Complexity of the Deadlock Avoidance Problem

Toshiro Araki
Yuji Sugiyama
Tadao Kasami
Department of Information & Computer Science
Osaka University
Jun Okui
Department of Information Science
Nagoya Institute of Technology

Introduction

Deadlock avoidance is how to grant only "safe" request under the condition that processes declare in advance their anticipated resource requirements. No efficient algorithms for avioding deadlocks among requestors in a general environment are known.

We condider the system which satisfies the following assumptions:

A1 There are T distinct resource types with a number of indistinguishable and interchangeable units of each type in the system.

A2: The system provides two systems macros, ALLOC for requesting resources and DEALLOC for releasing resources. The set of resources allocated to a process is held for exclusive use and they are not preempted by the system until it is explicitly released by the process.

A3: For each process, the flow-diagram, which represents the possible sequences of ALLOC or DEALLOC macros depending upon the control flow of the process, is available at the initialization of the process.

A4: There are no loops in flow-diagrams, but there may be conditional branches in them.

A state of a process p is defined as the flow-diagram of the process p with the indication where the process p has been executed. A state of the system (a state for short) is described by a pair <P, F>, where P is a set of the states of all the processes in the system and F is a set of all the unallocated resources. A state S is said to be safe if and only if there exists a sequence of states starting with S that will be able to fulfill 'worst case' requests. By the deadlock avoidance problem, we mean the following decision problem: "Given a state S, is S safe ?".

At first a polynomial-time-bounded algorithm is presented for the deadlock avoidance problem under the following restrictions.

(R1) The pairs of ALLOC macros to request resources and their associated DEALLOC macros to release the resources are to be well-nested.

(R2) Only one type of resources may be requested by an ALLOC macro.

The number of elementary operations required by the algorithm is bounded by O(UWT + VT + E), where V and E are the total number of nodes (i.e., macros) and that of edges in the flow-diagrams of all the processes in the system, respectively, U is the number of processes in the system, and W is the number of resources allocated.

The following three results are shown by reducing the 3-satisfiability problem to the deadlock aviodance problem.

(1) The deadlock avoidance problem is NP-complete even if the following restrictions are imposed:

(R3) The flow diagram of each process has no conditional branches.

(R4) There is only one unit of each resource type.

(2) The decision problem: "Given a state S, is S not safe ? " is NP-complete even if the following restictions are imposed:

(R1) The pairs of ALLOC macros and their associated DEALLOC macros are to be well-nested.

(R2') At most two types of resources may be requested by an ALLOC macro.

(R4') There are at most two units of each resource type.

(3) When the system provides a macro to request the generation of a resource, the deadlock avoidance problem is NP-complete even if all the restrictions above are imposed.

2. System Model

The system model considered here consists of U sequential processes $p_1, p_2, \ldots, p_U$ and resources of T distinct types $t_1, t_2, \ldots, t_T$ with $a_1, a_2, \ldots, a_T$ units of the respective types. Each resource unit of a given type is indistinguishable from other units of the same type.

The system provides two systems macros, ALLOC for requesting resources with exclusive control and DEALLOC for releasing resources. An ALLOC macro has, as actual parameters, the resource types and the numbers of units of the respective types to be requested. By parameters of a DEALLOC macro, the names of individual resources are specified. These parameters are not changed dynamically.

A process is assumed to be in one of two possible states: (1) active — that is, executing or prepared to execute. (2) waiting for the acquisition of resources which it has requested but which have not as yet been allocated to it. When a process is in "waiting" state, it issues no macro. An ALLOC macro which has been issued by a process p is said to be <u>granted</u> when all the resources requested by the macro are allocated to p. DEALLOC macros are analogous to ALLOC macros. The resources allocated to p are not preempted until they are explicitly released by p. A resource is said to be <u>free</u> if it is not allocated to any process.

3. States

We will introduce a flow-diagram of a process to represent the set of possible sequences of macros issued by the process.

Definition 1. A flow-diagram of a process p is a directed graph with a node corresponding to each macro which is possibly issued by process p, plus one extra node called the initial node. The macro corresponding to node n will be abbreviated as macro n. There is a directed edge from node n to node n' if and only if the macro n' is possibly issued by process p immediately after the macro n (or is possibly the first one issued by process p if n is the initial node). A node corresponding to a macro which is possibly the last one issued by process p is called a final node.

We assume that the flow-diagram of any process p in the system satisfies the following conditions.

(1) There are no directed loops, but there may be conditional branches in a flow-diagram.

(2) Let $\gamma = n_0 n_1 \cdots n_m$ be an arbitrary directed path from the initial node to a final node in the flow-diagram of process p (γ represents a possible sequence of macros issued by process p, that is, a control flow of process p). If the control of process p proceeds along γ, a resource which has been allocated to process p by an ALLOC macro n_i should be released by a DEALLOC macro n_j $(0 < i < j \leq m)$, and, conversly, a resource to be released by n_j has been allocated by n_i.

The condition (2) implies that, for any macro n in process p, the set of resources held by p at the granting time of n (held at n for short) is independent of the previous control flow leading to n of p and, therefore, is unique.

Definition 2. Let r be a resource held by a process p. A scope of resource r in process p is a directed path $n_i \; n_{i+1} \cdots n_j$ in the flow-diagram of process p such that r is held (by p) at n_k ($i \leq k \leq j-1$) and is not at n_j nor at any immediate ancestor of n_i. The nodes n_i and n_j above are called the request and release nodes of r respectively.

Definition 3. A state of a process p is a triple $<G_p, m_p, s_p>$, where G_p is the flow-diagram of p, m_p is the node corresponding to the last macro that has been issued and s_p is either "active" or "waiting" corresponding to active or waiting state of process p, respectively. A state of the system (a state for short) S is a pair <P, F>, where P is $(P_1, P_2, \ldots, P_U)$ in which P_i represents a state of the process p_i, and F is $(F_1, F_2, \ldots, F_T)$ in which F_i represents the number of free units of type t_i when the state of process p_i is P_i for $1 \leq i \leq U$.

Definition 4. A state of a process $<G_p, m_p, s_p>$ is said to be final if and only if m_p is a final node and s_p = "active". A state S = <P, F> is said to be final if and only if all the components of P are final states.

Definition 5. A state S is safe if and only if, for any control flow of each process, there is at least a sequence of allocations and deallocations which leads the system from S to a final state.

By the deadlock avoidance problem, we mean the following problem: "Given a state S, decide whether S is safe or not." We define the four restrictions on system models.

Restrictions.

(1) Single Unit: There is only one unit of each resource type in the system.

(2) Single Parameter: Only one resource may be specified by a macro.

(3) Straight Line: The flow diagram of each process has no conditional blanches.

(4) Nest Structure: For any pair of scopes which are defined on an arbitrary path from the initial node to a final node in any flow-diagram, either they are mutually disjoint or one of the pair includes another.

Let NP be the class of decision problems decidable by nondeterministic polynomially time bounded Turing machines. It is known that the following satisfiability problem with exactly three literals per clause (SAT3) is NP-complete [3]. Let n be a positive integer and $X(n) = \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n\}$. The elements of X(n) are called literals. The complement of x_i (or $\bar{x}_i$) is $\bar{x}_i$ (or x_i).

Problem SAT3. Let n and m be positive integers.

Given : $Q = \langle n, c_1, c_2, \ldots, c_m \rangle$ where $n \leq 3m$ and $c_j \subseteq X(n)$ and $|c_j| = 3$ for $1 \leq j \leq m$.

Question : Does there exist a set $K = \{z_1, z_2, \ldots, z_n\}$ such that $z_i =$ either x_i or $\bar{x}_i$ for $1 \leq i \leq n$ and $K \cap c_j \neq \phi$ for $1 \leq j \leq m$?

If there is such a set K for a given Q, we say that Q is satisfiable.

The following theorem shows that the problem DA, which is the decision problem whose answer is "yes" if and only if a given state is safe, is NP-complete.

Theorem 2. The problem DA is NP-complete even if the "Straight Line", "Single Unit" and "Single Parameter" restrictions are imposed.

(Proof) It is easy to show that the problem DA under the "Straight Line" restriction is in NP.

We will show that the problem SAT3 is reducible to the problem DA above (for the definition of "reducible", refer to [3]). Given an instance of SAT3 $Q = \langle n, c_1, c_2, \ldots, c_m \rangle$, we construct a state of a system (Fig. 1) consisting of 3m+2n+1 processes and 7m+3n+1 resource types which have only one unit. Let elements of c_j be denoted by y_{j1}, y_{j2} and y_{j3} for $1 \le j \le m$. The resource types are denoted by X_i, $\bar{X}_i$, B_i, C_j, C_{jk}, D_{jk} and A where $1 \le i \le n$, $1 \le j \le m$ and $1 \le k \le 3$. The processes are denoted by p_{x_i}, $p_{\bar{x}_i}$, p_{j1}, p_{j2}, p_{j3} and p_0 where $1 \le i \le n$ and $1 \le j \le m$. If $y_{jk} = x_i$ (or $\bar{x}_i$), then $Y_{jk} = X_i$ (or $\bar{X}_i$). Let S be defined as a state in which each process issued only its first macro and the macro has been granted. (In figures, this is represented by the mark "-" under a node.)

Since the size of state S is a linear function of that of given Q, we can obtain Fig. 1 by some deterministic polynomially time bounded Turing machine. Now we will show that S is safe if and only if Q is satisfiable.

If Q is satisfiable, then there exists a set $K = \{z_1, z_2, \ldots, z_n\}$ such that z_i is either x_i or $\bar{x}_i$ for $1 \le i \le n$ and $K \cap c_j \ne \phi$ for $1 \le j \le m$. Since B_i is free, if x_i is in K, then we make the process p_{x_i} proceed until a(A) macro is issued. Otherwise, we make the process $p_{\bar{x}_i}$ proceed until a(A) macro is issued. Then X_i or $\bar{X}_i$ is free. Without loss of generality, let y_{j1} be in K since $K \cap c_j \ne \phi$ for $1 \le j \le m$. Then since D_{j1}, D_{j2}, C_j and Y_{j1} are free, we terminate the process p_{j1}. Next, we make the processes p_{j2} and p_{j3} proceed until a(C_j) macro is issued. Do the operations above for every j ($1 \le j \le m$). Then since C_j and C_{jk} ($1 \le j \le m$, $1 \le k \le 3$) are all free, we terminate the process p_0. Thus A becomes free. As a result, we terminate the process p_{x_i} or $p_{\bar{x}_i}$ that is, X_i and $\bar{X}_i$ ($1 \le i \le n$) become free. Hence we terminate the processes p_{j2} and p_{j3}. Thus, we can terminate the all processes, that is, S is safe.

Assume that Q is unsatisfiable. In the state S, B_i is requested by the processes p_{x_i} and $p_{\bar{x}_i}$ only. We must allocate B_i to either p_{x_i} or $p_{\bar{x}_i}$ so that we terminate the processes p_{x_i} and $p_{\bar{x}_i}$. For each i ($1 \le i \le n$), allocate B_i to either p_{x_i} or $p_{\bar{x}_i}$ and consider the state in which either X_i or $\bar{X}_i$ is released. Define set K as follows. For each i ($1 \le i \le n$), if X_i (or $\bar{X}_i$) is released, then let x_i (or $\bar{x}_i$) belong to K. Since Q is unsatisfiable, there is c_j such that $c_j \cap K = \phi$. If the process p_0 does not terminate, then both p_{x_i} and $p_{\bar{x}_i}$ cannot terminate. For the sake of the termination of the process p_0, all C_j and C_{jk} ($1 \le j \le m$, $1 \le k \le 3$) must be free. Consider three processes p_{j1}, p_{j2} and p_{j3} which correspond to c_j such that $c_j \cap K = \phi$. Since Y_{j1}, Y_{j2} and Y_{j3} are all non-free, if C_j is allocated, then C_j cannot be released. Without loss of generality, assume that C_{j1} is first released among C_{j1}, C_{j2} and C_{j3}. Then in the process p_{j1}, D_{j1} and D_{j2} must be allocated, and if C_j is not allocated, then D_{j1} cannot be released. Thus C_{j3} cannot be released in p_{j3}. Consequently, it is impossible that C_j, C_{j1}, C_{j2} and C_{j3} are all free at the same time, that is, S is not safe. (Q.E.D.)

The following theorem shows that the problem DN, which is the decision problem whose answer is "yes" if and only if a given state is not safe, is NP-complete.

Theorem 3. The problem DN is NP-complete even if the following restrictions are imposed. (1) "Nest Structure". (2) At most two types of resources can be requested by an ALLOC macro. (3) Each resource type has at most two units.

(Proof) If a control flow of each process is chosen, then the question to decide whether there exists a sequence of macros such that all the processes

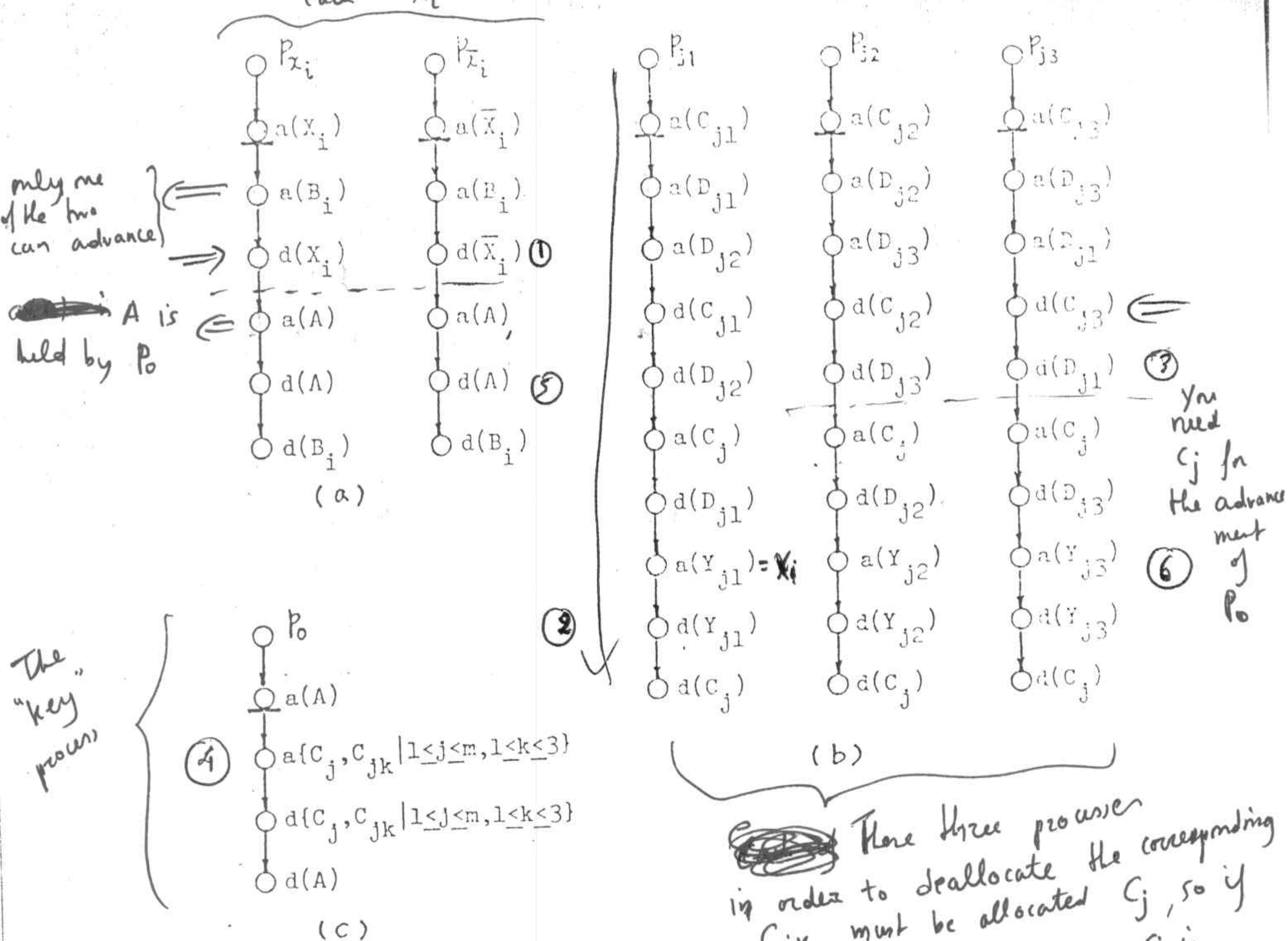

Fig.1 – States of the processes in the proof of theorem 2.

<u>Remark.</u> In figures, $a(X_1, X_2, \ldots)$ means ALLOC macro which requests one unit of each resource type X_i. Similarly, $d(X_1, X_2, \ldots)$ means DEALLOC macro. For simplicity, macros with two or more parameters are used, but they can be rewritten by several macros with one parameter except for $a(X_i, \bar{X}_i)$ in Fig. 2.