

# 消費資源を含むシステムにおけるデッドロック検出

芦原 評† 清水 謙多郎†,☆

特定の故障箇所が存在しないにもかかわらずシステムの永久停止をもたらすデッドロックは計算機システムにおける重要な問題である。デッドロック検出問題は、排他的に使用可能な資源群とそれを要求するプロセス群からなる一般資源システムの状態グラフ還元問題としてモデル化されるが、使用すると消滅する消費資源を含むシステムでのデッドロック検出問題は一般にはNP完全であり、その重要性にもかかわらず注目されることが少なかった。本論文では、消費資源を含むシステムに一定の制限を加えることで、3つのクラスにおいて多項式時間内のデッドロック検出が可能であることを示す。

## Deadlock Detection in Systems with Consumable Resources

HIYO ASHIHARA† and KENTARO SHIMIZU†,☆

Deadlock detection in computer systems can be modeled as a graph reduction problem of general resource systems, which consist of processes, reusable resources and consumable resources. Deadlock detection in reusable resource systems is well known. If all resources in the system are released after use, the problem has polynomial-time solutions. However, the problem is NP-complete in general resource systems with consumable resources, units of which vanish after use. This paper provides three special cases of systems with consumable resources for which polynomial-time deadlock detection algorithms exist. It is shown that those three factors, (1) consumable resources with non-trivial selections in the allocation of currently available units, (2) processes which request more than one unit and (3) resources which wait for more than one process, are all necessary for the problem to be NP-complete.

### 1. はじめに

デッドロックとは、決して起こらない事象を永遠に待つ現象、特に、システムのいくつかの部分が互いを待ち合うことによるそれをいう。デッドロックの検出はその解除をはじめ回避や予防とも密接なつながりを持つ重要な基本問題である。

いくつかの先駆的研究の後、Holt<sup>4)</sup>は、デッドロック解析のためのモデルとして、プロセスと資源からなる一般資源システムを提案した。資源はさらに、メモリ、物理デバイス、あるいはデータベース上のロックなどをモデル化したものである再用資源と、シグナルやメッセージなどのモデル化である消費資源とに分類され、一般資源システム上のデッドロック検出は、シ

ステムの与えられた状態を表現する一般資源グラフの還元問題として定式化される。

しかしながらこの問題は、一般の場合にはNP完全である<sup>3)</sup>。一般資源システムのうち、多項式時間内のデッドロック検出法が知られているのは、消費資源を含まないシステムなどいくつかの特殊なクラスに限られる。デッドロックに関するその後の研究は、これらの原理的に解ける場合についての分散システムへの適用を中心に行われており、消費資源を含むシステム、特に、同一の消費資源に対する複数の要求が競合する場合については、その重要性にもかかわらずほとんど注目されていない。

本論文では、2章において一般資源システムとそれに関する過去の研究を概観した後、3章で、消費資源を含む一般資源システムにおける、デッドロック検出可能な新たなクラスを提示する。

### 2. 一般資源システム

#### 2.1 一般資源グラフ

一般資源システムは、それぞれ有限個の、プロセス、

† 電気通信大学情報工学科

Department of Computer Science, The University of Electro-Communications

☆ 現在、東京大学農学生命科学研究科応用生命工学専攻

Presently with Department of Biotechnology, The University of Tokyo

再用資源および消費資源からなる。各資源はそれぞれ非負整数個の独立なユニットを持ち、各ユニットはいずれかのプロセスにより排他的に使用される。プロセスはいくつかの資源に対しそれぞれ一定数のユニットを要求し、そのすべてを獲得するまで封鎖（停止）する。

再用資源のユニット数はつねに一定であり、各ユニットは、いずれか1つのプロセスに占有されているか、自由であるかである。ある再用資源に対し、1つのプロセスが要求しているユニット数と占有しているユニット数の和は全ユニット数以下である。自由ユニットはすべて同等である。自由ユニットは、その資源を要求しているプロセスに獲得されることができる。これにより要求は占有に、獲得されたユニットは占有状態に変化する。プロセスが要求したすべてのユニットを獲得すると、プロセスは封鎖状態を抜けて稼働し、占有していた再用資源のユニットをすべて解放する。要求および占有は消滅し、解放されたユニットは自由ユニットに戻る。すべてのユニットが自由である再用資源は自由であるという。

一方、消費資源のユニット数は可変であり、存在するユニットはすべて自由である。消費資源のユニットはそれを要求するプロセスに獲得されると要求とともに消滅する。一方、各消費資源には、少なくとも1つの生産者と呼ばれるプロセスが対応し、生産者は封鎖していなければ必要な数のユニットを生産する。すなわち、消費資源に対する要求は、その資源の生産者のうち少なくとも1つが稼働しているならば、すべて満たされる。生産者が稼働している消費資源は自由であるという。

したがって、プロセスと資源の間には、(1)プロセスが資源に対し要求を行っている、(2)プロセスが再用資源のユニットを占有している、(3)プロセスが消費資源の生産者である、の3種類の関係が存在する。封鎖しているプロセスは資源に対しユニットの獲得を待ち、自由でない資源はプロセスに対しユニットの解放または生産を待っている。ある時点におけるシステムの状態はこれらの関係の総和であり、一般資源グラフの形で表現することができる。図1に、一般資源グラフの例を示す。方形がプロセス、大円が資源、小円がユニット、各辺（矢印）が、その元の要素が先の要素を待っていることを表す。要求、占有、生産の各関係を意味する辺を、それぞれ要求辺（プロセスから資源への矢印）、占有辺（再用資源ユニットからプロセスへの矢印）、生産辺（消費資源からプロセスへの矢印）と呼ぶ。すなわち、このグラフは以下のことを

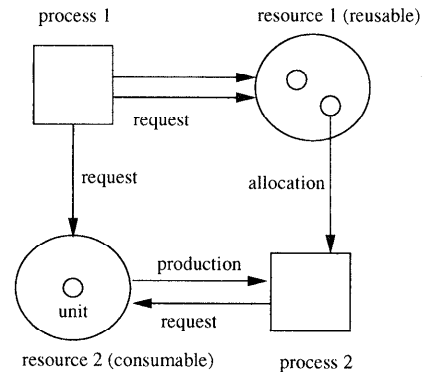


図1 一般資源グラフの例

Fig. 1 Example of a general resource graph.

意味する；プロセス1は再用資源1に2ユニットを、消費資源2に1ユニットを要求している。プロセス2は再用資源1の1ユニットを占有し、消費資源2に1ユニットを要求している。再用資源1は2ユニットを持ち、そのうち1つが自由である。消費資源2は1ユニットを持ち、その生産者はプロセス2である。

## 2.2 割当てとグラフ還元

ある状態のシステムがグラフとして与えられたとき、一連の割当てによりグラフを還元することができる。

割当てとは、グラフ上のある1つの要求辺について、要求されている資源に自由ユニットが存在する場合に行われる以下の操作である。

- (1) 要求が消費資源に対するものであるなら、その資源の自由ユニット1つと要求辺を同時に除去する。
- (2) 要求が再用資源に対するものであるなら、その資源の自由ユニット1つと要求辺を、占有ユニット1つと、そのユニットから要求を出していたプロセスへの占有辺に変更する。
- (3) 初期状態において、あるいは割当て操作の結果、要求辺を持たないプロセスが存在するなら、そのすべての占有辺を除去し、対応する占有ユニットを自由ユニットに変更する。また、そのプロセスが生産する各消費資源に十分な数のユニットを置き、生産辺を除去する。

(1), (2)はプロセスによる資源の獲得（ユニットの要求への割当て）、(3)はプロセスの稼働による資源の解放ないし生産に対応する。以下では、(3)は、あるプロセスの全要求が満たされた時、暗黙のうちに行われるものとして、(1), (2)の一部と考える。

すべての辺を除去されたプロセスおよび資源（稼働しているプロセスおよび自由な資源）は他要素との関係を持たず、実質的にグラフから除去できる。還元と

は、グラフ上において、あるプロセスが一連の割当てによって全要求を獲得し、資源を解放ないし生産してすべての辺を除去されることをいい、実際のシステムにおけるプロセスの稼働による待ちの解消を表現するものである。グラフのすべての辺を除去することを完全還元、すべての辺が除去された状態を完全還元状態と呼ぶ。

割当ての順序集合、すなわち、グラフ中の要求辺の部分集合に順序をつけたものであって、その順序によって(1)または(2)が実行可能である(実行時に自由ユニットが存在する)ものを割当て列と呼ぶ。初期状態から完全還元状態に至る割当て列、すなわちすべての要求辺を含む割当て列を完全割当て列と呼ぶ。

一般資源グラフは、ある割当て列の適用により完全還元されることもあるが、いくつかのプロセスが、それ以上の割当てができない状態で封鎖したまま残ることもありうる。これがデッドロックである。このどちらになるかは、最初に与えられたグラフと、割当ての手順の双方に依存する。グラフに表現されたあるシステムの状態は、完全還元されているか、または少なくとも1つの完全割当て列によって完全還元可能などとき安全であるといい、完全割当て列が存在しない、すなわちいかなる手順によっても必ずデッドロックに陥るとき非安全であるという。また、与えられた状態に対するある割当てが、安全な状態を非安全な状態に変えることがないとき、その割当てを安全であるという。一方、いかなる割当ても、非安全な状態を安全な状態に変えることはない。非安全状態における割当てはすべて安全であるとする。

あるプロセスがすべての要求を一時に獲得するような割当て列を一括割当てと呼ぶ。完全還元でない安全な状態においては少なくとも1つの安全な一括割当てが可能であり、完全割当て列が存在するならばそのうちの1つは一括割当てのみによるものである。

図1のグラフは、資源2のユニットをまずプロセス2の要求に割り当てることで完全還元可能であり、したがって安全であるが、このユニットを先にプロセス1の要求に割り当ててしまうとデッドロックに陥る。

デッドロック検出とは、与えられた状態が安全か否かを判定する問題である。この判定は、すべての可能な割当て列を尽くすことにより可能であり、かつ、結果が安全の場合には実際に完全割当て列を示すことができるため、たかだかNP問題である。しかしこの方法は、割当て列の数がプロセス数に対して指数関数的に増大するため、ある程度以上大きなシステムでは現実的でない。よって、本論文においてデッドロック検

出可能というときは多項式時間内での判定が可能であることを意味するものとする。

### 2.3 一般資源グラフと現実のシステム

現実の計算機システムにおいては、各プロセスは資源の要求、獲得、占有、解放、生産を動的に繰り返しつつ独自に走行するが、一般資源グラフは、このうちある瞬間におけるシステムの状態をモデル化したものである。

Holtの原定義においては、一般資源グラフ上の各辺は時刻 $t$ のシステム上で現に行われプロセスを封鎖させている要求、 $t$ において行われプロセス稼働後に解放される占有および同じく稼働後に行われる可能性のある生産を意味する。この場合、すべての要求が満たされることは資源の解放や生産の必要条件であるから、システムの安全性を楽観的に解析することができる。すなわち、完全還元できないグラフに対応するシステムは必ずデッドロックするが逆は真でない。

一方、 $t$ 以降におけるプロセスの動作が既知のとき、 $t$ におけるシステムに対応する一般資源グラフを以下のように構成することにより、 $t$ 以降に行われる要求も含めた悲観的解析ができる。

実際のシステム上のプロセス $P$ に対応するグラフ上のプロセスを $G(P)$ 、システム上の資源 $R$ に対応するグラフ上の資源を $G(R)$ とする。各 $P$ について、予測される動作シーケンス上の1点を定め、これを時刻 $t'_p(>t)$ とする。さらに各 $R$ について、

- (1)  $t$ における $P$ による再用資源 $R$ の占有であって、 $t'_p$ においては解放されているものを、グラフ上で $G(P)$ による $G(R)$ の占有とする。 $t$ において $P$ が占有して $t'_p$ においても占有している資源(ユニット)は、グラフ上ではまったく存在しないものとして扱う。
- (2)  $t$ から $t'_p$ までの間に $P$ が $R$ に対して行う要求( $t$ において現に行っているものを含む)を、グラフ上で $G(P)$ の $G(R)$ に対する要求辺とする。ここで、 $R$ が消費資源のときは $P$ による要求の延べユニット数を、再用資源のときは $\langle P$ が一時に占有しうる $R$ のユニットの最大数 $\rangle - \langle t$ において $P$ が占有している $R$ のユニット数 $\rangle$ を、それぞれ $G(P)$ の $G(R)$ に対する要求辺数とする。すなわち、期間中いったん解放した再用資源を再度要求する場合には、要求数は累計されない。
- (3)  $t$ から $t'_p$ までの間に $P$ が消費資源 $R$ の十分なユニットを生産するならば、 $G(P)$ を $G(R)$ の生産者とする。

ただし、 $P$  は、 $t$  において占有していない再用資源を、 $t_p$  において占有していることはないものとする。すなわち、 $t$  から  $t_p$  の間に要求・獲得された再用資源は、 $t_p$  の時点においてはすべて解放されていなければならない。この条件の範囲で、各  $P$  について  $t_p$  を自由に定めてよい。時刻はプロセスごとの論理的時間上で決定されるので、実行時の実時間において異なる  $t_p$  がグラフ上で共存しても矛盾は生じない。プロセスは消滅するまでには占有していたすべての資源を解放すると考えられるので、そのような  $t_p$  が1つは存在する。この、プロセス消滅の時点をとる  $t_p$  にとるのが容易であるが、上記条件が最初に満たされた後、次に新たな要求が発生する直前に  $t_p$  をとれば後述するような解析がより細かい粒度で行える。特に、プロセスが、各部分において要求した全資源を終了までに解放するような部分の集合の入れ子としてあらかじめ構成されているならば、 $t$  を含む最小の入れ子の終了時に  $t_p$  をとればよい。

制御流の分岐などにより、プロセスの動作が一意に決定できない場合は、最も悲観的な予測（最大の要求、最小の生産と解放）に基づいてグラフを構成する。

このとき、各資源について  $G(P)$  の要求辺以上の数の自由ユニットが与えられるならば、 $P$  は  $t_p$  までの動作を封鎖することなく実行可能である。したがって、実際にはプロセスの走行の過程で  $t$  から  $t_p$  までの間にある順序で行われる動作について、モデル上は、任意の順序で要求を獲得し、すべての要求が満たされると同時にすべての解放ないし生産が行われるものと見なすことができる。ゆえに、完全還元可能なグラフに対応するシステムは、実際にもデッドロックすることなく動作可能である。逆は必ずしも真でない。

悲観的解析と楽観的解析は状況に応じて使い分けあるいは組み合わせることができる。特に、プロセスの動作が決定的、かつ要求と解放・生産が2相であり  $t$  以降  $t_p$  までに新たな要求が発生しないならば両者は一致する。以下では、特に混乱のおそれがない場合、システムの状態とそれに対応するグラフを区別せずに論ずる。

プロセスの連続的な動作を扱うには、このモデルを繰り返し適用すればよい。すなわち、新たな資源要求や獲得、占有、解放、生産によってシステムの状態が変化すると、その時点で新たな一般資源グラフが構成されるものとする。必ずしもシステム全体が完全還元の状態になった後に新たな動作に移るとは限らないことに注意する。安全性とは、デッドロックに陥ることなく作動する手順がその時点で少なくとも1つ存在す

ることを保証するものであって、実際のシステムがそのとおりの順序で動く必要はない。

## 2.4 デッドロック検出可能な既知のシステム

現実の計算機システムの表現として、一般資源システムは完全なものではなく、共有資源、有限生産および部分要求<sup>3)</sup>、制御の分岐<sup>8)</sup>などいくつかの拡張も提案されている。しかしながら、一般資源システムにおけるデッドロック検出問題がすでにNP完全であるため、実際の多くの研究はむしろモデルに制限を加える方向で行われている。

制限の1つは消費資源を含まないシステムであり、これを再用資源システムと呼ぶ。再用資源においては、獲得されたユニットはそのプロセスの稼働により解放されるため、一括割当てはつねに安全である。このことから、再用資源システムにおいてはデッドロックは検出可能である<sup>4)</sup>。

また、システム内に、要求を受けかつ自由ユニットを持つような資源が存在しない状態を即時割当て状態といい、つねに即時割当て状態にあるシステム、すなわち自由ユニットが存在する限りすべての要求がただちに満たされるシステムを即時割当てシステムと呼ぶ。たとえば、1自由ユニットを持つ資源があり、2つのプロセスからそれぞれ2個の要求があるとき、その1つのユニットをいずれかに割り当ててしまう場合と最低2個の自由ユニットになるまで割当てを保留する場合が考えられるが、即時割当てシステムではつねに前者をとり、保留を行わない。

再用資源システムや即時割当てシステムでは、一般資源グラフはより簡略な待ち条件グラフの形に書き換えることができ、デッドロック検出が理論上容易であると同時に応用においても重要である。

さらに、各資源がそれぞれただ1つのユニットしか持たない再用資源システムを単資源システムと呼ぶ。また、各プロセスが一時にはただ1つのユニットしか要求しないシステムを単要求システムと呼ぶ。単資源システムにおいては、一般資源グラフに周（待ちの連鎖の円環）が存在することが、即時割当て状態の単要求システムにおいては、一般資源グラフに結び（プロセスと資源の集合であって、各要素がその集合自身の他の要素に対し、かつそれらにのみ、待ちの経路を持つもの）が存在することが、それぞれデッドロックの必要十分条件であり、いずれの場合もデッドロック検出可能である。なお、一般に、周はすべての一般資源システムにおけるデッドロックの必要条件、結びは単要求システムにおける必要条件であるとともに即時割当てシステムにおける十分条件である<sup>4)</sup>。

デッドロックに関するその後の研究は、主としてこれらデッドロック検出可能の場合についての分散システムへの応用<sup>9)</sup>に向けられてきた。分散システム上では効率が特に重要であり、また独自の問題も発生するため、多くの研究が現在も続けられている<sup>1),6)</sup>が、本質的には分散デッドロック問題における AND モデル<sup>7)</sup>は単資源システム、OR モデル<sup>2)</sup>は単要求即時割当てシステム、AND-OR モデル、 $k/n$  モデル<sup>5)</sup>は再用資源システムに相当する。

一方、消費資源を要求するプロセスがつねにその資源の生産者でもあるようなシステム<sup>3)</sup>などわずかな例を除き、即時割当て状態でない消費資源を含むシステムのデッドロックについては知られていない。

### 3. 消費資源を含むデッドロック検出

本章では、即時割当て性、単資源性、単要求性の概念をそれぞれ拡張した非選択消費性、非 OR 性、非 AND 性を定義し、この3つの性質の少なくとも1つを満たすならば、消費資源を含むシステムにおいてもデッドロックが検出可能であることを示す。

#### 3.1 NP 完全性に関わるグラフの性質

プロセスと資源の関係が1対1の自明な場合以外に、一般資源グラフでは、(1) 1 プロセスが複数の占有/生産辺を持つ、(2) 1 資源が複数プロセスからの要求辺を持つ、(3) 1 資源が複数プロセスへの占有/生産辺を持つ、(4) 1 プロセスが複数の要求辺を持つ、の関係が可能である。これらのうち、(1) を欠く場合、すなわちプロセスがただか1つのユニットまたは1つの資源しか解放または生産しない場合については一般性を失わない(任意の一般資源グラフはこの形に置換え可能)。

(2) により複数の要求を受け、かつ自由ユニットを持つ資源では、そのユニットをどの要求に割り当てるかの選択が生じる。再用資源では一括割当てを選べばよいが、消費資源においてはこの選択は自明でない。消費資源が自由ユニットを持ち、しかも複数のプロセスから、それぞれのプロセスについては自由ユニット数以下だが合計するとそれを超える数の要求を受けている状態を、**選択消費**と呼ぶ。一般資源システムにおいて、デッドロックが割当て手順に依存する(非安全な割当てが存在する)ことは、本質的には選択消費に由来し、このときプロセスの待ち条件は限定された形の NOT 条件を含む。選択消費が存在しないことを非選択消費性と呼ぶ。再用資源システムおよび即時割当てシステムは非選択消費性を持つ。

一方、プロセスは、すべての要求が満たされるまで

封鎖し、自由ユニットは、その生産者または占有プロセスのうちいずれか1つが稼働すれば発生する。したがって、一般資源グラフにおいて、(3) の多プロセスを待つ資源は OR 分岐、(4) の多ユニットを待つプロセスは AND 分岐を意味し、プロセスが資源を經由して他のプロセスの稼働を待つ条件には一般には AND と OR の任意の組合せが可能である。プロセスが、複数の資源を OR 条件によって待つことを許す場合についても、中間プロセスを仮定することによりこのモデルで扱うことができる。

単資源システムでは、再用資源のユニットを占有するプロセスもただか1つなので、(3) の分岐がなく、AND 分岐のみが存在する。しかし、再用資源においてユニットが複数であっても、それらのうち自由でないものを占有するプロセスがすべて同一であるなら、また、消費資源においては生産者がただ1つであるなら、同様の性質を持つ。本論文ではこの性質を非 OR 性と呼ぶ。

単要求システムでは(4)がないので OR 分岐のみを含むが、単要求システム以外で、プロセスが形式的に複数の要求辺を持つ場合でも、要求している資源が1種類のみであり、かつその資源を解放ないし生産しうるプロセスのうち任意の1つの稼働により要求がすべて満たされるなら、同様である。これを非 AND 性と呼ぶ。

図2に、Gold<sup>3)</sup>によるデッドロック検出不可能な一般資源グラフを示す。 $q$  個の再用資源が各  $p$  個のユニットを持ち、2 個ずつ  $p$  組になったプロセスが各組  $q$  個のユニットを占有している。この計  $pq$  個の占有関係の任意の組合せ(破線部)に対するデッドロック検出は、NP 完全問題となる。

図2は、選択消費、AND分岐およびOR分岐を含んでいる。以下では、NP完全性のためにはこの3要素のすべてが必要であることを示す。ただし、この例から分かるように、消費資源自身およびその周囲に非 OR 性、非 AND 性があっても、システム全体がそうでなければ一般にはデッドロック検出は保証されない。

#### 3.2 選択消費を持たないシステム

選択消費を持たない消費資源は、要求数がユニット数以下であるプロセスからの要求すべてを満たしうる。したがって、要求数の少ないプロセスから順にユニットを割り当てることにより、安全に即時割当て状態に変換できる。

即時割当て状態の消費資源は、ユニットを持たないか、要求を受けていないかであり、このうち要求を受けていない資源についてはただちに無視できる。

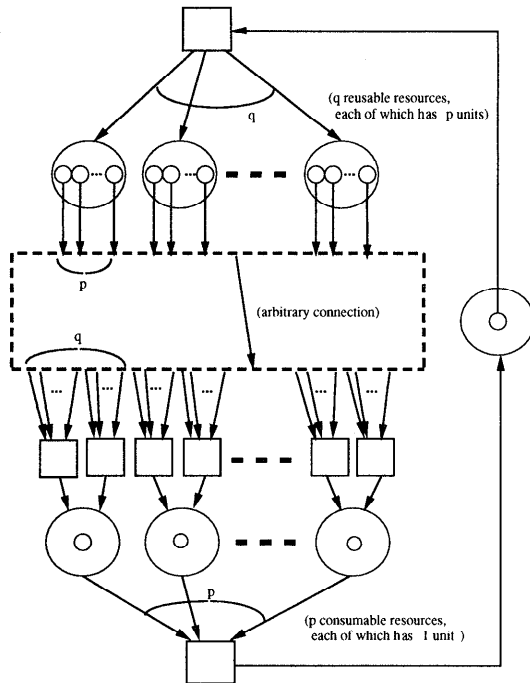


図2 デッドロック検出問題がNP完全なシステム  
Fig. 2 A system in which deadlock detection is NP-complete.

現在ユニットを持たない消費資源は、以下の手順により安全性を保存しつつ reusable 資源に置き換えることができる（これはグラフ上で行われる書換え操作であって、現実のシステムにおいて消費資源が reusable 資源に変わることを意味しない）。

- (1) その資源に対して行われている要求の総数を  $n$  とする。
- (2) その資源の生産者の数を  $m$  とする。
- (3)  $nm$  個のユニットを持つ reusable 資源に置き換える。
- (4) 生産者であった各プロセスに、 $n$  個ずつのユニットを占有させる。

プロセス数を  $p$ 、資源数を  $r$  とするとき、任意の資源の種別とユニット数、任意のプロセスから資源への要求辺数、および任意の資源から任意のプロセスへの占有辺数ないし生産辺の有無をいずれも  $O(1)$  で参照・更新できるとして、1つの資源について(1)、(2)、(4)はそれぞれ  $O(p)$ 、(3)は  $O(1)$  で行える。したがってこれを各資源について行えば、この置換えは  $O(pr)$  の多項式時間内に行われる。

また、元の消費資源、置き換えた reusable 資源のいずれにおいても、要求はすべて満たされるか1つも満たされないかのいずれかであり、任意の生産者（占有者）が稼働することがすべての要求が満たされる必要十分

条件であるから、この reusable 資源はデッドロック検出に関して元の消費資源と同等である。

以上により、一般資源システムにおいて、選択消費が存在しないならばデッドロック検出可能である。

### 3.3 OR 分岐を持たないシステム

各資源が、

- (1) reusable 資源であるならば、そのユニットを現在占有しているプロセスがただか1つ、
- (2) 消費資源であるならば、その生産者がただか1つ（ただし、生産者を持たない消費資源は自由であるとする）

のとき、一般資源グラフは AND 分岐のみを含む。前章であげた単資源システムはこの特殊な場合に当たる。このようなシステムにおいて、選択消費が存在してもデッドロックが検出可能であることを示す。

#### 3.3.1 消費資源システム

まず、 reusable 資源を含まないシステム（消費資源システム）について論ずる。与えられた状態に対して、以下の3種の操作を考える。

- (1) ある資源が、すべての要求を満たすことができる（要求辺以上の数のユニットを持つ）場合、その割当て。
- (2) 要求を受けていない資源の生産辺の除去。
- (3) 生産辺を持たないプロセスの要求辺の除去。

これらの操作を、それ以上どれもできなくなるまで任意の順序で繰り返す。このとき次が成り立つ。

**定理1** 操作(1)~(3)の繰返しによる最終状態が完全還元であるか否かは一意に定まり、完全還元するとき、かつそのときのみ、非 OR 消費資源システムは安全である。

(証明)

(A) : [(1)~(3)により完全還元できる  
→システムが安全である]

操作(1)は、2.2節で示した割当て操作の1つである。安全性とは、グラフが割当て操作の繰返しによって完全還元可能なことであるから、(1)の操作によって完全還元可能であるシステムは、安全である。また、(1)を行った結果が安全であるシステムも、安全である。

要求を受けていない資源の存在は実質的に他に影響しないため、除去しても非安全状態が安全になることやその逆はない。すなわち、(2)を行ったシステムが安全であるとき、かつそのときのみ、行う前のシステムも安全である。

同じく、生産待ちを受けていないプロセスは、稼働させても益がなく、また、システムが安全であるなら

まずこれを取り除いたグラフを完全還元した後に稼働させることができるので、このプロセスの要求は実質的に無視できる。すなわち、(3)を行ったシステムが安全であるとき、かつそのときのみ、元のシステムも安全である。

以上により、(1)~(3)により完全還元されるならば、最初のシステムは安全である。

ただし、(2)、(3)はこのままでは2.2節に示された割当て操作ではないことに注意する。システムが(1)~(3)のある順序での繰返しによって完全還元の状態になるとき、まず(1)の割当てのみをその順序で行い、その後プロセスの稼働・資源の生産を行いつつ(3)を逆順で行う、すなわち上記判定手順において最後に(3)によって除去した要求辺を最初に除去するように割当てを行うことによって、実際の完全割当て列を得る。

**(B)** : [システムが安全である

→(1)~(3)により完全還元できる]

(1)の割当てには選択の余地がないので、安全なシステムが(1)によって非安全になることはない。(2)、(3)の操作も、すでに述べたように安全性に関して同値な変形である。ゆえに、安全なシステムに対して(1)~(3)を繰返した結果は、その順序によらずつねに安全である。

完全還元でない状態において、(1)~(3)のいずれも行えない、すなわち

- (a) 要求を受けている資源へはユニット数より多い要求辺がある、
- (b) 自由でない資源へは少なくとも1つ要求辺がある、
- (c) 封鎖しているプロセスからは自由でない資源への生産辺がある

とする。(a)でないなら(1)、(b)でないなら(2)が、(c)でないなら(3)が可能であるので、(1)~(3)によって完全還元されないなら最終的にこの状態になる。この状態が安全でない、すなわち、その後残っている自由ユニットをどのように割り当ててもデッドロックすることをいえばよい。

完全還元でないので、少なくとも1つの封鎖しているプロセスが存在する。このとき、すべての自由ユニットを任意に割り当てる。システムは即時割当て状態になる。これにより稼働するプロセスがあるかもしれない。しかし、(a)により、割り当てられなかった要求が1つは存在し、したがって封鎖しているプロセスが1つは存在する。(c)により、そのプロセスの稼働を待つ資源が1つは存在する。(a)および(b)によ

り、その資源に対する要求が1つは満たされずに残っている。非OR性により、他のプロセスによりこの資源が生産されることはない。以下、自由でないすべての資源は少なくとも1つの要求をプロセスから受け、封鎖しているすべてのプロセスは少なくとも1つの生産待ちを資源から受けている。これによる待ちの連鎖を考えると、プロセス数は有限であるから、周が形成される。システムは現在即時割当て状態であり、生産者が単一で即時割当て状態の消費資源とユニットが単一の再用資源とは前節の議論により同等であるから、デッドロックに陥る。以上は最初のユニット割当てがいかなるものであっても成立するので、最終状態が完全還元でなければ非安全である。安全性は(1)~(3)によって失われていないので、この場合最初の状態が非安全である。

これにより題意の対偶が示された。また、(1)~(3)のある順序の適用によって完全還元でない最終状態に達するなら、システムは安全でないので、いかなる順序によっても完全還元はできない。すなわち、結果は、(1)~(3)の適用の順序によらず定まる。(証明終)

プロセス数  $p$ 、資源数  $r$  に対し、(1)、(2)を満たす資源は  $O(pr)$  回の参照で発見可能であり、(3)も  $O(pr)$  で行える。この繰返しを  $r$  回行うまでには最終状態に達するので、全体の計算量はたかだか  $O(pr^2)$  となる。

以上の手順とその計算量は上界を示したのみで、特に効率化を図ってはいない。

### 3.3.2 一般資源システム

OR分岐を持たない一般資源システムにおいて、すべての再用資源を消費資源に置き換える。再用資源のユニットを占有しているプロセスがあれば、それを置き換えた消費資源の生産者とする。元の再用資源が  $n$  個の自由ユニットを含み、その資源に対する要求数が1以上  $n$  以下であるプロセスが  $m$  個存在するとき、置き換えた消費資源には、 $m$  個のユニットを置く。元の要求数が1以上  $n$  以下であったプロセスからは1個の、 $n+1$  以上であったプロセスからは  $m+1$  個の要求を、それぞれ消費資源に対して行う。以上の置換えは各資源について  $O(p)$ 、全体で  $O(pr)$  で終了し、またこれにより非OR性は失われないので、得られた消費資源システムに対し前節の手順を適用する。

**定理 2** 上記手順による置換えで得られた消費資源システムが安全であるとき、かつそのときのみ、元となった非OR一般資源システムは安全である。

(証明)

(A) : [置換え後の消費資源システムが安全

→元の非 OR 一般資源システムが安全]

消費資源システムが安全であり、完全還元の手順が存在したとする。すなわち、一括割当てによる完全割当て列が存在する。一般資源システムに対してこれと同一の順序でプロセスを還元する一括割当て列を適用する。

このとき、再用資源を置き換えた消費資源において要求が獲得可能であるならば、要求数が1のみであるか消費資源が自由であるかであり、元の再用資源においても要求数が自由ユニット数以下であるか再用資源が自由であるかである。また、再用資源においては、獲得されたユニットはそのプロセスの稼働により解放されるため、それぞれのプロセスからの要求が個々に満たされるならば、すべてが同時に満たされると考えてよい。すなわち、消費資源システムにおいてあるプロセスからある資源への全要求のユニットが獲得可能ならば、元のシステムの対応する再用資源においても獲得可能である。ゆえに、消費資源システムが安全ならば、元のシステムも安全である。

(B) : [元の非 OR 一般資源システムが安全

→置換え後の消費資源システムが安全]

元の再用資源に対する要求のうち、資源を占有しているプロセスの稼働なしに要求を出すプロセスを稼働させるものについては、置き換えた消費資源においても十分な数のユニットが存在するのですべて要求を満たす。また、それ以外の要求は、生産者(元の占有者)となったプロセスが稼働すればすべて満たされる。ゆえに、元のシステムが安全ならば、置き換えた消費資源システムも安全である。 (証明終)

### 3.4 AND 分岐を持たないシステム

各プロセスについて、現在要求している資源がただか1つ(1種類)であり、かつそれが

- (1) 消費資源であるか、
- (2) 再用資源であって、要求を満たすのに不足しているユニット数(要求数)-(自由ユニット数)が、現在その資源の1プロセスにより占有されているユニット数の最小値以下である

ならば、一般資源グラフはOR分岐のみを含む。このようなシステムにおいて、選択消費が存在してもデッドロックが検出可能であることを示す。

#### 3.4.1 消費資源システム

この場合もまず、消費資源のみからなるシステムについて論ずる。非AND消費資源システムに対し、以下の3つの操作を適用する。

- (1) ある資源が、あるプロセスからの要求をすべて満たすだけのユニットを持ち、かつそのプロセスが一般資源グラフ上で周の一部であるとき、そのプロセスにユニットを割り当てる。
- (2) ある資源が、少なくとも1つのプロセスからの要求をすべて満たすだけのユニットを持ち、かつそのようなプロセスのいずれも、一般資源グラフ上で周の一部でないとき、そのようなプロセスの1つにユニットを割り当てる。
- (3) ある資源が、すべての要求を満たすだけのユニットを持つとき、それらの要求にユニットを割り当てる。

いずれの場合も、要求したユニットを獲得したプロセスは、そのプロセスが生産者である資源を生産する。

**定理3** 操作(1)~(3)の繰返しによる最終状態が完全還元であるか否かは一意に定まり、完全還元するとき、かつそのときのみ、非AND消費資源システムは安全である。

(証明)

(A) : [(1)~(3)により完全還元

→システムが安全]

(1)~(3)はいずれも実際に可能な割当て操作であるから、これにより完全還元されるシステムは、安全性の定義により、安全である。

(B) : [システムが安全

→(1)~(3)により完全還元]

システムが安全であり、かつ完全還元でないなら、なんらかの割当て操作によりいずれかのプロセスを稼働させることが可能である。すなわち、少なくとも1つの資源について、その資源を要求するプロセスのうち少なくとも1つの要求をすべて満たすことができる。したがって(1)または(2)のいずれかが可能である。

ここで、(1)、(2)がいずれも安全な割当てであるならば、割当て後のシステムも安全であるから、これを繰り返すことにより完全還元される。以下、(1)、(2)が安全であることをいう。割当て(3)は便宜上加えるものであり、選択がないので明らかに安全である。

(1)の割当てを行うと、そのプロセスはただちに稼働する。なぜならそれはそのプロセスからの唯一の要求だからである。次いで、(3)により、このプロセスが生産する資源を要求しているプロセスのすべてを稼働させることができる。この過程を繰り返し、周の順序に従って最終的に最初に割り当てた資源が生産されるので、その資源への他のプロセスからの要求もすべて満たされる。このとき、途中で経過したプロセスはすべて稼働し資源はすべて自由となる。したがってこ



の過程により還元可能性が失われることはなく、最初の割当ては安全である。

一方、ある資源に関し(2)の割当てが可能の場合、その資源はいかなる周の一部でもないので、どの選択を行っても無関係である。すなわち、他のなんらかの操作によりその資源が生産されるならすべての要求は満たされ、そうでないならデッドロックに陥るが、そのいずれであるかの決定に(2)の割当ては直接的にも間接的にも影響を持たない。ゆえに、(2)のどの選択も安全である。

よって、安全なシステムは(1)~(3)の割当てのみによって完全還元できる。また、各割当ての安全性により、最終状態が完全還元であるか否かは一意に定まる。

(証明終)

プロセス数  $p$ 、資源数  $r$  に対し、あるプロセスを含む周の検出は  $O(pr^2)$  で可能である。これをすべてのプロセスについて行えば、他の部分はそれ以下のオーダーで計算できるので、全体の判定を  $O(p^2r^2)$  で行うことができる。

### 3.4.2 一般資源システム

単要求システムは非 AND 性を持つ。逆に、再用資源に対する非 AND 性を持つ要求は、以下の手順により単要求に置換え可能である。

- (1) 各プロセスについて、再用資源に対する要求が
  - (a) その資源の自由ユニット数以下であるなら、要求を取り除く。
  - (b) そうでないなら、複数の要求を1つに置き換える。
- (2) 自由ユニットをすべて取り除く。

非 AND 性を持つシステムにおいて、再用資源に対し要求を行っている各プロセスが稼働しうると必要十分条件は、対象となる資源がすでに要求数以上の自由ユニットを持つか、またはその資源のユニットを現在占有しているプロセスのどれか1つが稼働することである。また、定理2の証明(A)において述べたように、再用資源においては各プロセスを独立に扱ってよい。ゆえに、上記の置換えによって得られたシステムはデッドロック検出に関して置換え前のシステムと同等であり、置換えは  $O(pr)$  で行える。以下では再用資源に対する要求は単要求であり、また、再用資源は自由ユニットを持たないものとする。自由ユニットが1つでも存在するならば、単要求かつ再用であるから、それを順に割り当てることでその資源への要求はすべて満たされるので、ただちにこれを取り除くことができる。

さらに、非 AND 一般資源システムにおいて、再用

資源をユニットを持たない消費資源に置き換える。ユニットを占有している各プロセスが生産者になる。この置換えも  $O(pr)$  で十分であり、またこれにより非 AND 性は失われないので、得られた消費資源システムに対し前節の手順を適用する。

**定理4** 上記手順による置換えで得られた消費資源システムが安全であるとき、かつそのときのみ、元となった非 AND 一般資源システムは安全である。

(証明)

(A): [置換え後の消費資源システムが安全]

→元の非 AND 一般資源システムが安全]

ユニットを持たない消費資源に対する要求が、生産者の稼働以前に満たされることはない。消費資源の生産者の1つが稼働するならば、対応する再用資源のユニットを占有しているプロセスの1つが稼働しユニットを解放するので、自由ユニットが生じ、すでに述べたようにすべての要求が満たされる。したがって消費資源でユニットを獲得可能な要求は再用資源でも獲得可能であり、置換え後のシステムが安全なら元のシステムも安全である。

(B): [元の非 AND 一般資源システムが安全]

→置換え後の消費資源システムが安全]

自由ユニットを持たない再用資源に対する要求が、その資源のユニットを占有しているプロセスが稼働する以前に満たされることはない。また、占有プロセスの置換えである生産者の1つが稼働した後は、消費資源に対するすべての要求が満たされる。したがって、あるプロセスから元の再用資源への要求が満たされるならば置換えた消費資源においても同様であり、したがって元のシステムが安全なら置換え後のシステムも安全である。

(証明終)

## 4. む す び

本論文では、一般資源システムに一定の制限を加えることで、消費資源を含むシステムにおいてもデッドロック検出が可能であることを示した。提示した手順はすべて、組合せ的な試行錯誤をともしないもので、多項式時間内に終了する。与えられた制限を持つシステムは、いずれも理論的興味と同時に応用上も重要な性質を持つものである。今後の研究課題として、検出アルゴリズムの効率化、分散化など実際のシステムへの応用のほか、デッドロック検出可能なシステムの条件と計算量のより細かい決定や、部分要求などモデルの拡張への対応がある。

謝辞 本研究にご指導いただいた前川守東京大学助教授(現電気通信大学教授)に感謝します。

## 参 考 文 献

- 1) Brzezinski, J., Helary, J., Raynal, M. and Singhal, M.: Deadlock Models and a General Algorithm for Deadlock Detection, *Journal of Parallel and Distributed Computing*, Vol.31, No.2, pp.112-125 (1995).
- 2) Chandy, K.N., Misra, J. and Haas, L.M.: Distributed Deadlock Detection, *ACM Trans. Comput. Syst.*, Vol.1, No.2, pp.144-156 (1983).
- 3) Gold, E.M.: Deadlock Prediction: Easy and Difficult Cases, *SIAM J. Comput.*, Vol.7, No.3, pp.320-336 (1978).
- 4) Holt, R.C.: Some Deadlock Properties of Computer Systems, *Comput. Surv.*, Vol.4, No.3, pp.179-196 (1972).
- 5) Knapp, E.: Deadlock Detection in Distributed Databases, *Comput. Surv.*, Vol.19, No.4, pp.303-328 (1987).
- 6) Kshemkalyani, A.D. and Singhal, M.: Efficient Detection and Resolution of Generalized Distributed Deadlocks, *IEEE Trans. Softw. Eng.*, Vol.20, No.1, pp.43-54 (1994).
- 7) Menasce, D.A. and Muntz, R.R.: Locking and Deadlock Detection in Distributed Data Bases, *IEEE Trans. Softw. Eng.*, Vol.SE-5, No.3, pp.195-202 (1979).
- 8) Minoura, T.: Deadlock Avoidance Revisited, *J. ACM*, Vol.29, No.4, pp.1023-1048 (1982).
- 9) Singhal, M.: Deadlock Detection in Distributed Systems, *Computer*, Vol.22, No.11, pp.37-48 (1989).

(平成8年6月28日受付)

(平成9年5月8日採録)



荻原 評 (正会員)

1964年生。1987年東京大学理学部情報科学科卒業。1992年同大学大学院理学系研究科博士課程修了。博士(理学)。同年より電気通信大学助手。並列/分散システムに関する研究に従事。ACM会員。



清水謙多郎 (正会員)

1957年生。1980年東京大学理学部情報科学科卒業。1985年同大学大学院理学系研究科博士課程修了。理学博士。同年東京大学大型計算機センター助手。1986年東京大学理学部助手。1991年電気通信大学助教授。1996年より東京大学農学生命科学研究科助教授。オペレーティングシステム、並列/分散処理、生命情報科学の研究に従事。本学会論文誌編集委員。ACM, IEEE, 電子情報通信学会, 日本ソフトウェア科学会, 日本シミュレーション学会, 日本生物物理学会各会員。