RAIRO
INFORMATIQUE THÉORIQUE

PIOTR BAK

A fast method of deadlock avoidance

RAIRO – Informatique théorique, tome 18, nº 1 (1984), p. 71-80.
<http://www.numdam.org/item?id=ITA_1984__18_1_71_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 18, n° 1, 1984, p. 71 à 80)

A FAST METHOD OF DEADLOCK AVOIDANCE (*)

by Piotr BAK [1]

Communicated by M. NIVAT

Abstract. — *The paper presents a fast deadlock avoidance method for the single resource type system. Proposed method is similar to the one described in Habermann. It is based on the process rank modification during resource granting or releasing.*

The auxiliary vector construction is simpler than in Habermann. The utilization of this method in multiple resource type system is also presented.

Résumé. — *On présente une méthode rapide pour éviter l'interblocage dans un seul type de ressources. Cette méthode ressemble à celle présentée par Habermann. Elle est basée sur l'idée de modifier le rang d'un processus quand les ressources sont allouées ou sont rendues. La méthode de construction des vecteurs auxiliaires est plus simple que celle de Habermann. On propose aussi l'utilisation de cette méthode pour les systèmes à types de ressources multiples.*

1. INTRODUCTION

System deadlock avoidance methods are based on constant preservation of so-called "system safe state" [4, 5, 2]. After every demand of resources the special-test is used to decide if the resources may be granted. The banker's algorithm is used but its execution time is proportional to the product of number of processes and number of resource types. The substantial computational complexity of this algorithm often limits its utilisation in practice. In [5] one of the first attempts for its simplification is presented, the execution time being in this case proportional to the number of resources only. The method presented in [5] is valid only for single-resource-type system.

The utilization of this results from [5] for multiple-resource-type system is presented in [1]. In this paper the deadlock prevention method [3] for different types of resources has been used.

In a safety test proposed in [5] the "state vector" must be hold. However its actualisation is complicated. The paper [5] dosen't present the complete proof

(*) Received in February 1982.

[1] Polish Academy of Sciences Complex Control System Dept. Bałtycka 5, 44-100 Gliwice, Poland.

of the proposed condition of system safe state which causes its modifications to be difficult.

In this paper a new method of deadlock avoidance for the single-resource-type system as well as utilization of this method for multiple-resource-type system is presented.

2. ASSUMPTIONS AND DEFINITIONS

In the paper we will consider, a system which consists of the finite set of sequential processes $\mathscr{P}=\{P_1, P_2, \ldots, P_n\}$, and the finite set of types of resources $\mathscr{R}=\{R_1, R_2, \ldots, R_m\}$.

The processes dynamically demand resources, and all necessary conditions of deadlock occurence are fulfilled [3].

The resource allocation state is defined similary as in [4] by means of matrices **B**, **C**, **D** and vector **a**.

Vector $\mathbf{a}=[a_i]$ $(i=1, 2, \ldots, m)$ describes the number of resources of each type available in the system (a_i – number of R_i type resources).

Matrix:

$$\mathbf{B}=[\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]=[b_{ij}] \qquad (i=1, 2, \ldots, m, j=1, 2, \ldots, n),$$

describes claims for the system resources (b_{ij} – maximum number of resources of type R_i possibly allocated to process P_j).

Matrix:

$$\mathbf{C}=[\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n]=[c_{ij}] \qquad (i=1, 2, \ldots, m, j=1, 2, \ldots, n),$$

describes the number of resources allocated to processes at a given moment of time (c_{ij}-number of resources of type R_i allocated to P_j).

Matrix:

$$\mathbf{D}=[\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n]=[d_{ij}] \qquad (i=1, 2, \ldots, m, j=1, 2, \ldots, n),$$

describes current demands of the processes (d_{ij}-number of resources of types R_i demanded by process P_j).

All demands of process P_j for resource type R_i are satisfed if $d_{ij}=c_{ij}$. The process demands $(d_{ij}-c_{ij})$ resources of type R_i if $d_{ij}>c_{ij}$, and whenever if $d_{ij}<c_{ij}$ process P_j declares the realise of $(c_{ij}-d_{ij})$ resources of type R_i.

We will introduce an auxiliary matrix:

$$\mathbf{Y}=\mathbf{B}-\mathbf{C}=[\mathbf{y}_1, y_2, \ldots, y_n]=[y_{ij}]\,(i=1, 2, \ldots, m, j=1, 2, \ldots, n),$$

which defines the number of resources that the processes can additionally request in order to complete. Let y_{ij} represent the "rank" of process P_j in relation to the type R_i resources [5].

In the paper we will consider only the system realizable state [4, 5].

The presentation of the safety test is based on the following definitions:

DEFINITION 2.1: We say that system is in its local safe state with regard to resource type R_i when it is possible to compose the processes realization sequence:

$$P_{f_i(1)} P_{f_i(2)} \ldots P_{f_i(n)} \tag{2.1}$$

such that:

$$\forall_{k \in [1, n]}, \quad b_{if_i(k)} \leq r_i + \sum_{l \leq k} c_{if_i(l)}, \tag{2.2}$$

where: $b_{if_i(k)}$, $c_{if_i(k)}$ denote maximal and allocated number of resources of type R_i to the process which occupies k-th place in the sequence, respectively; r_i- describes how many resources of type R_i remain free at a given moment of time; $f_i(k) = z$ denotes that the process $P_z \subset \mathscr{P}$ occupies the k-th place in the sequence.

DEFINITION 2.2: We say that the system is in its safe state when it is possible to compose the processes realization sequence:

$$P_{f(1)} P_{f(2)} \ldots P_{f(n)}, \tag{2.3}$$

$$\forall_{k \in [1, n]}, \quad \mathbf{b}_{f(k)} \leq \mathbf{r} + \sum_{l \leq k} \mathbf{c}_{f(l)}, \tag{2.4}$$

where: $\mathbf{b}_{f(k)}$, $\mathbf{c}_{f(k)}$-vector of maximal demands and vector of allocated number of resources to the process which occupies k-th place in the sequence, respectively:

$$\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix}$$

– describes how many resources of each remain free at the given moment of time;

— $f(k)=z$ denotes that the process $P_z \in \mathscr{P}$ occupies the k-th places in the sequence. It is easy to show that the Def. 2.2 is equivalent to safe state definition presented in [4].

The set of processes $\mathscr{P}$ is divided into two disjoint subsets:

$$\mathscr{P}=\mathscr{P}_1 \cup \mathscr{P}_2 \qquad (\mathscr{P}_1 \cap \mathscr{P}_2)=0.$$

Subset $\mathscr{P}_2$ is a set of passive processes for which:

$$\underset{P_j \in \mathscr{P}_2}{\forall}, \quad \underset{i \in [1, m]}{\forall}, \qquad c_{ij}=0$$

and $\mathscr{P}_1$ is a set of active and suspended processes, for which:

$$\underset{P_j \in \mathscr{P}}{\forall}, \quad \underset{i \in [1, m]}{\exists}, \qquad c_{ij} \neq 0.$$

Let u describe cardinality of subset $\mathscr{P}_1$.

3. A SAFETY TEST

We have noticed that the banker's algorithm may be used as the safety test. Because of its substantional complexity the simpler methods are being looked for.

To present the proposed safety test we will prove the following theorem:

THEOREM 3.1: *The realizable system state is a local safe state for the $R_i \subset \mathscr{R}$ type of resources if and only if in this state there exists the sequence of all active and suspended processes:*

$$P_{f_i(1)} P_{f_i(2)} \ldots P_{f_i(u)}, \tag{3.1}$$

which satisfies the following conditions:

$$1. \quad y_{if_i(1)} \leqslant y_{if_i(2)} \ldots \leqslant y_{if_i(u)}, \tag{3.2}$$

$$2. \quad \underset{j \in [1, u]}{\forall}, \quad b_{if_i(j)} - c_{if_i(j)} \leqslant r_i + \sum_{k=1}^{j-1} c_{if_i(k)}. \tag{3.3}$$

Proof: First we assume that the system state is a local safe state with regard to R_i type of resources. Thus, there exists at least one local safe sequence of processes for resource type R_i, for example:

$$P_{g_i(1)} P_{g_i(2)} \ldots P_{g_i(k)} \ldots P_{g_i(n-1)} P_{g_i(n)}. \tag{3.4}$$

Let:

$$P_{g_i(k)} \in \mathscr{P}_2 \quad \text{so} \quad c_{ig_i(k)} = 0.$$

From the Def. 2.1 it results that the process $P_{g_i(k)}$ can by moved at the end of the above sequence without changing the locations of other processes. In the same way we move all passive processes at the end of (3.4), so finally we have:

$$(3.5) \qquad P_{h_i(1)} P_{h_i(2)} \ldots P_{h_i(l)} P_{h_i(l+1)} \ldots P_{h_i(u)} P_{h_i(u+1)} \ldots P_{h_i(n)}$$

which satisfies:

$$(3.6) \qquad \forall_{j \in [u+1, n]}, \quad c_{ih_i(j)} = 0.$$

In order to examine that the state is local safe state for resource type R_i we need to examine only the subset of active and suspended processes. Now we assume, that the sequence of processes:

$$(3.7) \qquad P_{h_i(1)} P_{h_i(2)} \ldots P_{h_i(l-1)} P_{h_i(l)} P_{h_i(l+1)} P_{h_i(l+2)} \ldots P_{h_i(u)}$$

satisfies:

$$(3.8) \qquad \exists_{l \in [1, u]}, \quad y_{ih_i(l+1)} \geqslant y_{ih_i(l)}.$$

Writing inequalities (2.2) for the elements l and $(l+1)$ we have:

$$(3.9) \qquad b_{ih_i(l)} - c_{ih_i(l)} \leqslant r_i + \sum_{k=1}^{l-1} c_{ih_i(k)},$$

$$(3.10) \qquad b_{ih_i(l+1)} - c_{ih_i(l+1)} \leqslant r_i + \sum_{k=1}^{l} c_{ih_i(k)}.$$

Because condition (3.8) is satisfied, the inequalities (3.9) and (3.10) can be rewritten as below:

$$(3.11) \qquad b_{ih_i(l+1)} - c_{ih_i(l+1)} \leqslant r_i + \sum_{k=1}^{l-1} c_{ih_i(k)},$$

$$(3.12) \qquad b_{ih_i(l)} - c_{ih_i(l)} \leqslant r_i + \sum_{k=1}^{l-1} c_{ih_i(k)} + c_{ih_i(l+1)}.$$

Therefore processes $P_{h_i(l)}$ and $P_{h_i(l+1)}$ in the sequence (3.7) can be swapped with the locations of the other processes left unchanged. It is further possible to swap all other processes in the sequence (3.7) for which the conditions (3.2) and (3.3) are satisfied. Finally it is possible to obtain the sequence with properties (3.2) and (3.3). We can extend this sequence for all system's processes by placing the passive processes at the end of sequence (3.7), so that the system state is safe, from Def. 2.1, with regard to the R_i resource type.

THEOREM 3.2: *The realizable state is local safe state for the $R_i \in \mathcal{R}$ resource type if the inequality:*

$$\mathbf{v}_i \leqslant \mathbf{q}_i, \tag{3.13}$$

is satisfed.

In (3.13) we have denoted:

$$\mathbf{v}_i^T = [v_{i1}, v_{i2}, \ldots, v_{ij}, \ldots, v_{ia_{\max}}], \tag{3.14}$$

$$v_{ij} = \sum_{k=j}^{a_i} w_{ik}, \tag{3.15}$$

$$\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{ij}, \ldots, w_{ia_i}], \tag{3.16}$$

w_{ij} = the number of R_i type resources allocated to the process $P_k \in \mathcal{P}$ with the rank $y_{ik} = j-1$:

$$\mathbf{q}_i^T = [q_{i1}, q_{i2}, \ldots, q_{ij}, \ldots, q_{ia_{\max}}], \tag{3.17}$$

$$q_{ij} = \begin{cases} a_i - j + 1 & \text{for } j \leqslant a_i, \\ 0 & \text{for } j > a_i, \end{cases}$$

$$a_{\max} = \max_{a_i} \{ a_1, a_2, \ldots, a_i, \ldots, a_m \}.$$

Proof: From Def. 2.1 we have that the system state is safe, for R_i type of resource, if there exists the realisation sequence of processes (2.1) satisfying (2.2).

We will divide the set $\mathcal{P}_1 \subset \mathcal{P}$, with regard to the resource type R_i, into subsets:

$$\Delta_1, \Delta_2, \ldots, \Delta_k \ldots, \Delta_l, \ldots, \Delta_{a_i}$$

satisfying:

$$\forall_{P_j \in \Delta_k}, \quad y_{ij} = k-1,$$

i. e.: each process $P_j \in \Delta_k$ needs at most $(k-1)$ resources of type R_i to complite.

Let as assume that the above subsets are ordered in the following way:

$$\forall_{P_p \in \Delta_k}, \quad \forall_{P_j \in \Delta_l}, \qquad y_{ip} \leqslant y_{ij}. \tag{3.20}$$

The verification if the system state is local safe state is then possible (by means of the Theorem 3.1) by subsequent verification of the following subsets:

$$1) \quad \exists_{P_j \in \Delta_1}, \qquad b_{ij} - c_{ij} \leqslant r_i, \tag{3.21}$$

$$\vdots$$

$$k) \quad \exists_{P_j \in \Delta_k}, \qquad b_{ij} - c_{ij} \leqslant r_i + \sum_{P_p \in \Delta^{k-1}} c_{ip}, \tag{3.22}$$

$$\vdots$$

$$l) \quad \exists_{P_j \in \Delta_l}, \qquad b_{ij} - c_{ij} \leqslant r_i + \sum_{P_p \in \Delta^{l-1}} c_{ip}, \tag{3.23}$$

where:

$$\Delta^{k-1} = \Delta_1 \cup \Delta_2 \cup \ldots \cup \Delta_{k-1},$$

$$\Delta_{l-1} = \Delta_1 \cup \Delta_2 \cup \ldots \cup \Delta_{l-1}.$$

We find that the system state is not local safe state, with regard to the resource number R_i, when at least one inequality is not satisfied.

Let all processes in the subsets:

$$\Delta_1, \Delta_2, \ldots, \Delta_{l-1},$$

satisfy the above inequalities. Now, we are to test subset Δ_l, so:

$$\forall_{P_j \in \Delta_l}, \quad b_{ij} - c_{ij} \leqslant r_i + \sum_{P_p \in \Delta^{l-1}} c_{ip}. \tag{3.24}$$

We may write the right part of the inequality (3.24) in the form:

$$r_i + \sum_{P_p \in \Delta^{l-1}} c_{ip} = a_i - \sum_{P_p \in \Delta^{a_i}} c_{ip} + \sum_{P_p \in \Delta^{l-1}} c_{ip} = a_i - \sum_{P_p \in \Delta_l^{a_i}} c_{ip}, \tag{3.25}$$

where:

$$\Delta_l^{a_i} = \Delta_l \cup \Delta_{l+1} \cup \ldots \cup \Delta_{a_i}.$$

From (3.19) we have that the number of resources needed by process $P_j \in \Delta_l$ to complete itself equals $(l-1)$. Hence according to (3.24) and (3.25):

$$a_i - \sum_{P_p \in \Delta_l^{a_i}} c_{ip} \geqslant l-1 \tag{3.26}$$

and:

$$\sum_{P_p \in \Delta_l^{a_i}} c_{ip} \leqslant a_i - l + 1. \tag{3.27}$$

Notice, that if subset Δ_l is empty and if all previous subsets satisfy conditions (2.2) then the condition:

$$\sum_{P_p \in \Delta_{l-1}^{a_i}} c_{ip} \leqslant a_i - l. \tag{3.28}$$

is also satisfied.

From above, we have that condition (3.27) is also satisfied if:

$$\sum_{P_p \in \Delta_l} c_{ip} = 0. \tag{3.29}$$

If condition (3.28) is not satisfied we don't need to verify (3.27) for resource type $R_i \in \mathscr{R}$.

Condition (3.27) should be tested for all subsets Δ_l, so from (3.14), (3.15), (3.16) we have:

$$\mathbf{v}_i \leqslant \mathbf{q}_i, \tag{3.30}$$

where:

$$v_{il} = \sum_{P_p \in \Delta_{l-1}^{a_i}} c_{ip}, \tag{3.31}$$

which completes the proof.

EXAMPLE 3.1: We have the system in which:

$$\mathscr{P} = \{P_1, P_2\}, \qquad \mathscr{R} = \{R_1, \ldots, R_i, \ldots, R_m\}$$

and $a_i = 5$. Let the elements of matrix $\mathbf{B}$, for resource type R_i, equal:

$$b_{i1} = 4, \qquad b_{i2} = 3.$$

In the initial moment to the:

$$c_{i1} = 2, \qquad c_{i2} = 2,$$

i. e.:

$$y_{i1} = 3 \quad \text{and} \quad y_{i2} = 1.$$

Vectors $\mathbf{w}_i$, $\mathbf{v}_i$ and $\mathbf{q}_i$ are as below:

$$\mathbf{w}_i = [0 \quad 2 \quad 0 \quad 1 \quad 0],$$
$$\mathbf{v}_i^T = [3 \quad 3 \quad 1 \quad 1 \quad 0],$$
$$\mathbf{q}_i^T = [5 \quad 4 \quad 3 \quad 2 \quad 1].$$

Hence, the condition

$$\mathbf{v}_i^T \leqslant \mathbf{q}_i^T,$$

is satisfied.

The system state is then the local safe state for resource type R_i.

4. ONE AND MANY TYPES OF RESOURCES

In the case of the single-resource-type system (for example $\mathcal{R} = \{R_i\}$) the condition of the local safe state is the same as of the system safe state. According to this property we can apply the condition (3.13) from Theorem 3.2 for testing the system safe state.

This type of system is identical with the system in [5].

The results obtained here are more suitable for use because of the way the vector $\mathbf{w}_i$ is being constructed.

In the case of the multiple type resource system the existence of the local safe state sequence for any resource type is not sufficient for the existence of the system safe state [1].

The above conditions can be used if the deadlock prevention method is to be used for individual type of resources.

In order to accomplish it, the following theorem is proved in [1] (Theorem 2).

THEOREM 3.3: *The system state is safe if in this state there exist local safe sequences for every resource type R_i $(i = 1, 2, \ldots, m)$ and the sequence of processes:*

(4.1) $$P_{f(1)} P_{f(2)} \ldots P_{f(n)},$$

satisfying:

(4.2) $$\mathbf{y}_{f(1)} \leqslant \mathbf{y}_{f(2)} \leqslant \ldots \leqslant \mathbf{y}_{f(n)}$$

can be found.

It is easy to show that according the Theorem 3.1 the process sequence (4.1) may consist only of the passive and active processes.

The examination of the system safe state using (3.30) for R_i $(i=1, 2, \ldots, m)$ and Theorem 3.3, requires the additional constraints to be imposed on the manner processes receive and release the resources.

These constraints cause that large number of safe states is rejected in the system.

In [1] it is determined that the miniamal number of rejected states (35%) is obtained when matrix **B** is satisfying:

$$\sigma=0, \tag{4.3}$$

where:

$$\sigma=\sum_{i=1}^{m} \sum_{j=1}^{n} |\bar{r}_j-b_{ij}|,$$

$$\bar{r}_j=\frac{1}{m}\sum_{i=1}^{m} b_{ij}.$$

The application of Theorem 3.2 for local safe state examination and prevention of deadlock occurring within different types of resources is expected to give good results.

5. CONCLUSIONS

The paper prevents a fast deadlock avoidance method. Its computational complexity is proportional only to the number of resource types and number of resource unit of each type.

The results obtained here are similar to ones obtained in [5] with simpler construction of the auxiliary vector.

The utilization of this method for multiple type resources system is also proposed.

REFERENCES

1. P. Bak, *A New Method of Deadlock Avoidance in a Multiprocess Multiple Resource Type System*, Podstawy Sterowania, vol. 11, n° 1, 1981.
2. P. Bak, *A Comparative Study of Anti-Deadlock Methods*, Podstawy Sterowania, vol. 7, n° 5, 1976 (in Polish).
3. E. G. Coffman and P. J. Denning, *Operating System Theory*, Prentice Hall, 1973.
4. A. N. Habermann, *Prevention of System Deadlocks*, Comm. of ACM, vol. 12, n° 7, 1969.
5. A. N. Habermann, *A New Approach to Avoidance of System Deadlock*, Revue Française d'Automatique, Informatique et Recherche Opérationnelle, n° 9, B3, 1975.