アルゴリズム 59－6
（1997. 11. 28）

分散アルゴリズムの初期条件の作る構造について

坂本 直志
東京工業大学大学院情報理工学研究科
sakamoto@noc.titech.ac.jp

抄録 分散アルゴリズムにより初期条件 B を満たすあらゆる初期値から初期条件 A を満たすような初期値を作ることができるという関係を考える。この関係は、同値類に対して半順序関係になり、最大元、最小元を持つ無限の要素を持つ lattice を作る。

また、この関係を用い、従来から考えられている初期条件や新たに考えた初期条件の間の関係を調べた。

On the Structure Which Initial Conditions for Distributed Algorithms Make

Naoshi SAKAMOTO
Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology

Abstract Let's consider the relation between initial conditions A and B such that a distributed algorithm given any initial value satisfying B, can make some outputs satisfying A. Then we can find that this relation is a partial order for equivalence class, and makes an infinite lattice which has maximum and minimum elements.

And, we study the relation among ordinary and newly introduced initial conditions by this relation.

1 はじめに

分散アルゴリズムとはコンピュータ同士をネットワークで結び、互いに通信しあいながらネットワーク上の問題を解決して行くアルゴリズムである。この分野の中で、匿名ネットワークという、各コンピュータに唯一の番号が与えられいないモデルも研究されている。

本研究では匿名ネットワークに対する分散アルゴリズムに与える初期条件同士の関係の作る構造について幾つかの結果を示す。

この分野では、特定の条件や特定の問題に関する、決定可能性や通信の複雑さに関する議論が行なわれてきた。Angluin [Ang80] は匿名ネットワーク上で、リーダー選出問題を解くことができる前提条件やグラフの構造について議論し、

1. プロセッサが非決定性か、あるいは決定性か、
2. 全ての初期条件が同一か、あるいはある特定のプロセッサをリーダー (Angluin は centered という言葉を使っていた) として指定するか、
3. 停止するのは全てのノードか、あるいは一つのノードで良いか、

の組合せにより作られる8つの条件のもとで、区別可能なグラフのクラスの包含関係を明らかにした。Johnson と Schneider [JS85] は異なるモデル間のリーダー選出問題に関するモデルの等価性について調べている。

Yamashita と Kameda[YK88] は初期条件として、グラフのトポロジー、グラフの頂点数、グラフの頂点数の上限、無情報を考え、それぞれに対してリーダー選出問題、辺選択問題、生成木生成問題、トポロジー認識問題の決定可能性を調べている。

Yamashita と Kameda[YK89] は、初期条件として、隣接している頂点へのチャネルの情報のみ、ネットワークの頂点数の上限、ネットワークの頂点数の総数、グラフのトポロジーと自頂点の位置などを考え、それぞれの初期条件に対して、送信時にあらゆる頂点に届くような通信や、受信時にどの頂点から来たかわからないような通信といった通信のモデルの違いがある場合の、リーダー選出問題の決定可能性を調べている。

これらは「どんな初期条件が特定の問題を解くか?」についての結果であるが、Attiya, Snir, Warmuth [ASW88] は匿名リング上でサイズ (構造がリングなので、ネットワーク全体の構造を知ることと等価な情報) が分かっている時に、各プロセッサに与えた値を交換しあいながら、どのような関数がネットワーク上で計算できるかについて調べ、Yamashita と Kameda[YK96] ではさらに、この結果を一般のグラフに対して拡張している。

これらの論文における分散アルゴリズムの捉え方は若干相違があるが、分散アルゴリズムを動作させる条件の捉え方は、ネットワークを構成するグラフの構造と、各頂点から接続している辺を区別するために付けられるポートナンバーと、各頂点に与えられる初期値と、同期、非同期、決定性、非決定性、ランダム性など各頂点上のプロセッサの動作条件と、メッセージの送受の仕方と、各プロセッサに与えられるプログラムにまとめることができる。

本研究では任意のネットワークを構成するグラフ上で、任意の各頂点から接続している辺を区別するために付けられるポートナンバーと、ある条件を満たす、各頂点に与えられる初期値を非同期、決定性のプロセッサに与え、各隣接している頂点同士が、相互にポートナンバーの指定により通信が可能で、各プロセッサには同一のプログラムが与えられるというモデルで、ある条件を満たす初期値から別の条件を満たす初期値を分散アルゴリズムで生成できるかという関係を考え、これらの条件の持つ性質に関して調べた。

分散アルゴリズムはグラフの頂点に全て同じプログラムを与えるので、全ての頂点に同じ初期値を与えてプログラムを動作させた場合、各頂点上のプロセッサがそれぞれ違う出力を出せるとは必ずしも言えない。任意のグラフ上でどのような初期値が与えられるかがわからないような前提の分散アルゴリズムの場合、各プロセッサが必ず異なる値を出力することを期待するようなプログラムを作ることはできない。そこで、多くの場合、「頂点に一意に番号が付けられている。」や「全頂点数があらかじめ与えられている。」など、一定の条件を満たすような初期値が各頂点に与えられていることを前提にした分散アルゴリズムが提案されている。そのような初期値の与え方を限定するような条件を初期条件と言うことにする。特定の初期条件により、各頂点に与えられる初期値が限定されるため、その初期条件を前提とした分散アルゴリズムは、何も前提としていない分散アルゴリズムより複雑な出力を出すことができる。ある初期条件 B を満たす初期値を各頂点に与えると、別の初期条件 A を満たす初期値を必ず出力できるような分散アルゴリズムがあれば、

初期条件 A を前提とした分散アルゴリズムは、全て初期条件 B を前提とした分散アルゴリズムに書き換えることができるので、A より B の方がより使用できる分散アルゴリズムが多く、情報量が多いとみなせる。この関係を、$A \leq B$ と表すことにする。また、この初期条件はグラフと初期値の対の集合と考えることもできるが、この集合が recursive enumerable であるような初期条件を recursive coloring と呼ことにする。

本研究では、recursive coloring である初期条件間の関係が同値類の上で最大限、最小限を持つ無限の構造を持つ lattice を作ることを示した。また、先に示した過去に示されたいくつかの初期条件 (これはすべて recursive coloring に含まれる) と k 個の頂点に 1 を与える初期条件、全体として k 種類の値を与える初期条件と言う基本的な構造を持つ初期条件間の関係も明らかにした。

本論文では 2 節に基本的な定義を与え、、3 節では幾つかの初期条件の定義と基本的な結果を与え、4 節では具体的に与えたアルゴリズムにより示される結果を示し、5 節では view の概念を用いた否定的な結果を示し、6 節では初期条件同士の関係をまとめ、この構造に関する考察をした。

2 準備

λ は空列をあらわすものとする。$a \bmod b$ は a を b で割った余りをあらわすものとする。

ポートナンバーが付けられた有限単純連結無向グラフ $G = (V, E, \sigma)$ をこのネットワークをあらわすモデルとして導入する。$V = \{v_1, \ldots, v_n\}$ はプロセッサをあらわす頂点の集合、E は通信線をあらわす辺集合とする。頂点 v の次数を $\deg(v)$ で表す。ポートナンバーをあらわす σ は各頂点に対する関数の集合 $\{\sigma[v] \mid v \in V\}$ で $\sigma[v]$ は v に接続する辺に対して 1 から v の次数 $\deg(v)$ までの値を与える関数である。一般にこのポートナンバーは同じ辺でも両端の頂点からは違う番号がつくことが許されている。つまり一般には必ずしも $\sigma[v](u,v) = \sigma[u](u,v)$ は成り立たない。グラフ G の頂点集合を $V(G)$ と表し、path $p = v_1 \cdot v_2 \cdots v_k$ は v_i, v_{i+1} が隣接しているような頂点の列とする。

初期条件はグラフに対して各頂点に与える初期値の与え方を定めるものである。グラフの頂点に対する値の与え方を「色づけ」と呼ぶことにする。初期条件 A に対して、グラフ G を一つ固定すると G に対する初期条件を満たした色づけ $a_1, a_2, \ldots$ が幾つか具体的に定まる。つまり初期条件 A はグラフごとに色づけの集合を与える関数である。また、色づけ a に従って各頂点 v に値 $a(v)$ が割り当てられたラベル付のグラフを「着色グラフ」と呼び、G^a と表す。初期条件はグラフからグラフに対する色づけの集合を与える関数と考えることもできるが、着色グラフ G^a の集合と考えることもできる。

初期条件を着色グラフの集合として考えた時、その集合が recursive enumerable であれば、その初期条件は recursive coloring であると言う。以下簡単に r.c. と表す。

匿名ネットワーク上の分散アルゴリズムはネットワークを表現するグラフ G と、そのグラフに関する初期値の与え方である色づけ a とアルゴリズム M により計算が定義される。はじめに、各頂点上 v にプロセッサが配置され、プロセッサに入力として通信可能なポートナンバーの集合 $\{1, 2, \ldots, \deg(v)\}$ と初期値 $a(v)$ が与えられる。但し、v 自身の頂点の名前は与えられない。プロセッサは通信をするためメッセージの送信コマンドと受信コマンドを持っており、送信コマンドはメッセージと送りたいポートナンバーを指定すると送信され、受信コマンドは受信バッファをアクセスし、読まれていない一番前に着信したメッセージとそのメッセージの送られて来たポートナンバーを返す。

分散アルゴリズムの実行のモデルには同期モデルと非同期モデルがある。同期モデルでは各頂点上のプロセッサは共通の時計と同期し常に同時刻に状態が推移し、また全ての通信線で通信にかかるコストは一定である。非同期モデルでは各プロセッサは独立の時計により動き、一般にプロセッサ同士は同期していない。通信線の通信にかかるコストはそれぞれ異なり、時刻により変化する可能性もある。また、プロセッサからは通信線の通信コストを決定することはできない。但し、この論文では同じ通信線を通って着信するメッセージは送信順に受け取れる FIFO モデルを仮定している。なお、同期モデルは非同期モデルの特別な場合であり、同期モデルでの決定不能性は非同期モデルの決定不能性でもある。

なお、本論文では分散アルゴリズムの初期値を整数と定義しているが、複数の値や整数以外の値に対しても適当なエンコードにより整数に変換して与えることができるので、具体的な初期値として、特に断らずに

一つの整数以外の初期値を使う場合がある。

3 基本的な結果

ここでは、幾つかの初期条件と基本的な関係を示す。

定義 3.1 本稿で対象とする分散アルゴリズムの初期条件として次のものを定義する。

- *ALL*: グラフ G に対して、頂点 v には G の表現とその表現における v のラベルを与える
- *NUMBERING*: それぞれの頂点に 0 から頂点数より 1 小さい値まで、各頂点に唯一の番号を与える
- *k-LEADER*: k 個の頂点にだけ 1 を与え、残りの頂点には 0 を与える
- *TOPOLOGY*: グラフ G に対して、頂点に G の表現を与える
- *SIZE*: 全ての頂点にグラフの頂点数を与える
- *UPPERBOUND*: 各頂点にグラフの頂点数以上の有限な値を与える
- *k-COLOR*: 各頂点にそれぞれ $\{0,\ldots,k-1\}$ のどれかを与え、全体で k 種類の値を与える

これらは明らかに *r.c.* に属する。

二つの初期条件 A, B に対して、$A \leq B$ とは、ある分散アルゴリズム M が存在し、任意のグラフ G と、B の初期条件を満たす任意の色づけ $b \in B(G)$ に対して、ある初期条件 A を満たす色づけ $a \in A(G)$ が存在し、M を G^b 上で動かした時、各頂点 v が $a(v)$ を出力することを言う。$A \leq B$ かつ $B \leq A$ の時、$A \equiv B$ と、また、$A \leq B$ かつ $B \not\leq A$ の時、$A < B$ と表す。

簡単な例として *k-COLOR* の間の関係を示す。

定理 3.2 $k_1 \leq k_2$ ならば、*k_1-COLOR* $\leq$ *k_2-COLOR*

さて、このように定義された $\leq$ には次の性質がある。

定理 3.3 初期条件に対する $\leq$ は反射律、推移律を満たし、*r.c.* 上で同値類に関して最大限、最小限を持った *lattice* を作る。

4 構成的な結果

この節では、*k-LEADER* に関する生成木を作るアルゴリズムと、生成木同士が統合し合うアルゴリズムを示し、他の初期条件に対する構成的な結果を示す。

プログラムの記述として ALGOL 風のプログラミング言語に *Send*, *Receive* という二つの通信用の命令を加えた言語を用いる。*Send*(M,p) はポートナンバー p のポートにメッセージ M を送信するコマンド、*Receive*(M,p) は受信バッファから一つ送られて来たメッセージを取り出し、そのメッセージの内容が M に、送られて来たポートナンバーが p に入る。

はじめに *k-LEADER* に対する k 個の生成木を作るアルゴリズム、生成木内の情報を取得するアルゴリズム、作成した生成木全てが同形になるまでそれぞれの木を統合して行くアルゴリズムを示した後、このアルゴリズムから導かれる結果を述べる。

4.1 生成木の生成

k-LEADER の初期条件が与えられた時、1 を与えられた k 個の頂点を始動プロセスとして動作し、1 が与えられた頂点を根とする k 個の生成木を作るアルゴリズムを与える。

ここで言う生成木を作ることとは、各頂点が相互に親子関係になるように変数 *parent*, A の値を決めて、構成された親子関係が全体として 1 を初期値として与えられた頂点を根とする木にすることである。

なお、以下のアルゴリズムで作られる親子関係は、一つの頂点の中では変数 *parent* が指す親はただ一つで、子の集合である A には含まれない。また、初期値として 1 を与えられた頂点は *parent* を持たず、0 を与えられた全ての頂点は必ず 1 を与えられた頂点の子孫になるので、必ず k 個の木になる。

分散アルゴリズム 1 [*k-LEADER*から k 個の生成木を作るアルゴリズム]

$R = A = \emptyset$;
if (0が与えられた)
  then *Receive*("子になれ", x);
    parent := x;
    $PORT := PORT - \{parent\}$;
fi
foreach $p \in PORT$ **do** *Send*("子になれ", p); **od**
while $(R \cup A \neq PORT)$ **do**
    Receive(M,p);

if (M = "子になれ") **then** *Send*("拒否", p);
elsif (M = "承認") **then** $A := A \cup \{p\}$;
elsif (M = "拒否") **then** $R := R \cup \{p\}$; **fi**
od
if (0が与えられた) **then** *Send*("承認", *parent*); **fi**
停止する

以下の節ではこのアルゴリズムにより生成木が作られたことを前提にする。

4.2 生成木内の情報の取得

作った生成木に対して、「親から来たトークンを子に順番に渡し、全ての子に渡したら親に返す」というアルゴリズムを用い、根の頂点がトークンを発生させることにより、生成木内の全ての頂点にトークンを巡回させることが可能である。親から来るトークンは必ず一回であることを利用することで生成木内の頂点に順番にユニークな名前をつけることができる。さらに同様にして木のトポロジーと木の中での自分の位置を求めることもできる。

4.3 生成木の同士の通信

有向パス $p = v_0 v_1 \cdots v_m$ に対して、ポートナンバーの列 $r = \sigma[v_0](v_0, v_1)\ \sigma[v_1](v_1, v_2) \cdots \sigma[v_{m-1}](v_{m-1}, v_m)$ を v_0 に対する (p の) 相対パスと呼ぶことにする。相対パスが与えられれば、それに沿ってメッセージを送ることが可能になる。また、$head(r) = \sigma[v_0](v_0, v_1)$, $body(r) = \sigma[v_1](v_1, v_2) \cdots \sigma[v_{m-1}](v_{m-1}, v_m)$ とする。

前節までの議論を用い、ネットワーク上に生成木を作った場合、全ての頂点はある生成木に必ず含まれるので、生成木の外向辺 (その生成木に接続している生成木に含まれない辺) までの相対パスがあれば、根から隣接した生成木までメッセージを送ることが可能である。また、外向辺から入って来たメッセージに対して、そのポートナンバーを添えて親に送り、子から相対パスが添えられて来たメッセージが送られて来たらポートナンバーを相対パスに付加して親に送るようにすると、根には外向辺までの相対パスが添えられたメッセージが届く。ここで、外向辺までの相対パスを相対ポートナンバーと呼ぶことにすると、各生成木の根同士が相対ポートナンバーにより指定された通信チャネルでつながっているようなネットワークをシミュレートすることができる。(但しこのネットワークは単純グラフではない)。このネットワークに対しては全頂点数は k となる。これを相対ネットワークと呼ぶことにする。

4.4 生成木の統合

さて、このようにして作ったそれぞれの生成木に対して、全ての生成木が同形になるまで木をなるべく統合するアルゴリズムを示す。このアルゴリズムは対称性がないネットワークに対しては最終的に一本の生成木にまで統合される。

生成木内からローカルに振った頂点の番号や、ポートナンバーや、統合した生成木の数 (最初の入力で1を受け取った頂点の数) などを集め、生成木の表現を作る。この表現に対して何らかの全順序を与えることで、木に対して全順序を与えることは可能である。

木の統合の基本的なアルゴリズムのアイディアは次の通りである (これは相対ネットワーク上で実行される)。

分散アルゴリズム 2 [生成木を統合するアルゴリズムのアイディア]

while 全ての木が同形でない **do**
  if 自分の木が隣接している木の中で最大
    then 自分より小さい木に"統合せよ"というメッセージを送信; **fi**
  if "統合せよ"を受け取った
    then 統合する; **exit**; **fi**
od

「全ての木が同形でない」は次のようにして調べる。

まず、同期ネットワーク用の分散アルゴリズムはメッセージ通信量を気にしなければ、非同期ネットワーク上で実行することができる。これは、同期ネットワーク用の分散アルゴリズムが与えられた時、アルゴリズムを同期が必要な部分に分け、その部分ごとに、隣接している頂点全てに対して同期をするためのメッセージを送り、隣接している全ての頂点から同期をするためのメッセージが送られて来てから次のステップに移るようにアルゴリズムを変更することにより、非同期ネットワークで実行することが可能になるからである。

さらに、これは相対ネットワーク上でも同様の議論により、各木の根同士が同期しあいながらアルゴリズムを実行することができる。

そして、「全ての生成木が同形かどうか」は、同期ネットワーク上であれば、距離 k までの全ての情報を

収集することが k ステップで可能なので、上の議論により、相対ネットワーク上のアルゴリズムとしても実現可能である。

よって、上記のアルゴリズムのアイディアは「統合する」部分以外は相対ネットワーク上の同期アルゴリズムとして構成可能である。

しかし、木の統合は相対ネットワーク上の同期アルゴリズムとしては処理できない。統合するためには具体的には次のような操作を各頂点のレベルで実行する。

- "統合せよ" を相対パス r から受けた根

  $parent := head(r)$; $A := A - \{head(r)\}$;
  $Send(\langle$"統合する", $body(r)\rangle, head(r))$;

- $\langle$"統合する", $r\rangle$ を親から受け取った頂点

  $A := A \cup \{parent\}$;
  $parent := head(r)$;
  $Send(\langle$"統合する", $body(r)\rangle, head(r))$;

- $\langle$"統合する", $body(r)\rangle$ をポートナンバー p の外向辺から受け取った頂点

  $A := A \cup \{p\}$; $R := R - \{p\}$;
  $Send(\langle$"統合する", $p\rangle, parent)$;

以上のような操作を行うことにより "統合せよ" を送った生成木まで "統合する" というメッセージが届き、木自体が統合される。(図 1)

ここで、はじめの相対ネットワーク上で考えた同期アルゴリズムのアイディアを生かすためには、「統合する」という前後で各相対ネットワーク同士が相対ネットワーク上で同期を取る必要がある。つまり、アルゴリズムのアイディアは相対ネットワークに対する同期アルゴリズムで考えていたが、統合のプロセスは個々のプロセッサに対するアルゴリズムであり、しかも統合により相対グラフに関してはグラフの構造がダイナミックに変わって行くため、このアルゴリズムを完成させるには統合のプロセスの前後で同期がとれるかどうかを吟味する必要があるということである。具体的にはアイディアの部分の「全ての木が同形でない」という条件を調べるためには、全ての木が統合の操作を行っていないことが必要で、そのため全ての木に対して、統合するタイミングと、全ての木に対する調査のタイミングの同期を取る必要がある。このために、統合が終了した木や、統合をしなかった木は隣接した木に「統合フェーズ終了」のメッセージを送れれば良いが、そのためには、自分の木が、統合されるか否かが各時刻で分かれば良い。先に示したアイディアでは統合する木とされる木だけが統合をすることを知り、他の木はメッセージ待ちの状況になってしまうが、隣接している木に統合をしないという情報を積極的に流すようにすれば、隣接した木から送られて来るメッセージから統合されないかどうかが決定できる。

完成した分散アルゴリズムを分散アルゴリズム 3 として示す。

分散アルゴリズム 3 [木の統合]

while 全ての木が同形でない **do**
  if 自分の木が隣接している木の中で最大
    then 自分より小さい木に"統合せよ"というメッセージを送信;
      全ての"統合せよ"を送った木が、統合を完了したか、統合を断るまで待つ;
      隣接する全ての木に「統合フェーズ終了」のメッセージを送る;
    else 自分より小さい木に"統合しない"というメッセージを送信;
      全ての相対ポートナンバーからメッセージを受け取るまで待つ;
      if "統合せよ"を受け取った
        then "統合せよ"を受け取った一つの木Tを選ぶ;
          Tを除いて"統合せよ"を受け取った全ての木に"統合しない"というメッセージを送信;
          Tと統合する; **exit**;
        else 隣接する全ての木に「統合フェーズ終了」のメッセージを送る;
      fi
  fi
od

以上のアルゴリズムを実行することにより、ネットワーク上は全て同形な生成木により分割される。また、トークンを回すことにより、結果的に幾つの木を統合したかも調べることができる。

4.5 示したアルゴリズムから導かれる定理

定理 4.1 1-$\mathcal{LEADER} \equiv \mathcal{NUMBERING} \equiv \mathcal{ALL}$

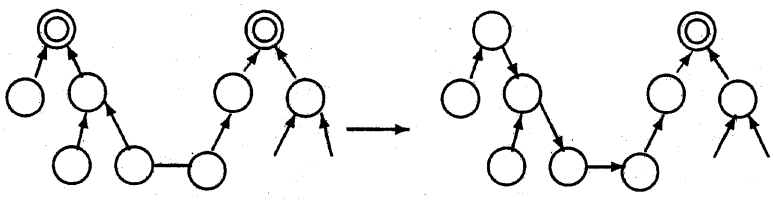

図 1: 木の統合

定理 4.2 $SIZE \leq k\text{-}LEADER$

定理 4.3 $k_1 \leq k_2$ で、k_1 が k_2 で割り切れれば、$k_2\text{-}LEADER \leq k_1\text{-}LEADER$。

定理 4.4 グラフのサイズが k_1k_2 以上ならば $k_1\text{-}COLOR \leq k_2\text{-}LEADER$。

5 View を用いた否定的な結果について

本節では view と呼ばれる概念を用いて、決定不能性に関する結果を示す。

view は Yamashita と Kameda[YK88] により導入された概念である。着色グラフ G^a に対して、グラフの頂点 v の view $T(v)$ とはラベル付の次数が有限な根付無限木で、根 x は v に対応し、ラベル $a(v)$ が付けられている。v と G^a 上で隣接している頂点 $v_1, \ldots v_{\deg(v)}$ に対して $T(v)$ ではそれに対応する $x_1, \ldots, x_{\deg(v)}$ が存在し、x と $x_1, \ldots x_{\deg(v)}$ の間にはそれぞれ $\sigma[v](v, v_i), \sigma[v_i](v, v_i)$ によりラベルが付けられている辺が存在する。さらに $x_1, \ldots, x_{\deg(v)}$ はそれぞれ $T(v_1), \ldots, T(v_{\deg(v)})$ の根になっている。また、深さ d までの view は $T^d(v)$ で表す。

Yamashita と Kameda[YK88] の表記では特定のグラフに対してポートナンバーと頂点を引数としていたが、本論文では複数のグラフの view に対して議論をするので、view を一意に決めるために、グラフの構造とポートナンバーとグラフの色付のすべてを与える G^a を引数とした $T_{G^a}(v)$ という表記をする。

view の性質としてグラフ G^a のある頂点 v と別のグラフ $G'^{a'}$ の頂点 v' で view が同形な場合を考えると、各頂点間に view が同形な対応が存在する。

補題 5.1 着色グラフ $G^a, G'^{a'}$ 上に同型な *view* を持つ頂点 v, v' が存在する時、G^a 上の任意の頂点 u に対し、同型な *view* を持つ $G'^{a'}$ 上の頂点 u' が存在する。

また、同期ネットワーク上では view が等しい頂点は互いに同じ出力を出す。

補題 5.2 着色グラフ $G^a, G'^{a'}$ 上の頂点 v, v' の *view* が同型である時、同期ネットワークにおいて全ての時刻において v, v' 上のプロセッサは同じ状態に入る。

この補題により、初期条件 A を満たす異なる出力を出さなければならないプロセッサが初期条件 B により同じ view をもつことを示すことができれば、B の初期条件を満たす色づけから A の初期条件を満たす色づけを出力するような分散アルゴリズムが存在しないことを示すことができる。

ここで、$k\text{-}COLOR$ の view の性質を調べる。

自然数 m に対して、頂点集合 $\{0, \ldots, km-1\}$ 個のリング $(E = \{(i, i+1 \bmod km) \mid i = 0, \ldots km-1\})$ を考え、ポートナンバーは $\sigma[i](i, i+1 \bmod km) = 2, \sigma[i](i+km-1 \bmod km, 1) = 1$ とし、色づけの関数 a を $a(v) = v \bmod k$ と置く。

頂点 i と $i+kj \bmod km$ $(j = 1, 2, \ldots)$ に対して、以下のことが成り立つので view は同形である。

- 頂点同士の色づけは等しい。
- 深さ t までの view が等しい時、同じ長さ t までの相対 path に対して、隣接する頂点の着色とポートナンバーはそれぞれ等しいので、深さ $t+1$ の view も等しい。

このグラフで、同じ色の頂点は同じ view を持ち、また、m に制限がないので、このようなグラフは無限に定義できる。

これにより次の性質が示せる。

定理 5.3 $k_1 > k_2$ ならば $k_1\text{-}COLOR \not\leq k_2\text{-}COLOR$.

定理 5.4 $k \geq 2$ ならば $k\text{-}COLOR \not\leq TOPOLOGY$。

定理 5.5 k_1-*LEADER* $\not\leq$ k_2-*COLOR*

定理 5.6 *UPPERBOUND* $\not\leq$ *k-COLOR*

また、同様の発想で、*k-LEADER* において各対応する頂点同士が、同形な view を持つようなグラフを無限個作ることができるので、次のことも示せる。

定理 5.7 k_1 が k_2 で割り切れなければ k_1-*LEADER* $\not\leq$ k_2-*LEADER*

定理 5.8 $k \geq 2$ の時 *TOPOLOGY* $\not\leq$ *k-LEADER*

6 まとめ

以上の議論により、紹介した*ALL*, *NUMBERING*, *k-LEADER*, *TOPOLOGY*, *SIZE*, *UPPERBOUND*, *k-COLOR*の間の関係は図 2 により示される。但し矢印の結ばれてない自明でない関係は全て相互に決定不能である。

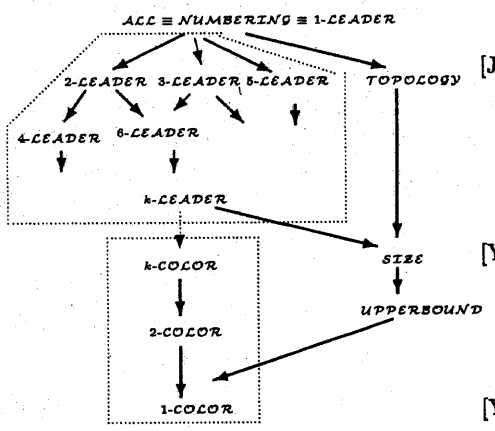

図 2: r.c. の初期条件の作る構造

なお、k_1-*COLOR* $\leq$ k_2-*LEADER* は有限個の場合には成り立たないので、図上では点線で表した。また、前節で示した他の否定的な結果は特別なグラフに関する証明であるが、容易にそのような構成不可能なグラフが無限個あることを示すことができる。それにより、$\leq$ の条件を緩めて「有限の場合を除き全てのグラフに対して決定可能」とすれば、全ての関係はそのまま成り立つ。

さて、本研究に関する未解決問題としては、「*k-LEADER* と *k-COLOR* を分離するような問題は存在するか?」「*SIZE* と *UPPERBOUND* を分離するような問題は存在するか?」「無限の階層は他にどのような含まれ方をするのか?」などが現在考えられる。

今後は recursive coloring に対する確率的な分散アルゴリズムによる分析や、*SIZE* など一般に良く用いられる初期条件に関する別の特徴付けなどが可能かどうか等に対しても研究を行っていきたいと思う。

参考文献

[Ang80] D. Angluin. Local and global properties in networks of processors. *Proc. 12th ACM Symp. on Theory of Computing*, pp. 82–93, 1980.

[ASW88] Hagit Attiya, Marc Snir, and Manfred K. Warmuth. Computing on an anonymous ring. *Journal of the Association for Computing Machinery*, Vol. 35, No. 4, pp. 845–875, 1988.

[JS85] Ralph E. Johnson and Fred B. Schneider. Symmetry and similarity in distributed systems. *Proc. 4th Ann ACM Symp. on Principles of Distributed Computing*, pp. 13–22, 1985.

[YK88] M. Yamashita and T. Kameda. Computing on anonymous networks. *Proc. 7th Ann ACM Symp. on Principles of Distributed Computing*, pp. 117–130, 1988.

[YK89] M. Yamashita and T. Kameda. Electing a leader when processor identity numbers are not distinct. *Lecture Notes in Computer Science*, Vol. 392, pp. 303–314, 1989. Distributed Algorithms.

[YK96] M. Yamashita and T. Kameda. Computing function on asynchronous anonymous networks. *Mathematical Systems Theory*, Vol. 29, pp. 331–356, 1996.

[亀田 94] 亀田恒彦, 山下雅史. 分散アルゴリズム, アルゴリズムシリーズ, 第 6 巻. 近代科学社, 1994.