collaboration
PSPACE-completeness
NP-completeness

ABSTRACT

Petri nets are widely used to model and analyze Resource Allocation Systems (RASs). Since they are a kind of structuralized formal method, they can well describe the allocation/release of resources and their markings can directly reflect whether an RAS enters a (partial) deadlock caused by misallocating resources. In general, the key step of preventing/avoiding deadlocks is to decide if deadlocks occur or not in an RAS. This paper is about the complexity of deciding the deadlock problem for Petri nets modeling RAS. We define a very general class of Petri nets called Petri Nets of Resource Allocation (PNRA) to model as many RASs as possible. PNRAs not only focus on the resources shared by processes also pay attention to the interaction/collaboration among processes. We show that the deadlock problem is PSPACE-complete for PNRAs. This paper also proves that for the well-known G-system as a subclass of PNRAs, the deadlock problem is NP-complete.



## 1. Introduction

Petri nets as a kind of mathematical model can well characterize concurrent systems regarding resource allocation because they can clearly depict the relation of allocating and releasing resources. For example, lots of subclasses of Petri nets such as PPN [3,17], $S^3$PR [13,25,27,37,41–45,51], $WS^3$PR [26,28], $S^3PGR^2$ [5,11,36,40], CADPN [18], PNR [19], ERCN-merged net [49], and G-systems [53] have been used to model flexible manufacturing systems that are typical RASs. A flexible manufacturing system contains a group of manufacturing processes that share a set of common resources [8,9,21–23,46,50,52]. A manufacturing process usually contains several manufacturing steps and each step is supported by some resources such as machine and robot. In these Petri net classes, some reflect that each manufacturing step only needs one unit from one type of resources (e.g., $S^3$PR), some reflect that each step needs multiple units from one type of resources (e.g., $WS^3$PR), and some reflect that each step needs multiple units from multiple types of resources (e.g., ERCN-merged net). Additionally, different subclasses have different requirements for their manufacturing processes. Some processes permit choice but not parallel (e.g., $S^3$PR and $WS^3$PR) while some permit parallel (e.g., PNR and ERCN-merged net). G-system is the largest one in the above classes [24].

The systems modeled by the above Petri net classes own two common characters. One is that every resource is restored to its initial state when a system terminates correctly. The quantity of each resource is not increased during the execution period. Another one is that all processes of a system are mutually independent. Here the independence means that there is no collaboration/interaction between any two processes [4,12,30,31]. For some other concurrent systems regarding resource allocation,

* Tel.: +86 02169589864.
*E-mail address:* liuguanjun@tongji.edu.cn

http://dx.doi.org/10.1016/j.ins.2015.11.025
0020-0255/

however, multiple processes need to collaborate/interact in order to execute one task. For example, grid computing [16,34] may be viewed as such a kind of system in which multiple processes need to not only share some common resources but also interact with each other. Unfortunately, The above Petri net classes cannot model these systems since they do not consider the collaboration/interactions among processes. Therefore, this paper defines a class of Petri nets called Petri Nets of Resource Allocation (PNRA) that is a more generalization than G-systems and can model the collaboration/interactions among processes.

Both G-system and PNRA face a common problem, i.e., deadlock. From the aspect of processes, a deadlock implies that some processes cannot reach their final states after they start. From the aspect of resources, a deadlock means that some resources cannot be released after they are allocated. Therefore, deadlock is undesired and a system should be deadlock-free before it is used [2,6,7,13,15,25,33,47,48]. It is necessary and important for a designed system to decide whether it has a deadlock. This paper focuses on the complexity of deciding the deadlock problem. Is it easy (solvable in polynomial time) or hard to decide this problem? If it is hard, is it NP-hard, or PSPACE-hard, or EXPTIME-hard? This paper proves that the deadlock problem is PSPACE-complete for PNRA and NP-complete for G-system.

The remainder of this paper is organized as follows. Section 2 introduces some basic knowledge of Petri nets. Section 3 defines PNRA and reviews G-system. Section 4 proves the PSPACE-completeness of the deadlock problem for PNRA. Section 5 shows the NP-completeness of this problem for G-system. Section 6 concludes this paper.

## 2. Preliminary

In this section, we review Petri nets. For more details, one may refer to [35] and [38].

Let $\mathbf{N} = \{0, 1, 2, \ldots\}$ be the set of non-negative integers and $\mathbf{N}^+ = \{1, 2, \ldots\}$ be the set of positive integers. Given $m \in \mathbf{N}^+$, let $\mathbf{N}_m = \{1, 2, \ldots, m\}$ be the set of integers from 1 to $m$ and $\mathbf{N}_m^0 = \mathbf{N}_m \cup \{0\}$ be the set of integers from 0 to $m$.

**Definition 2.1.** A *net* is a 4-tuple $N = (P, T, F, W)$, where $P$ is a set of *places*, $T$ is a set of *transitions*, $F \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs*, $W$: $F \to \mathbf{N}^+$ is a *weight function*, $P \cup T \neq \emptyset$, and $P \cap T = \emptyset$.

A transition $t$ is called an *input transition* of a place $p$ and $p$ is called an *output place* of $t$ if $(t, p) \in F$. *Input place* and *output transition* can be defined similarly. $\forall x \in P \cup T$, ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$ and $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$ are called the *pre-set* and *post-set* of $x$, respectively.

The *incidence matrix* of a net $N = (P, T, F, W)$ is a $|T| \times |P|$ matrix $U = [a_{ij}]$ such that $a_{ij} = W(i, j) - W(j, i)$ where $W(i, j)$ is the weight of the arc from transition $i$ to place $j$ and $W(j, i)$ is the weight of the arc from place $j$ to transition $i$. Notice that $W(x, y) = 0$ if there is no arc from node $x$ to node $y$. A nonnegative integer vector $I$ is a *P-semiflow* if $U \cdot I = 0$. $\|I\| = \{p \mid I(p) \neq 0\}$ is called the *support* of $I$. A P-semiflow $I$ is *minimal* if $\|I\|$ is not a superset of the support of any other P-semiflow and the components of $I$ are mutually prime.

Let $I$ be a P-semiflow and $t \in T$ be a transition in net $N = (P, T, F, W)$. Then, $\rho(t, I) = \sum_{p \in {}^\bullet t} I(p) \cdot W(p, t)$. This symbol is from [24].

A *marking* of $N = (P, T, F, W)$ is a mapping $M$: $P \to \mathbf{N}$. Place $p \in P$ is *marked* at $M$ if $M(p) > 0$. A marking may be viewed as a $|P|$-dimensional non-negative integer vector in which every element represents the number of tokens in the corresponding place. For example, $M = (1, 0, 6, 0)$ over $P = \{p_1, p_2, p_3, p_4\}$ represents that at $M$, $p_1$, $p_2$, $p_3$, and $p_4$ have 1, 0, 6, and 0 tokens, respectively. For convenience, $M$ is denoted as $M = \sum_{p \in P} M(p) \cdot p$ in this paper. For the above example, it is written as $M = p_1 + 6p_3$.

Given a marking $M$ over $P$ and a subset $P'$ of $P$, $M \uparrow P'$ is the projection of $M$ onto $P'$, i.e., $M \uparrow P'$ is a marking over $P'$ such that $\forall p \in P'$: $(M \uparrow P')(p) = M(p)$.

Given a marking $M_1$ over $P_1$ and a marking $M_2$ over $P_2$, $M = M_1 \cup M_2$ is a marking over $P_1 \cup P_2$ such that $\forall p \in P_1 \cap P_2$: $M(p) = max\{M_1(p), M_2(p)\}$, $\forall p \in P_1 \setminus (P_1 \cap P_2)$: $M(p) = M_1(p)$, and $\forall p \in P_2 \setminus (P_1 \cap P_2)$: $M(p) = M_2(p)$.

If $\forall p \in {}^\bullet t$: $M(p) \geq W(p, t)$, then transition $t$ is *enabled* at $M$, which is denoted as $M[t\rangle$. *Firing* an enabled transition $t$ produces a new marking $M'$, which is denoted as $M[t\rangle M'$, such that $\forall p \in P$: $M'(p) = M(p) - W(p, t) + W(t, p)$.

A marking $M_k$ is *reachable* from a marking $M$ if there exists a transition sequence $\sigma = t_1 t_2 \ldots t_k$ such that $M[t_1\rangle M_1[t_2\rangle \cdots \rangle M_{k-1}[t_k\rangle M_k$. Notation $M[\sigma\rangle M_k$ represents that $M$ reaches $M_k$ after firing sequence $\sigma$. The set of all markings reachable from $M$ in a net $N$ is denoted as $R(N, M)$. Notice that when no ambiguity is yielded, $R(N, M)$ is replaced by $R(M)$.

A net $N$ with an *initial marking* $M_0$ is called a *Petri net* and denoted as $(N, M_0)$.

A Petri net $(N, M_0)$ is *k-bounded* for a positive integer $k$ if $\forall p \in P$, $\forall M \in R(N, M_0)$: $M(p) \leq k$. A Petri net is *bounded* if it is *k-bounded* for a positive integer $k$.

A Petri net $(N, M_0)$ is *quasi-live* if $\forall t \in T$, $\exists M \in R(N, M_0)$: $M[t\rangle$.

A non-empty set $S \subseteq P$ of $N = (P, T, F, W)$ is a *siphon* if ${}^\bullet S \subset S^\bullet$. Siphon $S$ is *minimal* if it it contains no other siphon as a proper subset. Siphon $S$ is *controlled* in $(N, M_0)$ if $\forall M \in R(N, M_0)$, $\exists p \in S$: $M(p) \geq max\{W(p, t) \mid t \in p^\bullet\}$. $(N, M_0)$ satisfies the *controlled-siphon property* (*cs-property* for short) if each minimal siphon is controlled.

## 3. Petri nets of resource allocation

In this section, we first recall G-system and then define PNRA.