通信グラフの形状判定を行う個体群プロトコル

安見　嘉人,　大下　福仁,
井上　美智子
奈良先端科学技術大学院大学

概要

本研究では，個体群プロトコルモデルにおいて，通信グラフの形状判定を行うプロトコルを提案する．個体群プロトコルモデルとは，個体と呼ばれる低能力のデバイスで構成されるネットワーク上の計算を表現する抽象モデルである．本研究で提案する形状判定を行うプロトコルは，任意の通信グラフが与えられた個体群プロトコルモデルにおいて，その通信グラフの形状が所望のグラフ形状であるかどうかを判定する．本研究では，通信グラフの判定対象としてライン・リング・木・スター・二部グラフ・k-正則・完全グラフに着目する．このとき，さまざまな仮定において，形状判定を行うことが可能かを明らかにする．具体的には，1) 初期値が一定か任意か, 2) 全体公平性か弱公平性か, 3) 入力に個体数が与えられるか個体数の上界が与えられるか何の情報も与えられないか, という 3 つの全組み合わせにおいて，各グラフ形状を判定可能かを明らかにする．

1　はじめに

個体群プロトコルモデル (population protocol model)[2] とは，低能力のデバイス（個体，agent）で構成されるネットワーク上の計算を表現する抽象モデルである．このモデルでは，個体がそれぞれ状態を持ち，他の個体と交流を繰り返すことで状態を更新して計算を進める．このとき，どの個体がどの順序で交流するかをシステム自体はコントロールできない．このモデルの応用として，低性能な分子ロボットを用いた分子スケールのシステムが挙げられる．このシステムでは，多数の分子ロボットが協力して体内の情報を収集したり，病気の細胞へ薬を届けたりする応用が考案されている．また，別の応用として，鳥や小動物に取り付けたセンサで構成されるセンサネットワークなども挙げられる．

これまでに個体群プロトコルモデルに対する多くのプロトコルが研究されている [4]．例えば，全体の個体数を求めるプロトコル [5]，リーダー選挙プロトコル [1]，多数決プロトコル [3] などが提案されている．

本稿では，形状判定を行うプロトコルに注目する．これは与えられた通信グラフが所望のグラフ形状かどうかを判定するプロトコルである．形状判定を行うプロトコルにより通信グラフの形状がわかれば，特定の形状で動作するプロトコルが動作可能か知ることができる．

本稿では，さまざまな仮定のもとで，与えられた通信グラフがライン・リング・木・スター・二

表 1　判定に必要な状態数

モデル			グラフ特性	
初期値	公平性	入力	閉路	最大次数 k 以上
一定	全体公平	n	$O(1)$	$O(k)$
		P	$O(1)$	$O(k)$
		なし	$O(1)$	$O(k)$
	弱公平	n	不可	
		P		
		なし		
任意	全体公平/弱公平	n	不可	
		P		
		なし		

表 2　判定に必要な状態数

モデル			グラフ形状
初期値	公平性	入力	ライン・リング・木・二部グラフ
一定	全体公平	n	$O(1)$
		P	$O(1)$
		なし	$O(1)$
	弱公平	n	不可
		P	
		なし	
任意	全体公平/弱公平	n	不可
		P	
		なし	

部グラフ・k-正則・完全グラフであるかどうかを判定できるかをそれぞれ考える．具体的には，1) 初期値が一定か任意か, 2) 全体公平性か弱公平性か, 3) 入力に個体数が与えられるか個体数の上界が与えられるか何の情報も与えられないか, という 3 つの全組み合わせにおいて，各グラフ形状を判定可能かを明らかにする．初期値が一定の場合，実行開始時に個体の初期化が必要となる．初期値が任意の場合，個体の初期化が必要ない．そのため，一時故障が発生して個体が任意の状態になったとしても，いつか目的の状況を達成することができる．弱公平性は各個体間で交流が無限回発生することのみを仮定している．詳細は後述するが，全体公平性は弱公平性よりも強い仮定である．入力に個体数が与えられる仮定では，その入力を利用することで効率的なプロトコルを作成できる．一方で，個体数の上界が与えられる仮定では各個体は個体数を知らない．ただし，個体数の上界を知っている．入力として何の情報も与えられない仮定では，各個体は個体数も個体数の上界も知らない．

表 3　各グラフ判定に必要な状態数

モデル			グラフ形状			
初期値	公平性	入力	ライン・リング 木・二部グラフ	k-正則	スター	完全
一定	全体公平	n	$O(1)$	$O(k \log n)$	$O(1)$	$O(n)$
		P	$O(1)$	$O(k \log P)$	$O(1)$	$O(P^2)$
		なし	$O(1)$		$O(1)$	
	弱公平	n	不可		$O(n)$	$O(n)$
		P	不可			
		なし				
任意	全体公平/ 弱公平	n	不可			
		P				
		なし				

本研究の結果を表 1, 2, 3 に示す．表 1 はいくつかのグラフ特性の判定に必要な状態数を表している．ここで P は個体数の上界，n は個体数を表している．また，不可はその仮定の下で判定を行うことが不可能であることを示している．閉路は通信グラフに閉路が存在するかどうかを判定する．最大次数 k 以上は通信グラフに次数 k 以上を持つ個体が存在するかどうかを判定する．

これらのプロトコルの組み合わせで表 2 の結果が得られる．つまり，ライン・リング・木・二部グラフの判定が可能となる．例えば，次数 3 以上の個体が存在しないと判定された場合，閉路が存在すると判定されればその通信グラフはリングであり，閉路がないと判定されればその通信グラフはラインである．また，閉路が存在しないと判定されればその通信グラフは木である．

さらに，本研究では k-正則, スター, 完全グラフの判定を行うプロトコルも提案する．よって，本研究での包括的な結果は表 3 に示される．ここで，空白部分はまだ明らかになっていないことを表している．

2　諸定義

個体群プロトコルモデルでは，個体群と呼ばれる連結グラフ $G = (V, E)$ によってシステムが構成される．ここで，V は個体の集合であり，$E \subseteq V \times V$ は辺の集合である．各辺は交流可能かどうかを表している．つまり，もし個体 $u \in V$ と $v \in V$ が交流可能であれば，その場合に限り $(u, v) \in E$ である．

プロトコル P は，各個体の取りうる状態 (state) の集合 Q と Q 上の遷移の集合の組で定義される．各遷移は，$(p, q) \to (p', q')$ の形で表され，これは状態 p を持つ任意の個体 x と状態 q を持つ任意の個体 y が交流したとき，x は状態 p から状態 p' に変わり，y は状態 q から状態 q' に変わることを表す．プロトコル P において，すべての状態のペア (p, q) に対して，ただひとつの遷移 $(p, q) \to (p', q')$ が存在するならば，P は決定性のプロトコルであるという．また，$p \neq q$ であり $(p, q) \to (p', q')$ が存在するならば，$(q, p) \to (q', p')$ が存在するとする．本稿では決定性のプ

ロトコルのみを考慮する．

個体群の大域的な状態を状況 (configuration) と呼び，すべての個体の状態を要素としたベクトルで表す．ある状況 C と C' において，C' が個体のペアの一回の交流によって C から得られるとき，$C \to C'$ と表す．これは，状況 C において状態 p の個体 x と状態 q の個体 y が存在し，ある遷移 $(p, q) \to (p', q')$ によって，個体 x と y の状態が p' と q' へ変わった状況 C' に遷移することを表す．

2 つの状況 C と C' に対して, もし各 i $(0 \leq i < m)$ に対して $C_i \to C_{i+1}$ を満たすような状況列 $C = C_0, C_1, \ldots, C_m = C'$ が存在するならば，C' は C から到達可能と呼び, $C \xrightarrow{*} C'$ と表す．無限状況列 $\Xi = C_0, C_1, C_2, \ldots$ がそれぞれの $i(i \geq 0)$ において $C_i \to C_{i+1}$ を満たすとき，Ξ をプロトコルの実行という．ある実行 Ξ において，交流可能な全ての個体同士が無限回交流するとき，Ξ を弱公平な (weakly fair) 実行という．ある実行 Ξ において，C が無限回発生するならば，$C \to C'$ が成り立つすべての C' も無限回発生するとき，Ξ を全体公平な (globally fair) 実行という．

個体の数を n と表し，個体の数の上界を P と表す．V_p を全ての個体の集合とし，個体 $a \in V_p$ に対して，s_a を a の状態とする．

2.1 判定問題

本稿で考えるグラフ $G = (V, E)$ に対する各グラフ特性を以下のように定義する．

- 閉路: G 上に閉路が存在する．
- 最大次数 k 以上: ある個体 $a \in V$ の次数 l が $l \geq k$ を満たす．
- k-正則: V に含まれる全個体の次数が k である．
- スター: ある個体 $a \in V$ の次数が $n-1$ であり，$V - \{a\}$ に含まれる全個体の次数が 1 である．
- 完全: V に含まれる全個体の次数が $n-1$ である．

$f : Q \to \{yes, no\}$ を個体の状態を yes と no へ関連付ける関数とする．あるグラフ特性 X に対して, 実行 $\Xi = C_0, C_1, C_2, \ldots$ が X 判定問題を解くとは，以下の条件を満たす t が存在することである．

1. 与えられたグラフ $G = (V, E)$ がグラフ特性 X を満たすときに, 全ての $t' \geq t$ で状況 $C_{t'}$ において，$\forall a \in V : f(s_a) = yes$ が成立
2. 与えられたグラフ $G = (V, E)$ がグラフ特性 X を満たさないときに, 全ての $t' \geq t$ で状況 $C_{t'}$ において，$\forall a \in V : f(s_a) = no$ が成立

プロトコル P の全ての実行 Ξ が X 判定問題を解くとき，プロトコル P は X 判定問題を解くと定義する．

3　形状判定プロトコル

本稿では，いくつかの判定プロトコルを示す．具体的には, 初期値一定で全体公平性の仮定の下で定数状態で動作する閉路判定プロトコルと最大次数 k 以上判定プロトコルを示す.

3.1　閉路判定プロトコル

本節では，初期値一定で全体公平性の仮定の下で定数状態で動作する閉路判定プロトコルを示す.

このプロトコルの基本的な戦略は以下の通りである.

- グラフ中を自由に移動する 3 つのトークンを選出する.
- その内の 2 つが定期的に隣り合って停止し，残りの 1 つはそれらを目印として閉路を探す. 停止トークンは隣り合って停止するため，左右を持つ目印となる．閉路がない場合，探索トークンが左のトークンと交流した後は，その左のトークンとすれ違わない限り右のトークンとは交流できない．一方で，閉路がある場合は，探索トークンが左のトークンと交流した後に閉路を迂回して右のトークンと交流することができる．この違いを利用して，閉路の存在性を判定する.
- 閉路が存在するとわかれば，探索トークンがそれを全個体にそれを伝える.

このプロトコルが持つ変数 2 つは定数個の値を取るので，このプロトコルは定数状態プロトコルである．プロトコルの疑似コードを Algorithm 1 と 2 に示す．以下に各状態変数とその役割を示す.

- $LF \in \{L_1, L_2, L_3, L_1^d, L_1^s, L_1^{d'}, L_2^s, L_3^s, F\}$: トークン変数であり, 初期値は L_3 を持つ. L がトークンを表している．トークンには L_1 トークン，L_2 トークン, L_3 トークンの 3 つがある．L_1 トークンが閉路を探索するトークンであり，L_2, L_3 トークンが停止するトークンを表す．L_2^s, L_3^s はそれぞれ，停止している L_2 トークン, L_3 トークンを表す．詳細は後述するが，L_1^d は探索している L_1 トークンを表し，$L_1^s, L_1^{d'}$ は L_2 トークン, L_3 トークンの停止を制御するための L_1 トークンの状態を表す．F はトークンを持たない個体を表す.
- $cycle \in \{yes, no\}$: 閉路判定の変数であり，初期値は no を持つ． 個体 a に対して $cycle_a = yes$ であれば $f(s_a) = yes$(つまり，閉路が存在すると判定）であり, $cycle_a = no$ であれば $f(s_a) = no$(つまり，閉路が存在しないと判定）である．本プロトコルでは，閉路があるかの判定は L_1 トークンが行い，それを全個体に伝える．つまり，L_1 トークンと交流した個体の $cycle$ は L_1 トークンが持つ $cycle$ の値に更新される.

以下から, プロトコルの詳細な説明を行う．2-8 行目は 3 つのトークンの選出動作を表している．全個体は最初 L_3 トークンを持った状態から始まる．L_3 同士による交流が行われるとその内の一方が L_2 トークンになる．同様に L_2 同士が交流すると一方が L_1 トークンになり，L_1 同士が交流すると一方のトークンが消える．トークンは 9-16 行目の動作でグラフ中を移動するため，全体公平性から同じトークンがある限り，それらによる交流が発生し続け，最終的に各トークン

Algorithm 1 閉路判定プロトコル (Part 1)

A variable at a mobile agent a:

$LF_a \in \{L_1, L_2, L_3, L_1^d, L_1^s, L_1^{d'}, L_2^s, L_3^s, F\}$: トークンを表す変数, 初期値は L_3.

$cycle_a \in \{yes, no\}$: 閉路判定の変数, 初期値は no.

1: **while** two mobile agent a and b interact **do**

{ トークンの選出 }

2: **if** $LF_a, LF_b \in \{L_3^s, L_3\}$ **then**

3: $LF_b \leftarrow L_2$

4: **else if** $LF_a, LF_b \in \{L_2^s, L_2\}$ **then**

5: $LF_b \leftarrow L_1$

6: **else if** $LF_a, LF_b \in \{L_1^d, L_1^{d'}, L_1^s, L_1\}$ **then**

7: $LF_b \leftarrow F$

8: $cycle_a \leftarrow no$, $cycle_b \leftarrow no$

{ トークンの移動 }

9: **else if** $LF_a \neq F \wedge LF_b = F$ **then**

10: **if** $LF_a \in \{L_1^d, L_1^{d'}, L_1^s, L_1\}$ **then**

11: $cycle_b \leftarrow cycle_a$

12: **if** $LF_a = L_k^s (k \in \{1, 2, 3\})$ **then**

13: $LF_a \leftarrow L_k$

14: **else if** $LF_a = L_1^{d'}$ **then**

15: $LF_a \leftarrow L_1$

16: $LF_a \leftrightarrow LF_b$ *

{ 判定 }

17: **else if** $LF_a = L_1 \wedge LF_b = L_2$ **then**

18: $LF_a \leftarrow L_2^s$, $LF_b \leftarrow L_1^s$

19: $cycle_b \leftarrow cycle_a$

20: **else if** $LF_a = L_1^s \wedge LF_b = L_3$ **then**

21: $LF_a \leftarrow L_3^s$, $LF_b \leftarrow L_1^d$

22: $cycle_b \leftarrow cycle_a$

23: **else if** $LF_a = L_1^d \wedge LF_b = L_2^s$ **then**

24: $LF_a \leftarrow L_2$, $LF_b \leftarrow L_1^{d'}$

25: $cycle_b \leftarrow cycle_a$

26: **else if** $LF_a = L_1^{d'} \wedge LF_b = L_3^s$ **then**

27: $LF_a \leftarrow L_3$, $LF_b \leftarrow L_1$

28: $cycle_a \leftarrow yes$, $cycle_b \leftarrow yes$

* $p \leftrightarrow q$ は p と q の値の交換を意味する.

▷ Part2 に続く

Algorithm 2 閉路判定プロトコル (Part 2)

29: **else if** $LF_a \neq F \wedge LF_b \neq F$ **then**
30: **if** $LF_a = L_1$ **then**
31: $cycle_b \leftarrow cycle_a$
32: $LF_a \leftrightarrow LF_b$
33: **if** $LF_a = L_k^s (k \in \{1,2,3\})$ **then**
34: $LF_a \leftarrow L_k$
35: **if** $LF_b = L_k^s (k \in \{1,2,3\})$ **then**
36: $LF_b \leftarrow L_k$
37: **if** $LF_a = L_1^d \vee LF_a = L_1^{d'}$ **then**
38: $LF_a \leftarrow L_1$
39: **if** $LF_b = L_1^d \vee LF_a = L_1^{d'}$ **then**
40: $LF_b \leftarrow L_1$

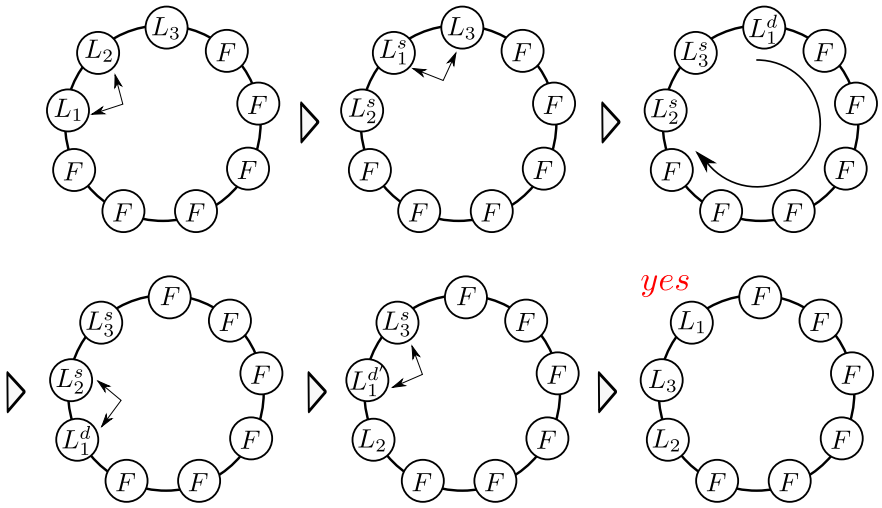
が 1 つずつ選出される．また，トークンが減るときには, L_1 トークンの判定が no にリセットされる (6-8 行目)．よって，各トークンが 1 つずつになるとき，L_1 トークンの判定はリセットされ，no を持つ.

9-16 行目のトークンの移動動作では, トークンを持っていない個体とトークンを持っている個体が交流したときにトークンの移動を行う動作を示している．具体的には, 16 行目がトークンの移動を表している．10-11 行目では L_1 トークンの判定の伝達を行っている．他の部分の説明は判定動作の説明後に行う.

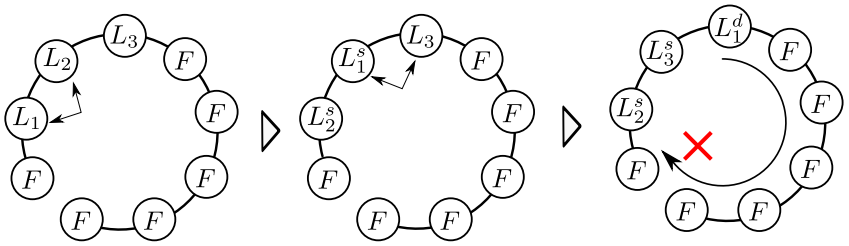
17-28 行目の判定動作では，閉路判定を行う．図 1 に判定時の動作例を示す.

L_1 が L_2, L_3 と連続で交流し (19-24 行目)，閉路上を一周してきて L_2^s と L_3^s と (つまり，L_2, L_3 が L_1 と初めに交流した位置のまま) 交流した場合にのみ閉路があると判定する．上記の試行を繰り返す．試行が失敗した場合にはトークンの状態はリセットされる（この動作はトークン移動動作の 12-15 行目や 29 行目以降に該当し，後ほど説明する)．ここで，L_1^s は L_2 と L_3 を隣り合わせに停止させるために，一時的に L_1 が L_3 を待つために停止している状態である．また，$L_1^{d'}$ は L_2^s を見つけた後に L_3^s を待つために停止している状態である．つまり，$L_1^{d'}$ は L_2^s を見つけたときに L_3 トークンが停止中であるか確認するための状態である．後述するが, L_3^s の停止状態が解除される場合があるため, この確認は必要となる．また，L_1^d は L_2 トークンと L_3 トークンを停止させた L_1 トークンを表しており，閉路を通り L_2^s と交流するまで，グラフ中を移動する.

つづいて，トークン移動動作の 12-15 行目について説明する．12-15 行目では，$L_1^s(L_1^{d'})$, L_2^s, L_3^s をそれぞれ L_1, L_2, L_3 に戻す動作を行っている．これは停止しているトークンがトークンを持たない個体と交流したときに停止が解除されることを示唆している．この動作が行われた後は, 17-28 行目の判定条件のいずれかを満たさなくなるため，閉路判定の試行は失敗である．したがって，閉路判定の試行は停止状態の個体が 17-28 行目で示されているような所定の交流のみ行うときに成功する．全体公平性から，閉路が存在する場合は閉路判定の試行はいつか成功するため，閉路があるという判定が行われる．注意点として, L_1^d はトークンを持たない個体と交流しても



(a) 閉路あり



(b) 閉路なし

図 1　閉路判定の動作例

L_1 には戻らないことが挙げられる．これは，L_1^d は停止せずにグラフ中を移動する必要があるためである．L_1^d が閉路判定の試行が失敗したとわかり，L_1 に戻るのは 29 行目以降の動作となる．

29 行目以降では, 閉路判定で使われる組み合わせ以外でのトークン同士の交流の動作を表している．この場合にも, 閉路判定の試行は失敗とし，各トークンを L_1, L_2, L_3 に戻す．L_1^d はこの動作で所定のトークン (L_2^s) と交流できなかったときに，L_1 にリセットされる．

3.2　最大次数 k 以上判定プロトコル

本節では，初期値一定で全体公平性の仮定の下で $O(k)$ 状態で動作する最大次数 k 以上判定プロトコルを示す．

このプロトコルの基本的な戦略は以下のとおりである．

- リーダトークンを 1 つ決める．そのリーダトークンを持つ個体は隣の個体をマークしていき，自身の次数が k 以上かどうか調べる試行を行う．マークを行うごとにカウントを行い，k 回連続でカウントできたら, 次数 k 以上の個体が存在すると判定する．
- リーダトークンは k 回カウントする前にマークした個体と交流したとき，試行が失敗した

Algorithm 3 最大次数 k 以上判定

A variable at a mobile agent a:

$LF_a \in \{L_0, L_1, L_2, \ldots, L_{k-1}, F, F'\}$: リーダートークン変数, 初期値は L_0. L はリーダートークンを意味する.

$degree_a \in \{yes, no\}$: 最大次数 k 以上判定の変数, 初期値は no.

1: **while** two mobile agent a and b interact **do**

{ リーダートークンの選出 }

2: **if** $LF_a = L_i \wedge LF_b = L_j$ $(i, j \in \{0, 1, 2, \ldots, k-1\})$ **then**

3: $LF_a \leftarrow L_0$, $LF_b \leftarrow F$

4: $degree_a \leftarrow degree_b \leftarrow no$

{ 判定・移動 }

5: **else if** $LF_a = L_i \wedge LF_b = F$ $(i \in \{0, 1, 2, \ldots, k-2\})$ **then**

6: $LF_a \leftarrow L_{i+1}$, $LF_b \leftarrow F'$

7: **else if** $LF_a = L_i \wedge LF_b = F'$ $(i \in \{0, 1, 2, \ldots, k-1\})$ **then**

8: $LF_a \leftarrow F$, $LF_b \leftarrow L_0$

9: $degree_b \leftarrow degree_a$

10: **else if** $LF_a = L_{k-1} \wedge LF_b = F$ **then**

11: $LF_a \leftarrow L_0$, $LF_b \leftarrow F'$

12: $degree_a \leftarrow degree_b \leftarrow yes$

とし，マークを消しトークンが移動する．これにより, トークンはグラフ中を自由に移動できる.

- 全体公平性から，次数 k 以上の個体が存在する場合，いつか全ての隣接個体のマークが消えた状態で次数 k 以上の個体がリーダートークンを持ち，試行が成功する.
- リーダトークンは判定を全個体に伝える.

このプロトコルは，定数個の値を取る変数と $k+2$ 個の値を取る変数を持つので，$O(k)$ 状態プロトコルである．プロトコルの疑似コードを Algorithm 3 に示す．以下に各状態変数とその役割を示す.

- $LF \in \{L_0, L_1, L_2, \ldots, L_{k-1}, F, F'\}$: リーダートークンを持つか表す変数であり, 初期値は L_0 を持つ. L がリーダートークンを表している．L_x はリーダートークンが x 個の次数を数えていることを表す．F はリーダートークンを持たない個体を表し, F' はリーダートークンに既に数えられたことを表す.
- $degree \in \{yes, no\}$: 最大次数 k 以上判定の変数であり, 初期値は no を持つ. 個体 a に対して $degree_a = yes$ であれば $f(s_a) = yes$(つまり，次数 k 以上の個体が存在すると判定）であり, $degree_a = no$ であれば $f(s_a) = no$(つまり，次数 k 以上の個体が存在しないと判定）である．本プロトコルでは，最大次数 k 以上判定はリーダートークンが行い，それを全個体に伝える．つまり，リーダートークンと交流した個体の $degree$ はリーダートークンが持つ $degree$ の値に更新される.

以下から, プロトコルの詳細な説明を行う. 2-4 行目はリーダートークンの選出動作を表している. 具体的には, リーダートークンを持つ個体同士が交流したときに一方のトークンが削除される動作を行っている. またこのとき, リーダートークンの判定が no にリセットされる. これはトークン選出がまだ終わっていなかったことを意味し, リーダートークンの判定が全体に伝えられるため, このリセットにより実質的に個体群全体の判定のリセットを実現している.

5-12 行目はリーダートークンの移動動作と判定動作を表している. リーダートークンは F と交流すると F にマークを付け (F'), 数えている次数のカウントを増加させる (5-6 行目). このカウントが k になったら, 現在リーダートークンを持っている個体の次数が k 以上あると判断し, $degree$ を yes に更新する (10-12 行目). マークしている個体と交流すると, リーダートークンは移動し, マークを消す (7-9 行目). これらの動作から, リーダートークンはマークのない個体にもマークをつけて移動することができるため, グラフ中を自由に移動できる. また移動時にマークを消すため, 全体公平性から, 次数 k 以上の個体があれば, いつかその個体がリーダートークンを持ち, 隣接個体がマークを持たない状況に遷移する. さらに, 全体公平性から, その状況からマークを持たない個体と k 連続で交流することが起こるため, 最大次数 k 以上が存在すると判定される.

4 結論

本稿では, 個体群プロトコルモデルにおけるグラフの形状を判定するプロトコルに着目した. 具体的には, さまざまな仮定の下で, 与えられた通信グラフがライン・リング・木・スター・二部グラフ・k-正則・完全グラフであるかどうかを判定できるかをそれぞれ考えた.

今後の課題として, 以下が考えられる.

- 時間効率のよいプロトコルの提案
- より複雑なグラフ形状の判定 (グリッド, トーラス, カクタス, 木幅の計算など)

参考文献

[1] Dana Angluin, James Aspnes, Melody Chan, Michael J Fischer, Hong Jiang, and René Peralta. Stably computable properties of network graphs. In *Proc. of International Conference on Distributed Computing in Sensor Systems*, pages 63–74, 2005.

[2] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed computing*, 18(4):235–253, 2006.

[3] Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008.

[4] James Aspnes and Eric Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, pages 97–120, 2009.

[5] Joffroy Beauquier, Janna Burman, Simon Claviere, and Devan Sohier. Space-optimal

counting in population protocols. In *Proc. of International Symposium on Distributed Computing*, pages 631–646, 2015.