ソフトウェア基礎論　25−12
(1988. 5. 27)

一方向分散型計算システムに対する停止判定アルゴリズム

A Termination Detection Algorithm
for Unidirectional Distributed Computing System

溝口　一郎　　今井　正治
Ichiro MIZOGUCHI　　Masaharu IMAI
(豊橋技術科学大学　情報工学系)
(Toyohashi Univercity of Technology)

あらまし　停止判定は分散型計算システムでの重要な問題の一つである．本稿では一方向分散型計算システムに対する停止判定アルゴリズムを提案し，アルゴリズムの評価を行う．まずはじめに一方向分散型計算システムを有向グラフで表わし，この停止判定のアルゴリズムを示す．このアルゴリズムは，ある制限を加えた一方向分散型計算システムに適応することができる．また，アルゴリズムの計算時間の複雑さについて考察し，正当性の証明を行う．

Abstract　Termination detection is one of the most important problems in Distributed Computing Systems (DCS's). In this paper, an algorithm to detect termination of unidirectional DCS's (UDCS's) is proposed. A UDCS can be represented as a directed graph G=(V,E). In the proposed algorithm, two kinds of tokens 'cat' and 'mouse' are used. Here, 'mouse' is a data item to be processed, and 'cat' is a token which is used to detect the termination of the calculation in the UDCS. The algorithm is easy to implement, efficient and applicable to a wide range of UDCS's, such as VLSI computing systems and multicomputer systems.

1．はじめに

分散型計算システム(Distributed Computing System:DCS)は，個々の処理装置（Processing Element:PE）をメッセージリンクで結んだものである．コンピュータ・ネットワーク，ＬＡＮ，マルチ・コンピュータシステム，ＶＬＳＩシステム等は，ＤＣＳとして扱うことができる．

ＤＣＳは双方向のメッセージリンクを持つ双方向ＤＣＳと一方向のメッセージリンクしか持たない一方向ＤＣＳの２種類に分類される．今回議論の対象とするのは一方向ＤＣＳである．この種のＤＣＳの例としては，シストリック型計算システム[Kung81]，ある種のマルチ・コンピュータシステムなどを挙げることができる[Imai84]．

単独のＰＥに問題を与えた場合の停止判定は，そのＰＥが動作を停止したかどうかの判定によって行うことができる．これに対し，ＤＣＳに問題を与えた場合の停止判定は，ＤＣＳに含まれる全てのＰＥが動作を停止したかどうかを判定することによって行わなければならない．しかしながらＤＣＳ中の全てのＰＥに対して個々に停止判定を行うのは非現実的であり，非効率的である．そこである特定のＰＥの状態のみを用いて停止判定を行うことを考える．

双方向ＤＣＳに対する停止判定アルゴリズムは，既にDijkstraらによって提案されている[Dijk80]．しかし一方向ＤＣＳに対する停止判定アルゴリズムは，これまであまり議論されていない．本稿ではある制約を持った一方向ＤＣＳに対する停止判定アルゴリズムを提案する．また，アルゴリズムの計算時間の複雑さについて考察し，アルゴリズムの正当性の証明を行う．

2．定義及び記法

2．1　一方向分散型計算システム

一方向ＤＣＳは以下のように有向グラフＧを用いて表わすことができる．

$$G = (V, E) \qquad (1)$$

ここでＶは節点の集合であり，Ｅは枝の集合である．Ｖ中の節点はそれぞれＰＥに対応し，Ｅ中の枝はそれぞれメッセージリンクに対応する．一方向ＤＣＳ中で用いられるメッセージは有向グラフＧ中の枝の方向に沿って送られる．

節点υ_i，υ_jがＶに含まれ，かつ枝$e_k = (\upsilon_i, \upsilon_j)$がＥに含まれるならば，$\upsilon_i$は$\upsilon_j$の前者(predecessor)，$\upsilon_j$は$\upsilon_i$の後者(successor)と呼ぶ．

パスは連続した枝の系列である．パス中の枝の後者は最後の枝を除いて次の枝の前者になっている．すなわち

$$(\upsilon_1, \upsilon_2, \cdots, \upsilon_p) \qquad (2)$$

は

$(\upsilon_1, \upsilon_2), (\upsilon_2, \upsilon_3), \cdots, (\upsilon_{p+1}, \upsilon_p)$

という枝順のパスを表す．

閉路(circuit)はe_1，e_2，…，e_qというパスでe_1の前者がe_qの後者になっているものである．ｎ個（ｎ≧３）の節点を持つ閉路において，節点υ_i（$1 \leq i \leq n$）が互いに異なるとき，この閉路を単純な閉路(simple circuit)と呼ぶ．もし２つの閉路が，たかだか一つの節点を除いて同じ節点を含まないならば，これらの閉路は互いに素(independent)であるという．

ＤＣＳの停止判定のために，グラフＧ中より節点を１つ選び，この節点を環境節点(environment vertex)と呼ぶ．また，この節点以外の節点を内部節点(internal vertex)と呼ぶ．図１に一方向ＤＣＳの例を示す．

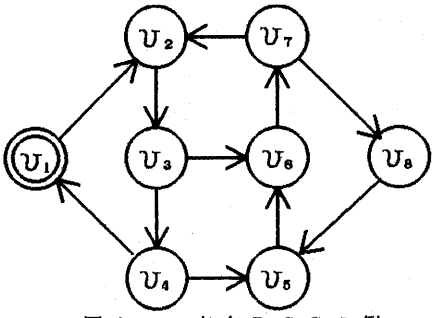

図１　一方向ＤＣＳの例
（節点υ_1が環境節点である）

2．2　ランク

グラフＧに含まれる閉路，枝，および節点の集合を，以下で定義されるランク(rank)と呼ばれる概念を用いることによって階層化する．

定義１（ランク１の閉路，節点，枝）

$C^{(1)}$を，環境節点υ_eを含む単純な閉路の集合とする．次に，$E^{(1)}$を$C^{(1)}$中の閉路に含まれる枝の集合とする．さらに$V^{(1)}$を，$C^{(1)}$中の閉路に含まれる節点の集合とする．このとき，

（１）$C^{(1)}$に含まれる閉路のランクは１である．

（２）$E^{(1)}$に含まれる枝のランクは１である．

（３）$V^{(1)}$に含まれる節点のランクは１である．□

もし枝の集合Ｅが$E^{(1)}$中に含まれない枝を含むとき，グラフは以下で定義される２またはそれ以上のランクの枝を含んでいる．

定義２（ランクｒの閉路，節点，枝）

Ｅ’を次のような枝の集合とする．

$E' = \{e \mid e=(\upsilon_i, \upsilon_j)$は

E－(E$^{(1)}$∪…∪E$^{(r-1)}$)に含まれ，
かつv_iはV$^{(r-1)}$に含まれる．}

またC$^{(r)}$を，E'に含まれる枝から始まる閉路の集合とする．次にE$^{(r)}$を，C$^{(r)}$中の閉路に含まれる枝の集合とする．さらに，V$^{(r)}$を，C$^{(r)}$中の閉路に含まれる節点の集合とする．このとき，

(1)C$^{(r)}$に含まれる閉路のランクはrである．

(2)E$^{(r)}$に含まれ，かつE$^{(1)}$∪…∪E$^{(r-1)}$に含まれない枝のランクはrである．

(3)V$^{(r)}$に含まれ，かつV$^{(1)}$∪…∪V$^{(r-1)}$に含まれない節点のランクはrである．□

最後にグラフのランクを以下のように定義する．

定義3（グラフのランク）

環境節点v_eから見たグラフのランクは，グラフ内の最も高いランクの枝のランクとする．□

例1

図1において，環境節点v_eは節点v_1である．定義1より閉路(v_1,v_2,v_3,v_4,v_1)がC$^{(1)}$に含まれ，ランク1となる．また枝(v_1,v_2),(v_2,v_3),(v_3,v_4),(v_4,v_1)がE$^{(1)}$に含まれるのでランク1となり，節点v_1，v_2，v_3，v_4がV$^{(1)}$に含まれるのでランク1になる．

次に定義2より閉路(v_4,v_5,v_6,v_7,v_2,v_3,v_4)と(v_3,v_6,v_7,v_2,v_3)がC$^{(2)}$に含まれ，ランク2となる．この2つの閉路に含まれる7本の枝(v_2,v_3),(v_3,v_4),(v_4,v_5),(v_5,v_6),(v_6,v_7),(v_7,v_2),(v_3,v_6)がE$^{(2)}$に含まれる．しかしこの7本の枝のうち，(v_2,v_3)，(v_3,v_4)の2本の枝は既にランクが1と定義されているので，残りの5本の枝(v_4,v_5),(v_5,v_6),(v_6,v_7),(v_7,v_2),(v_3,v_6)がランク2となる．また節点の場合にも，v_2,v_3,v_4,v_5,v_6,v_7の6個の節点がV$^{(2)}$に含まれるが，ランク2となるのはv_5，v_6,v_7の3つの節点だけである．

同様の方法により，ランク3の閉路，枝，節点を求める．閉路(v_7,v_8,v_5,v_6,v_7),枝(v_7,v_8),(v_8,v_5),節点v_8がランク3である．図1のグラフではランク4以上の閉路，枝，節点は存在しない．このためこのグラフのランクは，定義3より3となる．

以上まとめると以下のようになる．

(1)閉路のランク

(v_1,v_2,v_3,v_4,v_1)	ランク1
(v_4,v_5,v_6,v_7,v_2,v_3,v_4)	ランク2
(v_3,v_6,v_7,v_2,v_3)	ランク2
(v_7,v_8,v_5,v_6,v_7)	ランク3

(2)枝のランク

(v_1,v_2), (v_4,v_1)	ランク1:E$^{(1)}$
(v_2,v_3), (v_3,v_4)	ランク1:E$^{(1)}$,E$^{(2)}$
(v_4,v_5), (v_3,v_6) (v_7,v_2)	ランク2:E$^{(2)}$
(v_5,v_6), (v_6,v_7)	ランク2:E$^{(2)}$,E$^{(3)}$
(v_7,v_8), (v_8,v_5)	ランク3:E$^{(3)}$

(3)節点のランク

v_1	ランク1:V$^{(1)}$
v_2,v_3,v_4	ランク1:V$^{(1)}$,V$^{(2)}$
v_5,v_6,v_7	ランク2:V$^{(2)}$,V$^{(3)}$
v_8	ランク3:V$^{(3)}$

(4)グラフのランク

図1のグラフのランクは3である．

グラフG中に含まれる任意の2つの節点v_i，v_jを選んだとき，v_iからv_jへのパスとv_jからv_iへのパスとが存在するならば，このグラフGは強連結である．図1に示したグラフは強連結である．

定義4　衝突

グラフG中に含まれる任意の互いに異なる3つの節点v_i，v_j，v_kを

C_1＝(…，v_i，…，v_j，…，v_k，…)

C_2 = (…, υ_i, …, υ_k, …, υ_j, …) の形で含む閉路が存在しないならば，このグラフGを無衝突であると呼ぶ．衝突のあるグラフの例を図２に示す．

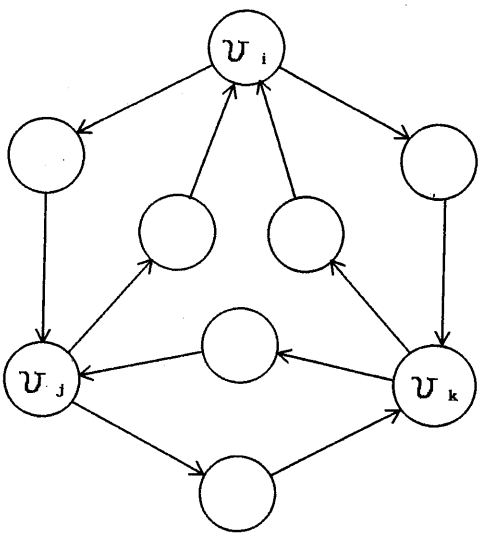

図２　衝突のあるグラフ

３．停止判定アルゴリズム

３．１　メッセージ

停止判定アルゴリズムでは以下の２種類のメッセージを用いている．

ネズミ：ＰＥ中で処理されるデータ項目．

ネ　コ：ＤＣＳの停止判定のために用いるトークンで，通過する枝の種類に応じたランクを持つ．

ネズミもネコも，ともに環境節点から送り出される．ネズミとネコは以下の性質を持つ．

（１）ネズミの性質

（ａ）環境節点が送り出すまでは，内部節点に存在しない．

（ｂ）節点の中でのみ，個数が変化する．

（ｃ）ネズミが存在しない節点で，自動的に発生することはない．

（ｄ）枝を通過するとき，他のネズミやネコを追い越さない．

（ｅ）処理が終わると，節点内で消滅する．

（２）ネコの性質

（ａ）環境節点が送り出すまでは，内部節点に存在しない．

（ｂ）節点の中でのみ，個数が変化する．

（ｃ）ネコが存在しない節点で，自動的に発生することはない．

（ｄ）枝を通過するとき，他のネズミやネコを追い越さない．

（ｅ）節点内で消滅することはない．□

３．２　アルゴリズム

どの節点にもネズミが存在しなくなったとき，全てのＰＥは動作を停止しているので，明らかにＤＣＳは終了している．停止判定アルゴリズムは，ネズミが存在しなくなったことを環境節点の状態からのみ判断する．

アルゴリズムは，環境節点に対する手続き，および内部節点に対する手続きの２つからなる．アルゴリズムの基礎となる定理を以下に示す．

定理１

節点υ_eを環境節点とするグラフＧのランクを１とする．また節点υ_eとυ_eにつながる全ての枝を取り去って得られるＧの部分グラフをＧ’とする．このときグラフＧ’は閉路を持たない．□

［証明］

ランクの定義から，もしグラフＧのランクが１ならば，υ_eはＧの全ての閉路に含まれる．Ｇ’はυ_eとυ_eにつながる枝を持たないので，閉路を持たない．（図３参照）□

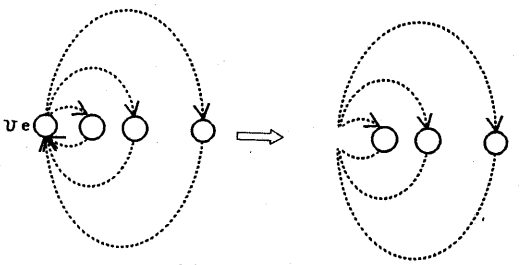

図３　定理１

定理２

もし節点U_iのランクがｒならば，節点U_iにつながる枝は$E^{(r)}$ ∪ $E^{(r+1)}$の中に含まれる．□

定理２の証明は付録１に示す．定理２より，ランクｒの節点とつながる枝を３つのタイプに分類することができる．

定義５（枝のタイプ）

（１）$E^{(r)}$にのみ含まれる枝をType 1とする．

（２）$E^{(r+1)}$にのみ含まれる枝をType 2とする．

（３）$E^{(r)}$と$E^{(r+1)}$に含まれる枝をType 3とする．□

定義５で，各枝のタイプは視点となる節点によって異なる場合があることに注意する．例えば図１のグラフでは，枝(U_7，U_2)は，節点U_2から見るとtype 2，節点U_7から見るとtype 1である．

３．３　環境節点に対する手続き

環境節点は，４つの状態BUSY，IDLE，CONT，FINのうちいずれかの状態をとる．初期状態はBUSY状態で，この状態は内部節点内にネコが存在しないことを示している．

（１）BUSY状態

ネズミが環境節点に存在しないとき，ランク１のネコを全ての外向きの枝へ送り出し，IDLE状態にする．

（２）IDLE状態

ランク１のネコが全ての中向きの枝から帰って来たら，FIN状態にする．ネズミが帰って来たら，CONT状態にする．

（３）CONT状態

ランク１のネコが全ての中向きの枝から帰って来たら，BUSY状態にする．

（４）FIN状態

ＤＣＳは停止した．□

環境節点の状態推移図を図４に示す．

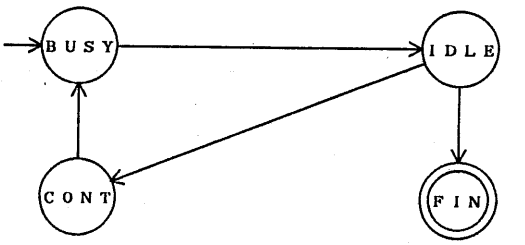

図４　環境節点の状態遷移図

３．４　内部節点に対する手続き

ランクｒの各内部節点は，５つの状態REST，AWAKE，SEND，CHECK，WAITのうちいずれかの状態をとる．初期状態はREST状態で，節点での計算が停止していることをを示している．

（１）REST状態

ネズミかネコが到着したら，AWAKE状態にする．

（２）AWAKE状態

Type 1の全ての中向きの枝からランクｒのネコが，Type 3の全ての中向きの枝からランクｒ＋1のネコが到着したら，SEND状態にする．

（３）SEND状態

（ａ）外向きの枝がType 1のみの場合は，その枝へランクｒのネコを送り出しREST状態にする．

（ｂ）それ以外の場合は，Type 2，Type 3の全ての外向きの枝へランクｒ＋1のネコを送り出し，CHECK状態にする．

（４）CHECK状態

ネズミが来たら，WAIT状態にする．Type 2，Type 3の全ての中向きの枝からネコが帰って来たら，Type 1，Type 3の全ての外向きの枝へランクｒのネコを送り出し，REST状態にする．

（５）WAIT状態

Type 2，Type 3の全ての中向きの枝からネコが帰って来たら，SEND状態にする．□

内部節点の状態推移図を図５に示す．また図６に図１の有向グラフで表される一方向ＤＣＳにアルゴリズムを適用した場合の各節点の状態遷移を示す．

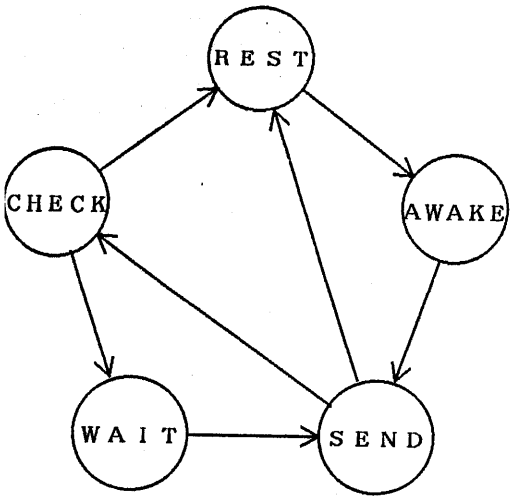

図５　内部節点の状態遷移図

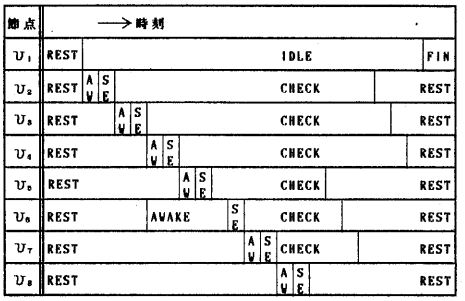

図６　各節点の状態遷移

３．５　アルゴリズムの正当性の証明

これまで示したアルゴリズムで一方向ＤＣＳの停止判定が行えることを証明する．

定理３

Ｇをランク Ｐの強連結の無衝突グラフとする．グラフＧ中のどの節点にもネズミが存在しないとき，かつそのときに限り環境節点v_e はＦＩＮ状態になる．□

証明は付録２へ示す．

３．６　アルゴリズムの複雑さ

提案したアルゴリズムの計算時間の複雑さは以下の定理で与えられる．

定理４

環境節点が全ての外向きの枝へネコを送り出すために必要な時間，内部節点がネコを通過させるために必要な時間，および環境節点が全ての中向きの枝からネコを受け取り停止判定をするまでに必要な時間を，それぞれ１単位時間とする．また，グラフＧの節点の個数をｎとする．グラフＧ中にネズミが存在しないとき，停止判定アルゴリズムの計算時間は最小３単位，最大２ｎ－１単位である．□

証明は付録３へ示す．

4．おわりに

本論文では衝突のない一方向ＤＣＳに対する停止判定アルゴリズムを示し，その正当性を証明した．衝突があるＤＣＳについてのアルゴリズムの開発は今後の課題である．

謝辞

日頃ご指導賜る本学の本多波雄学長，名古屋大学の福村晃夫教授，稲垣康善教授，京都大学の茨木俊秀教授，広島大学の山下雅史助教授，討論いただいた研究室の諸兄に感謝する．なお，本研究の一部は，文部省科学研究費補助金（総合研究（Ａ）研究課題番号62302032）による．

参考文献

[Dijk80] E. W. Dijkstra:"Termination Detection for Diffusing Computations," Information Processing Letters, Vol.11,No.1 (Aug. 1980).

[Imai85] M. Imai:"Termination Detection Algorithms for Unidirectional Distributed Computing Systems," Tech.Rep. USCMI 85-02, Univ. of South Carolina (Jan.1985).

[Imai86] M. Imai:"A Double-Tree Structured Multicomputer System and its Application to Combinatorial Problems," The Transactions of the IECE of Japan ,Vol.E69,No.9 (Sep.1986).

[Kung81] H. T. Kung,et al.:VLSI Systems and Computations ,Springer-Verlag (1981).

[Mizo88] 溝口一郎，今井正治:"一方向分散型計算システムに対する停止判定アルゴリズム，" 電子情報通信学会全国大会シンポジウムの論文

[Toku83] 徳田雄洋:"分散型アルゴリズムの基礎，" 情報処理，Vol.24 ，No.4 (Apr.1983).

付録１

定理２の証明

（１）節点v_i にランクｑ（ｑ＜ｒ）の枝が入っていると仮定する．

このとき節点v_i はランクｑの枝を持つ閉路に含まれ，そのランクは少なくともｑ以下となる．このためv_i のランクｒという仮定と矛盾する．

（２）節点v_i に節点v_j よりランクｑ（ｑ≧ｒ）の枝が入っていると仮定する．

（ａ）ｒ＝１のとき

このとき節点v_i は環境節点を含む閉路中に存在する．グラフが強連結であることから，閉路内のある節点v_k から他の節点v_j へのパスが存在し，新たな閉路ができあがる．新たな閉路が環境節点を含むならば，枝（v_j，v_i）のランクは１となる．また環境節点を含まないならば，枝（v_j，v_i）のランクは２となる．（図７，８参照）

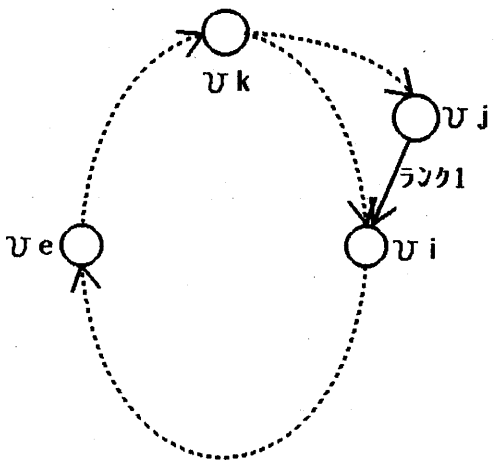

図７　閉路が環境節点を含む場合

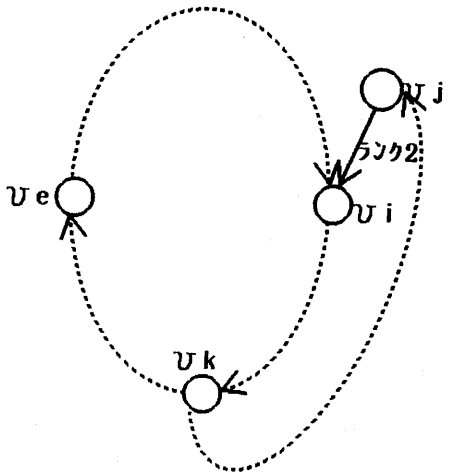

図８　閉路が環境節点を含まない場合

（ｂ）ｒ＞１のとき

このとき節点υ_kはランクｒ－１の節点を少なくとも１つ含む閉路の中の節点の１つであるである．グラフが強連結であることから，閉路内の節点υ_kから他の節点υ_jへのパスが存在し，新たな閉路が出来上がる．新たな閉路がランクｒ－１の節点υ_aを含めば，枝（υ_j，υ_i）のランクはｒとなる．またランクｒ－１の節点υ_aを含まなければ，枝（υ_j，υ_i）のランクはｒ＋１となる．（図９，１０参照）

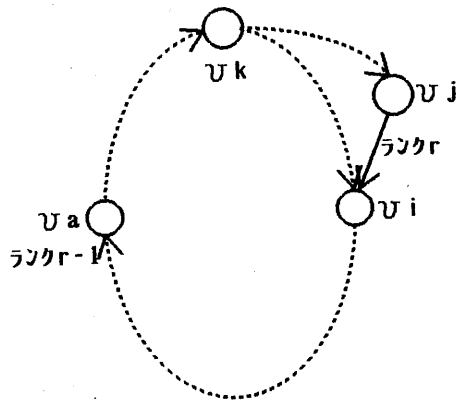

図９　閉路がランクr-1の節点υ_aを含む場合

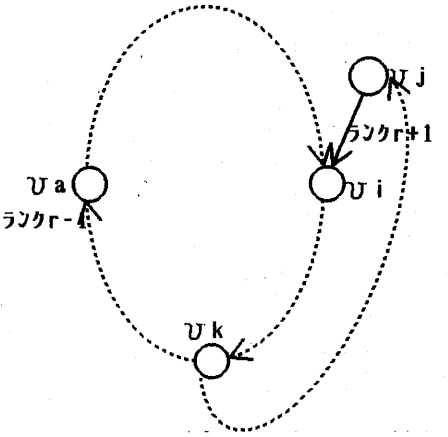

図１０　閉路がランクr-1の節点υ_aを含まない場合

以上（1），（2）より題意成立□

付録２

定理３の証明

定義２．２から明らかなように，任意のランクＰのグラフから，ランクＰ－１の点を除去すると，ｎ個（ｎ≧１）の節点とそれらを結ぶ枝が残り，閉路は存在しない．そしてこれらｎ個の節点はｍ個（１≦ｍ≦ｎ）の木を構成している．以下では，まずランク１のＤＣＳに対し定理３の主張が成立することを帰納法を用いて証明する．

（１）ランク１のＤＣＳ

定理１より，ランク１のＤＣＳから環境節点とそれにつながる枝を除去すると，閉路を持たないことが知られる．そこで，ランク１のＤＣＳに属する節点に対し最長距離の概念を定義し，これを用いて証明を行う．各節点の最長距離とは，環境節点からその節点へ至る最長パスの長さである．例えば図１１のグラフにおいて節点υ_1の最長距離は５，節点υ_2の最長距離は６である．

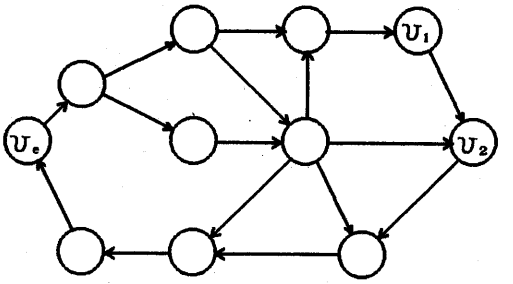

図１１　節点の最長距離

（ａ）十分性（グラフ中にネズミが存在しないとき）

ある有限な時間の後，環境節点から送られたネコにより，環境節点からの最長距離が１である節点はREST状態になり後者の節点に対しネコを送り出す．環境節点からの最長距離が１である全ての節点がREST状態になってから，ある有限時間の後，環境節点からの最長距離が２である節点がREST状態になる．同様にして全ての内部節点がREST状態になる．よって環境節点に入る全ての枝からネコが戻って来るので，環境節点ははFin 状態になる．

（ｂ）必要性（環境節点がFin 状態のとき）

環境節点からの最長距離がｎである節点でのネズミの存在について考える．

（Ⅰ）ｎ＝１のとき

REST状態になり後者の節点にネコを送りだした後，環境節点からネコが送られて来ることはないので，この節点の内部状態はREST状態のままであり，ネコは存在しない．

（Ⅱ）ｎ≦ｒ－１の節点にネズミが存在しないと仮定

このときｎ＝ｒの節点では，REST状態になり後者の節点にネコを送りだした後，ｎ＝ｒの節点からネコが送られて来ることはないので，この節点の内部状態はREST状態のままであり，ネコは存在しない．

以上（Ⅰ）（Ⅱ）よりランク１のＤＣＳに対して定理３が成立する．

（２）ランク２以上のＤＣＳ

（Ⅰ）Ｐ＝１のとき

（１）よりＰ＝１のとき明らかに定理が成り立つ．

（Ⅱ）Ｐ≦ｋ－１のとき

Ｐ≦ｋ－１のとき，定理が成り立つと仮定する．

（Ⅲ）Ｐ＝ｋのとき

ランクｋ－１以下の点を除去したときｍ個の木が残る．ランクｋ－１の節点のうちランクｋの枝がつながっているものは除去しない場合，ｍ個のグラフが残る．このｎ個のグラフは，除去の対象とならなかったランクｒ－１の点を環境節点とみなすことにすると，全てランク１のグラフである．よってこれらｍ個のグラフでは，（１）より定理が成り立ち，Ｐ＝ｋの場合にも定理が成り立つ．

以上（1），(2)より題意成立．□

付録３

定理４の証明

（１）最小値

停止判定に必要な時間が最小になるのは，グラフＧが図１２のような形の場合である．この時，明らかに計算時間の最小値は３となる．

（２）最大値

仮定よりグラフは強連結であるので，環境節点からグラフ中のある節点U_aに至る単純なパスの長さは，たかだかｎ－１である．逆に節点U_aから環境節点U_eに至る単純なパスの長さもたかだかｎ－１である．このような節点U_aが存在するグラフの例を図１３に示す．このようなグラフにおいて，Ｇ

中にネズミが存在しないとき，ネコは v_a 以外の内部節点をそれぞれ２回，v_a を１回通過する．したがって，アルゴリズムの計算時間はグラフのランクに依存することなく最大２ｎ－１となる．

以上(1),(2)より題意成立□

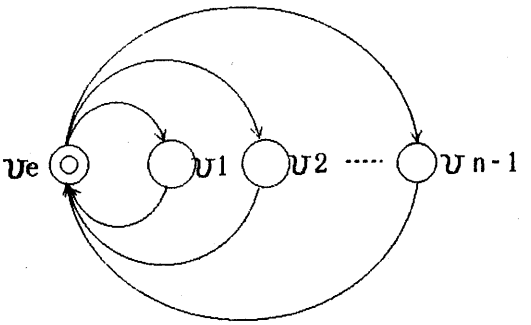

図１２　計算時間が最小となる例

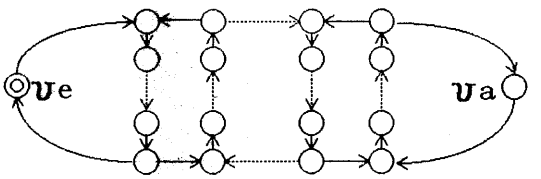

図１３　計算時間が最大となる例