# 死锁的PETRI NETS 模型

吴启明 袁崇义

(云南软件中心，昆明) (中科院数学所，北京)

# A PETRI NETS MODEL OF DEADLOCK

Wu Qiming Yuan Chongyi

(*Yunan software center, kunming*) (*Institute of Mathematics, Academia sinica*)

## ABSTRACT

In this paper, we gave a formal model of OS based on Petri Nets, it transformed the deadlock problem into linear algebric problem. We obtained a sufficient and necessary condition of deadlock existence in OS, and an optimal method which removed all deadlocks and guaranteed the regular operation in the system. Specialling, for distributed OS, we have greatly improved the method of detect deadlock.

## 摘 要

本文给出了操作系统(OS) 的一种基于Petri 网的形式化模型，由此把死锁问题转化为线性代数问题。我们得到了OS 中死锁存在的充要条件，以及一种消除系统中所有的死锁、保证系统正常工作的最优化方法。特别是对于分布式OS，我们大大改进了以往的一些处理死锁的方法，最后还通过举例进行了说明。

## §1. 引 言

并发与非同步过程是计算机系统的主要特征之一。Petri 网作为一种描述系统行为、分析系统性质的数学模型，拥有一些其它模型所不具备的优点[1]，这为我们分析计算机系统的性质提供了强有力的工具。

Petri 网的描述能力是很强的，但Petri 网本身也有简单网(如C/E 网、P/T 网) 和高级网(如谓词/ 变迁网) 之分。就描述能力而言二者是等同的，譬如五个哲学家问题，用高级网和简单网均可进行模拟[1] [7]。尽管用高级网模拟会使网结构变得简单、状态空间减少，但是这给模拟后的分析带来一些困难。下文介绍的Petri 网是一种简单网。

本文提出了一种处理OS 死锁问题的方法。它以Petri 网为工具，通过建立OS 的形式化模型，把死锁问题化为简单的线性代数问题来处理。文中给出了OS 存在死锁的充要

1989 年8 月25 日收到，1990 年2 月12 日定稿。中国科学院自然科学基金资助课题。

条件，并解决了在任一时刻OS中有多少死锁，以及哪些进程正处于死锁状态问题；进一步，找出了为消除OS中所有死锁、保证系统正常工作而使撤消的进程个数最少的方法。对于分布式OS，不再象以往方法那样通过构造全局等待图来判断死锁[2]，大大改进了传统的方法。

## §2. Petri 网

Petri 网的一些概念，如P/T 网、关联矩阵、变迁规则、子网、变迁序列、网的图形表示等参见[1][7]，下面介绍两个重要概念：

定义1 关联矩阵

设$\sum = (S, T, F, K, W, M_0)$ 是有限的P/T 网，令$C: S \times T \longmapsto Z$ 为$\sum$ 上的关联矩阵，其中

$$C(s,t) = \begin{cases} w < t, s > & \text{若 } s \in t^{\cdot} \setminus {}^{\cdot}t \\ -w < s, t > & \text{若 } s \in {}^{\cdot}t \setminus t^{\cdot} \\ 0 & \text{其它} \end{cases}$$

定义2 网有界

设$\sum$ 是$P/T$ 网，$M_0$ 是$\sum$ 的初始标识，$[M_0>$ 为可达标识集。库所$s \in S$ 有界当且仅当存在正整数$n$ 使得：对于$\forall M \in [M_0>$ 满足$M(s) \leq n$。$\sum$ 有界当且仅当它的每个库所均有界。

## §3. 模型构造

在给出构造模型的算法之前，我们提出一些假设作为建立模型的前提条件。

3.1 基本假设

(1) 任何时刻，OS 中进程个数均有限。

(2) 每类资源为单一资源[2]。

(3) 一个进程在结束时释放它占有的全部资源。

(4) 若一个进程申请的资源都能得到满足，则它一定能在有限的时间内结束。

(5) 一个进程当且仅当所申请的资源得不到满足时才会变成等待状态。

(6) 资源或是闲置，或已分给某个进程，并且只能分给一个进程。

(7) 先讨论非分布式OS 情况。

3.2 构造算法

设任一时刻OS 有$n$ 个进程，不妨记为$P_1, \cdots, P_n$，则可按下述算法构造OS 在此时刻的一个Petri 网$\sum = (S, T, F, K, W, M_0)$：

算法

(1) 令$S = (P_1, \cdots, P_n)$。对$\forall s \in S$，令$M_0(S) = 0$，$K(S) = 1$。

(2) $T := 0$，且$k := n$。

(3) $i := 1$，执行(4)。

(4) 若$P_i$ 申请资源，而该资源正由$P_k$ 享有，则取$t \notin T$，使得$(< P_i, t > \in F) \wedge (< t, P_k > \in F) \wedge (W < P_i, t >= 1) \wedge w < t, P_k >= 1$，令$T := T + \{t\}$ 转(5)。

(5) $i := i + 1$，若$i < n$ 则转(4)，否则转(6)。

(6) $k := k - 1$，若$K = 0$ 则转(7)，否则转(3)。

(7) 停机。

**3.3 OS 的模型特征**

用上述算法构造的模型有如下特征，用以区别任何其它Petri 网:

(1) $M_0(S)=0, K(S)=1$, 对$\forall s \in S$。

(2) 对$\forall < x,y > \in F$，用$W < x,y >= 1$。

(3) 设$C$ 是$\sum$ 的关联矩阵，则对$\sigma=(1,1,\cdots,1)$ 有: $\sigma C=\underline{0}$。

(4) 存在向量$\mu=(u_1,\cdots,\mu_n)$，其中$u_i=0$ 或1，且$\sum u_i>0$，满足$u*C\leq\underline{0}$。

(5) $\sum$ 是有界的且是有限的$P/T$ 网。

根据上述特征，我们得到两个用于判断一个Petri 网是OS 模型的必要条件:

**性质1**　若存在向量$Y_{m\times 1}=(y_1,\cdots,y_m)^T$，使$C_{mn}Y_{m\times 1}>\underline{0}$，则关联矩阵为$C$ 的网$\sum$ 不是OS 的模型表示。

**证明:**　假设以$C$ 为关联矩阵的网$\sum$ 是OS 在某一时刻的模型，则由特征(3) 知: $\sigma C_{mm}=\underline{0}\Longrightarrow(\sigma C_{mm})\cdot Y_{m\times 1}=\underline{0}$，但$(\sigma C_{n\times m})Y_{m\times 1}=\sigma(C_{n\times m}Y_{m\times 1})>\underline{0}$，显然矛盾。证毕。

**性质2**　若存在向量$Y_{m\times 1}=(y_1,\cdots y_m)^T$ 使得$C_{n\times m}Y_{m\times 1}<\underline{0}$，则关联矩阵为$C_{nm}$ 的网不是OS 的模型表示。

**证明:**　与性质1 的证明类似。

在任一时刻，我们可由模型构造算法构造该时刻的OS 的模型，由此就可以借助于Petri 网的分析方法来处理在该时刻OS 是否有死锁等问题。

本文约定: 向量$X\geq\underline{0}$ 当且仅当$X$ 的每一分量不小于0; $X=\underline{0}$，当且仅当它的每一分量等于0，其它的如$X<\underline{0}$ 等可类似定义。

## §4. 分析

有了模型以后，就可以利用Petri 网的分析工具如矩阵代数、可达树等对该网本身的性质进行分析。

在任一时刻，设OS 在该时刻有$n$ 个进程$P_1,\cdots,P_n$，记$P=(P_1,\cdots,P_n)$，$Q\subseteq\{P_1,\cdots,P_n\}$，用$æ_Q$ 表示$Q$ 的特征向量。如$Q=\{P_1,P_2,P_3\}$，则$æ_Q=(1,1,1,0,\cdots,0)$，而$æ_P=(1,1,\cdots,1)$。设由模型构造算法得到该时刻的OS 模型为$\sum$，$C$ 为$\sum$ 的关联矩阵，则:

**定理1**　OS 在该时刻存在死锁$\Longleftrightarrow\sum$ 满足下列条件:

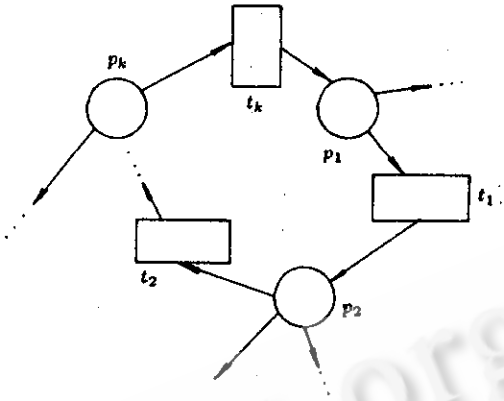
(1) 存在着向量$X_2>0$，且$X_2$ 由0、1 元组成并满足$CX_2=0$。

(2) 存在着由0、1 元组成的向量$X_1$，满足$X_1>0$ 且使(3) 成立，其中$X_1$ 是行向量。

(3) 若$M$ 是$C$ 关于$X_1$ 和$X_2$ 的导出矩阵*，则成立: $M\sigma^T=\sigma M=\underline{0}$，其中$\sigma=(1,1,\cdots,1)$。

证明($\Longrightarrow$)

设在该时刻OS 存在死锁; 不妨设进程$P_1,\cdots,P_k$ 处于死锁状态。依死锁出发的条件[2][3]，以及模型构造算法，必有OS 模型$\sum$ 的一个子网:




则该时刻的OS 模型$\sum$ 的关联矩阵有如下特点:

$$C=\begin{array}{c} \\ P_1 \\ P_2 \\ \\ P_k \\ \\ P_n \end{array}\begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & & t_k & & t_n \end{array} \\ \left(\begin{array}{cccccc} -1 & 0 & & 1 & & \\ 1 & -1 & & & 0 & * \\ & & \ddots & & & \\ 0 & & & 1 & -1 & \\ & & & & & * \end{array}\right)_{n\times n} \end{array}$$

令$C=\begin{pmatrix} M_{kk} & * \\ 0 & * \end{pmatrix}_{n\times n}$，其中 M=$\begin{bmatrix} -1 & 0 & & 1 \\ 1 & -1 & & 0 \\ & \ddots & \ddots & \\ 0 & & 1 & -1 \end{bmatrix}_{k\times k}$，显然成立$\sigma_M = M\sigma^r = 0$。

令$X_1=(\underbrace{1,\cdots,1}_{k\text{个}},0\cdots0)$，显然$X_1 > \underline{0}$

令$X_2=(\underbrace{1,1,\cdots,1}_{k\text{个}},0\cdots0)^r$，显然$X_2 > \underline{0}$，且$CX_2=\underline{0}$，而$C$ 关于$X_1$ 和$X_2$ 的导出矩阵就是$M$，因此(1)(2) (3) 成立。

($\Longleftarrow$)

依条件(1) 知，存在$X_2 > \underline{0}$ 满足$CX_2=\underline{0}$。因此由状态方程[8]: $M=M_0+CX$ 知，若取$X=X_2$，则有$M=M_0+CX_2=M_0$，说明从初始状态$M_0$ 出发经过某个变迁序列后又返回到初始状态$M_0$。又依(2) (3) 不妨设$X_1=(\underbrace{1,1,\cdots,1}_{k\text{个}},0\cdots0)$，$Z_2=(\underbrace{1,1,\cdots,1}_{k\text{个}},0\cdots0)^r$，因此$M$ 是$C$ 的前$k$ 行、$k$ 列组成的矩阵，由$\sigma M = M\sigma^r = \underline{0}$ 知，存在$m_{1i}$ 和$m_{ji}$ 满足: $(m_{1i}=-1)\wedge(m_{ji}=1)$ 不妨设$j=2$，由构造算法知: 此时$P_1$ 正申请$P_2$ 独享的资源，同理可得$P_2$ 正申请$P_3$ 占有的资源，$\cdots$，而$P_k$ 申请正被$P_1$ 占有的资源。因此由死锁出现的条件[3] 知: $P_1,\cdots,P_k$ 在该时刻将处于死锁状态。

推论1　若存在向量$X_1 > \underline{0}$ 使得$X_1C > \underline{0}$，则OS 在该时刻无死锁。

证明:　设该时刻OS 出现死锁，则由条件(1) 知存在向量$X_2 > \underline{0}$ 满足$CX_2 = \underline{0}$，故$(X_1C)X_2 > \underline{0}$，而$(X_1C)X_2 = X_1(CX_2) = \underline{0}$，矛盾。证毕。

推论2　若存在向量$X_1 > \underline{0}$，满足$X_1C < \underline{0}$，则OS 在该时刻不会出现死锁(deadlock)。

证明:　与推论1 类似。

定理2　若定理1 的条件(1) (2) (3) 成立，且$X_1 \neq \underline{0}$，则与$X_1$ 相对应的进程处于死锁状态。

证明:　从定理1 的证明过程可知定理成立。

例如　设$X_1 = (1,0,1,0,\cdots,0)$，则进程$P_1$ 和$P_3$ 处于死锁状态。

定理3　若定理1 的条件(1) 成立，那么满足条件(2) (3) 的$X_1$ 的个数就是该时刻OS 的死锁出现的个数。

证明:　从定理1 的证明过程可知定理成立。

定理4　存在一种消除OS 死锁的方法，它能保证系统正常工作，并使撤消的进程个数最少。

证明:　设任一时刻，系统在该时刻出现了$\gamma$ 个死锁$P_1,\cdots,P_\gamma$，其中$P_i(1 \leq i \leq \gamma)$ 均是进程集合，令$\beta_i$ 为$æ_{P_i}$ 的编码，例如若$æ_{P_i} = (1,0,1,1,0,\cdots,0)$，则$\beta_i = 10110\cdots0$。在定理证明之前，先证明三个命题:

命题1　若$æ_{P_1}\cdots æ_{P_\gamma}$ 线性相关，则只需要撤消一个进程就可消除死锁$P_1,\cdots,P_\gamma$。

证明:　由条件不妨设$æ_{P_1} = k_2\, æ_{P_2} + \cdots + k_\gamma\, æ_{P_\gamma}$，其中$k_i$ 为非负整数，且$\sum k_i > 0$，因此$P_1 \cap P_2$，$P_1 \cap P_3\cdots,P_1 \cap P_\gamma$ 至少含一个非空元素，不妨设为$P_t$，显然通过撤消进程$P_t$ 就可消除死锁$P_1\cdots P_\gamma$。

命题2　若存在$k$、$j$ 满足:　$\beta_k \oplus \beta_j \neq 0$，则只需要撤消一个进程就可消除死锁$P_k$ 和$P_j$。

证明:　由$(\beta_k \oplus \beta_j) \neq 0 \Longrightarrow P_k \cap P_j \neq 0$，不妨设$P_k \cap P_j = \{P_l,\cdots,\}$，显然撤消进程$P_l$ 就可消除死锁$P_k$ 和$P_j$。

命题3　若$æ_{P_1}\cdots æ_{P_j}$ 线性无关，且对$\forall i,k$ 有:　$(i \neq k) \wedge (\beta_i \oplus \beta_k) = 0$，则需要撤消$j$ 个进程才能消除死锁$P_1,\cdots,P_j$。

证明:　根据命题1 和命题2 可知命题成立。

由命题1、2、3 可证定理成立(略)。

定理5　定理4 的方法是一种最优的消除OS 所有死锁的方法。

证明:　略。

现在讨论分布式OS 的死锁处理问题。以往的方法是[2][3][4][5][6]:　每个Site 管理一个局部等待图，该图的顶点对应于所有这样一些进程:　它们当前正占有或申请占用局部于该Site 的资源;　对全局可能出现的死锁，通过“控制者”进程构造全局等待图来判断。但这样一来，每个Site 的局部等待图都要以信息方式传给“控制者”进程，如果信息传输延迟，那就会导致假死锁。另一方面管理全局等待图要增加系统的大量开销。

本文介绍的方法不需要构造全局等待图，而且传输的信息量和处理的信息量都大为减少。我们的方法是这样的:　每个Site 管理这样一个Petri 网，该网的库所是所有这样的进程:　它们当前正占有或申请占有局部于该Site 的资源(具体构造方法见前面)。一个Site 上是否出现死锁，可由定理1 来判断。

为判断全局是否出现死锁，不妨设有$n$ 个Site，且每个Site 管理的Petri 网分别是$\sum_1,\cdots,$

$\sum_n$，其关联矩阵分别是$C_1,\cdots,C_n$。其中$\sum_i=(S_i,T_i,F_i,K_i,W_i,M_{0i})$，令$S=\bigcup_{i=1}^{n}S_i$，$T=\bigcup_{i=1}^{n}T_i$，并令$|S|=m$，$|T|=k$，令$\bar{S}=(p_1,p_2,\cdots,p_m)$，$\bar{T}=(t_1,\cdots,t_k)$，将每个$C_i$扩充为$C_i':\bar{S}\times\bar{T}\to Z$，若$<s,t>\notin F_i$，则$C_i'<s,t>=0$，令$C=\sum_{i=1}^{n}C_i'$，对$C$而言可用定理1来判断全局是否会出现死锁。

定理6　全局中出现的死锁个数及最优消除全部死锁的方法由前述定理确定。

定理7　局部出现的死锁在全局中仍是死锁。

证明：　由定理1和定理2以及$C$的构造可知成立。

定理7从另一个方面说明判断全局死锁的方法是正确的。

## §5 举例

设一分布式OS有Site A和Site B，在Site A上有一进程A申请一资源，该资源是Site A拥有的，且正由进程$B$占有。在Site B上有一进程B申请A正占有的资源，而进程A申请$C$占有的资源，$C$申请$A$正占有的资源，且这些资源都局部于Site B。

根据构造算法我们得到：

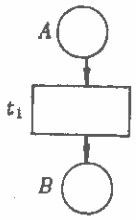



图1　Site A 的Petri 网

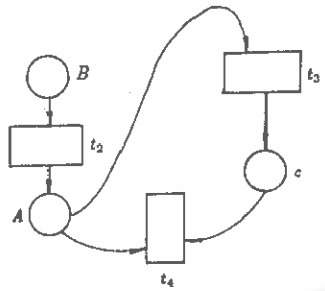



图2　Site B 的Petri 网

图1的关联矩阵：$C_1=\begin{matrix}&t_1\\A&\\B&\end{matrix}\begin{pmatrix}-1\\1\end{pmatrix}$

图2的关联矩阵$C_2$：$C_2=\begin{matrix}&t_2&t_3&t_4\\A&1&-1&-1\\B&-1&0&0\\C&0&1&-1\end{matrix}$，对$C_1$而言令$X_1=(0,1)$，显然$X_1C_1>\underline{0}$，因此由推论知Site A此时刻无死锁，这显然正确。令$X_2=(0,1,1)^\tau$，$X_1=(1,0,1)$，显然：$C_2X_2=\underline{0}$，而$C_2$关于$X_1$和$X_2$的导出矩阵是$M=\begin{pmatrix}-1&1\\1&-1\end{pmatrix}$，显然$\sigma M=M\sigma^\tau=\underline{0}$。因此由定理1和2知，Site B此时刻出现了死锁，且死锁是$\{A,C\}$。

为判断全局的死锁，由$C$的构造知：

$$C_1'=\begin{matrix}&t_1&t_2&t_3&t_4\\A&-1&0&0&0\\B&1&0&0&0\\C&0&0&0&0\end{matrix}\qquad C_2'=\begin{matrix}&t_1&t_2&t_3&t_4\\A&0&1&-1&1\\B&0&-1&0&0\\C&0&0&1&-1\end{matrix}$$

（下转第58页）