個体群プロトコルモデルにおけるリーダ選挙

技術解説

首　藤　裕　一*

Leader Election in Population Protocol Model

Key Words：Anonymous networks, Sensor networks, Self-stabilization, Loose-stabilization

1．はじめに

昨今，インターネットやセンサネットワーク，アドホックネットワークなど，複数の計算機で構成される分散システムが普及している．本稿では，分散システムの計算モデルのひとつとして近年活発に研究が行われている**個体群プロトコルモデル**を紹介し，分散システムにおける重要かつ基本的な問題である**リーダ選挙問題**を本モデルにおいて解決するための手法を紹介する．

分散システムは一般に多数の計算機で構成されるために，各ノードにおける故障の発生や性能劣化などにより，機能面・性能面で不安定になりやすいという特徴がある．そのため，故障発生によってシステムが不安定になっても自律的にシステムが正しい状態に回復する**自己安定**と呼ばれる性質をプロトコル（＝アルゴリズム）に持たせることが重要である．しかしながら，個体群プロトコルモデルにおいて自己安定なリーダ選挙プロトコルを設計することは不可能であることが知られている．本稿では，通常のリーダ選挙プロトコルに加え，実用上は自己安定と変わらない故障耐性を保証する**緩安定**と呼ばれる性質をリーダ選挙プロトコルに持たせる手法を紹介する．

* Yuichi SUDO
1986年5月生まれ
大阪大学　大学院情報科学研究科　博士前期課程修了（2011年）
日本電信電話（株）平成23～28年勤務，
西日本電信電話（株）平成28～29年勤務．
現在、大阪大学　大学院情報科学研究科
コンピュータサイエンス専攻　助教
博士（情報科学）　自律分散アルゴリズム
グラフアルゴリズム　ネットワークセキュリティ
TEL：06-6879-4117
FAX：06-6879-4119
E-mail：y-sudou@ist.osaka-u.ac.jp

2．個体群プロトコルモデル

個体群プロトコルモデル（Population protocols）とは，計算資源が極めて制約されたセンサノードによって構成されるモバイルセンサネットワークを表現する計算モデルとして2004年にAngluinら[1]によって提案された．本モデルではネットワークを構成するセンサノードを**個体**と呼び，ネットワーク全体を**個体群**と呼ぶ．各個体は有限状態機械であり，常にひとつの状態を持つ．2つの個体間で互いに自身の状態を伝え合う交流と呼ばれるイベントがときおり発生し，その際，両個体の現在の状態と，プロトコル（＝アルゴリズム）が定める状態遷移規則によって，両個体は新たな状態に遷移する．プロトコルの実行はこの交流によって進行する．後述するように，個体群中の任意の個体ペアは幾度となく交流を繰り返すことを仮定する．また，個体群を構成する個体の数（個体数）をnで表す．

ここで，個体群プロトコルモデルを直感的に理解するために，簡単な応用例として，千羽の鳥の群れの一羽一羽に小型のセンサノードすなわち個体を付けた個体群（個体数$n = 1000$）を考える．各鳥が好き勝手に移動することで，2羽の鳥が極めて近い距離に接近する事象が頻繁に発生する．この事象が交流に相当する．交流の発生時，無線通信によって個体は互いの状態を教え合い，自身の状態を更新する．このような例において，任意の個体ペアが幾度となく交流を繰り返すという前述の仮定は自然である．この個体群で簡単な計算を行うことを考える．具体的には，1000羽の鳥のうち，発熱している（体温が閾値を超えている）鳥が10羽以上いるかどうかを全個体に把握させるという問題を解くことを考える．実行開始時，観測対象の鳥が発熱している個体は初期状態を1に，そうでない個体は初期状態を0に設定する．交流発生時の状態遷移規則は以下の

ように定める．

$$(x, y) \to (x + y, 0) \quad \text{if } x + y < 10$$
$$(x, y) \to (10, 10) \quad \text{otherwise}$$

上記の規則は，状態xの個体と状態yの個体が交流したときに，前者と後者の次状態を指定する規則である．第一の規則は，xとyの和が10に満たないときに，2個体の状態を加算して一方に記憶させ，他方の状態を0にする．第二の規則は，2個体の状態の和が10以上であるときに，両個体の状態をともに10にする．いま，このプロトコルの実行開始時において発熱している個体の総数をfとする．第二の規則が適用されるまでは，全個体の状態の総和とfは常に一致する．したがって，$f < 10$のとき第二規則は決して適用されることがなく，すべての個体の状態は10未満でありつづける．一方で，任意の個体ペア間で交流が繰り返し発生するため，発熱している個体の総数は第一規則によって少数の個体に集約されていく．したがって，$f \geq 10$のときは，必ず第二規則がいつか適用され状態10の個体が少なくともひとつ発生する．状態10の個体がひとつでも生じると，第二規則により状態10の個体は増殖し，やがて全個体が状態10となることが保証できる．ゆえに，「1000羽の鳥のうち，発熱している鳥が10羽以上いるか」という問いに対して，状態10の個体は“Yes”と出力し，それ以外の状態の個体は“No”と出力すれば，全個体がやがて正しい答えを出力することを保証できる．このようなプロトコルが個体群プロトコルである．

上述のプロトコルの第一規則は2個体の対称性を破壊していることに注意されたい．すなわち，同一の状態を持つ2個体が交流した場合においても2個体が異なる状態に遷移する．これは，交流に相当する無線通信において，交流を呼びかけた個体とその呼びかけに応答した個体が存在するはずだという仮定に基づくものであり，両個体の状態が同一であっても，**呼びかけ側**の個体（initiator）と**応答側**の個体（responder）の次状態は個別に決定して良いという意味でプロトコルの設計者に強い自由度を与える．したがって，他の一部の（分散システムの）計算モデルと異なり，システム全体の対称性は個体群プロトコルモデルでは問題にならないことが多い．

交流の発生パターンについてはさまざまなモデルがあるが，個体群プロトコルを扱う多くの研究では，一様ランダムな確率的スケジューラを仮定する．これは，各時刻（＝ステップ）において，個体群中から一様ランダムに選択された2つの個体間で交流が発生するという仮定である．すなわち，各時刻において，任意の個体ペアは確率$1/{}_nC_2$で選択され交流を行う．ここでいう時刻とは抽象的なものであり，実際には，交流は個体群中で同時多発的に発生するため，後に説明する期待収束時間などのプロトコルの時間計算量は，計算の完了に要したステップ数を個体数nで割った**並列時間**で評価することが主流である．本稿においても，この確率的スケジューラを仮定し，各種時間計算量を並列時間で評価する．

個体群プロトコルで計算可能な述語のクラスはAnglulinら[1,2]によって正確に分析されている．具体的には，個体群に与えられる入力を引数としてある述語を解こうとするとき（前述の例で言えば，発熱している鳥の総数fを引数とする述語$f \geq 10$を解く），その述語がPresburger算術[3]で表現可能なとき，かつ，そのときのみその述語を解く個体群プロトコルが存在することが証明されている．一方で，個体群に与えられる入力とは独立に定義される個々のタスクを解くことができるかどうか，あるいは，そのタスクを解くのにどの程度の時間を要するのかということは実用上・理論上興味深い問いであり，この方面で盛んに研究が行われている．

3．リーダ選挙

入力とは独立に定義されるタスク（問題）としてもっとも重要なもののひとつにリーダ選挙問題がある．これは，個体群中のただひとつの個体をリーダとして指名する問題である．リーダ選挙問題は分散システム一般にとって重要かつ基本的な問題であるが，個体群プロトコルモデルでは特に重要な意義を持つ．というのは，個体群中にただひとつのリーダが存在することを仮定できれば，Presburger算術で定義可能な任意の述語が極めて高速に，具体的には個体数nに関する対数多項式並列時間で解けることが2006年に証明されているからである[4]．

リーダ選挙問題は，厳密には，一様ランダムなスケジューラによって交流が次々と発生していく結果として，やがてただひとつの個体が記号Lを出力し，それ以外のすべての個体が記号Fを出力するような

状況に個体群を収束させることを目的とする問題である．ここでいう状況とは，個体群全体の状態のことであり，具体的には，全個体の状態で構成されるn次元ベクトルである．リーダ選挙問題は次に示す極めて単純なプロトコル [1] によって解くことができる．

状態空間：$\{l, f\}$
初期状態：l
出力：状態がlのときL，状態がfのときF
状態遷移規則：$(l, l) \to (l, f)$

このプロトコルにおいて各個体はl, fのいずれかの状態をとる．実行開始時の初期状態は入力にかかわらずlである．各個体は，自身の状態がlのとき，かつ，そのときのみ記号Lを出力し，そうでないとき記号Fを出力する．すなわち，状態lの個体がリーダであり，状態fの個体が非リーダである．交流による状態遷移は両個体がリーダのときのみ発生する．リーダどうしが交流したとき，一方（呼びかけ側）がリーダであり続け，他方（応答側）は非リーダになる．本プロトコルの実行において，実行開始時にn個のリーダが存在し，かつ，リーダの数は2個以上のリーダが交流して一方が非リーダに変化することによってのみ減少するので，リーダの数が0になることは決してない．一方で，2個以上のリーダが存在する場合，やがてそれらのリーダ間で交流が発生してリーダの数が1減る．したがって，個体群中のリーダの数はやがて1になる．以後，ただひとつのリーダはリーダであり続け，他の個体は常に非リーダであり続ける．よって，本プロトコルは個体群中からただひとつのリーダを選出する．すなわち，正しくリーダ選挙問題を解く．

ここで，このプロトコルの収束時間の期待値（**期待収束時間**）を考察する．上述の確率的スケジューラの仮定により，各時刻（ステップ）において任意の個体ペアは確率$1/{}_nC_2$で交流する．したがって，リーダの数がi個の状況において，1ステップでリーダ同士が交流してリーダの数が1減少する確率は${}_iC_2/{}_nC_2 = i(i-1)/n(n-1)$である．ゆえに，ただひとつのリーダが選出されるまでに要するステップ数の期待値は次式で得られる．

$$\begin{aligned}\sum_{i=2}^{n} \frac{n(n-1)}{i(i-1)} &= n(n-1)\sum_{i=2}^{n}\left(\frac{1}{i-1}-\frac{1}{i}\right)\\ &= n(n-1)\left(1-\frac{1}{n}\right)\\ &= O(n^2)\end{aligned}$$

2節で述べたとおり，期待収束時間は，収束に要するステップ数をnで除して得られる並列時間として評価する．したがって，期待収束時間は$O(n)$である．先述の通り，個体群中にただひとつのリーダが存在することを仮定したうえで，Presburger算術で定義可能な任意の述語を対数多項式の期待並列時間で解くプロトコルが存在する．一方で，ただひとつのリーダが存在するというこのプロトコルの仮定を除去するために上述のリーダ選挙プロトコルを用いたのでは，リーダの選出までに線形時間を要するので元のプロトコルの高速性を失ってしまう．したがって，対数多項式の期待並列時間で収束するリーダ選挙プロトコルを設計することが極めて重要となる．2006年以降，長らくそのようなプロトコルは提案されてこなかったが，2014年にAlistarhとGelashvili[5]によって$O(\log^3 n)$期待並列時間で収束するリーダ選挙プロトコルがはじめて考案された．上述のリーダ選挙プロトコルが2状態で動作するのに対し，この新しいプロトコルは$O(\log^3 n)$状態を必要とする．すなわち，定数ビットのメモリでは実装できなくなる．2004年に提案された当初の個体群プロトコルモデルは各個体が取り得る状態の数を定数個に限定しており，その意味でこの高速なリーダ選挙プロトコルは正規の個体群プロトコルから逸脱している．しかしながら，定数個の状態によるリーダ選挙プロトコルは線形時間の期待収束時間を要することが証明されており [6]，このプロトコルが考案された意義は極めて大きい．さらに，2018年には，Berenbrinkら [7] によって$O(\log^2 n)$並列時間で収束する状態数$O(\log n)$のリーダ選挙プロトコルが考案された．

4．緩安定リーダ選挙

個体群プロトコルモデルは計算資源が非常に制約された多数の計算機（＝個体）で構成されるネットワーク（＝個体群）を想定している．したがって，個体群中の一部の個体においてメモリの改変等の故障が発生することは自然であり，そのような故障が

発生したとしても正しく機能し続けるような，故障耐性の高いプロトコルを設計することは非常に重要である．

分散システム一般において故障耐性の高いプロトコルを設計する手法として，Dijkstra が 1974 年に提唱した**自己安定**の概念は極めて有用である．分散システム上で動作するプロトコルは，次の 2 つの要件を満たすとき自己安定であるという．

1. **収束性**：任意の初期状況から実行を開始しても，システムはやがて**安全状況**と呼ばれる状況に到達する．
2. **閉包性**：一度システムが安全状況に到達すると，それ以降，システムは永遠に所望の性質を満たし続ける．

自己安定なプロトコルは，どのような一時故障（記憶情報の喪失・改変）などが発生したとしても，プロトコルを記述するプログラムコードが改変されないという保証さえあれば（e.g., ハードウェア実装，耐タンパ領域への記憶），やがてその故障から自律的に回復するという優れた故障耐性を有する．なぜならば，システムを構成する一部あるいは全部の計算機において故障が発生し，システムがでたらめな状況に陥ったとしても，収束性および閉包性によってシステムは正常な機能を回復し，新たな故障が発生するまで正常な機能を維持することができるからである．（新たな故障が発生した後も，再び収束性・閉包性によって回復する．）しかし，多くの場合，自己安定プロトコルは高いコスト（時間計算量・空間計算量）を要し，いくつかの問題に対しては，自己安定プロトコルの設計自体が不可能であることもある．

残念ながら，個体群プロトコルにおける自己安定なリーダ選挙プロトコルの設計についても強い不可能性が知られている．2006 年，Angluin ら [8] は，個体群中の各個体が個体数 n を正確に把握していない限り自己安定リーダ選挙プロトコルは実現できないことを証明した．これは，たとえば，個体数 10000 の個体群で動作する自己安定リーダ選挙プロトコルは，個体数がちょうど 10000 の個体群でしか動作せず，個体数が 9999 や 10001 に変化した瞬間に正しい機能を継続できないことを意味する．膨大

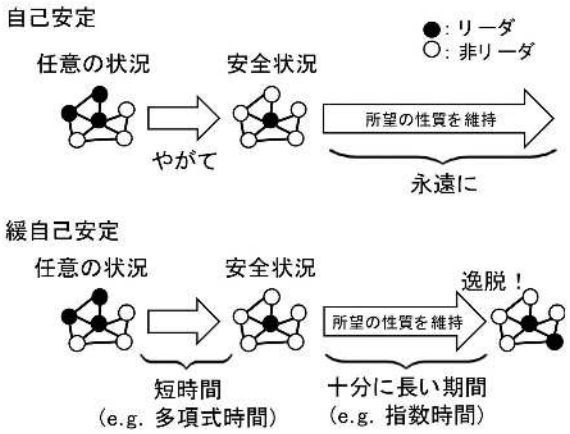

図1　自己安定と緩安定（リーダ選挙の場合）

な数の個体で構成される個体群において個体数を正確に把握することは実用上極めて困難であり，この不可能性は実用的な自己安定プロトコルの実現が不可能であることを意味している．

この不可能性に対処するため，2009 年に筆者ら [9] は，自己安定の 2 要件のうち，閉包性の条件を次のように緩めた**緩安定**という新たな概念を導入した．

2'. 緩安定における閉包性：一度システムが安全状況に到達すると，それ以降システムは極めて長いあいだ所望の性質を満たし続ける．

自己安定と緩安定の差異を図 1 に示す．緩安定プロトコルは，閉包性の要件における，システムが所望の性質を満たし続ける期間が十分に長ければ（たとえばネットワークサイズに関して指数関数的な時間であれば），実用上，自己安定プロトコルと同等の故障耐性を有するとみなせる．この緩安定の概念は抽象的なものであり，とりわけ上記要件における「極めて長いあいだ」という表現は曖昧である．実際には，緩安定の概念は個々の計算モデルに応じて厳密に定義される．個体群モデル上での緩安定リーダ選挙プロトコルは次のように定義される．

【定義】プロトコル P は，次の 3 条件を満たす安全状況の集合 S が存在するとき (α, β)-緩安定リーダ選挙プロトコルであるという．
(i) P の状態空間によって定義される任意の状況 C から開始する P の実行において，個体群が状況 C から S 中のいずれかの状況に到達するまでの期待ステップ数は α 以下である．

(ii) S 中の任意の状況において個体群中のただひとつの個体がリーダであり（記号Lを出力し），その他の全個体が非リーダである（記号Fを出力する）．

(iii) S 中の任意の状況 C から開始する P の実行において，個体群が唯一のリーダを維持し続ける期待ステップ数（C における唯一のリーダが非リーダとなる，あるいは，他の個体がリーダになるまでの期待ステップ数）は β 以上である．

(α, β)-緩安定リーダ選挙プロトコルにおいて，α/n を期待収束時間，β/n を期待維持時間と呼ぶ．いずれも並列時間で定義するために α，β を個体数 n で除している．当然，α は短ければ短いほど，β は長ければ長いほど優れたプロトコルであるといえる．

以降では，筆者らが2009年に緩安定の概念とともに提案した，個体数 n の上界 N が既知であるという仮定のもとで動作する，期待収束時間 $O(N \log n)$，期待維持時間 $\Omega(e^N)$ の緩安定リーダ選挙プロトコル [9] を紹介する[1]．個体数 n の上界 N が既知であるという仮定のもと動作するプロトコルは，個体数 n が N 以下の任意の個体群で動作するので，個体数 n を正確に把握する必要があるプロトコル（前述の通り個体数 n が既知であることは自己安定リーダ選挙の必要条件）に比べ，実用上はるかに有用である．また，期待維持時間が個体数 n に関して指数関数的に大きい値であるため，元来の定義による自己安定プロトコルと実用上は同等の故障耐性を有するといえる．

本プロトコルにおいて，各個体の状態は，記号 l, f のいずれかを表現する1ビットのリーダービットと，0から $s = 96N$ までの値をとるタイマによって構成される．すなわち，個体の状態空間は $\{l, f\} \times \{0, 1, \cdots, s\}$ であり，状態数は $192N = O(N)$ である．各個体は，リーダビットが l のときリーダであり，記号Lを出力する．そうでないとき，すなわちリーダビットが f のときは非リーダであり，記号Fを出力する．本プロトコルは緩安定プロトコルであるため，各個体の初期状態は任意である．したがって，実行開始時の初期状況において，リーダの数は0以上 n 以下の任意の値を取り得る．また，初期状況における各個体のタイマの値も0から s の範囲で任意である．本プロトコルの状態遷移規則は下記の通りである．

規則1: $((l,*),(*,*)) \to ((l,s),(f,s))$
規則2: $((f,*),(l,*)) \to ((f,s),(l,s))$
規則3: $((f,0),(f,0)) \to ((l,s),(f,s))$
規則4: $((f,i),(f,j)) \to ((f,m),(f,m))$

ただし，規則1,2において $*$ はドントケアを表す．また，規則4において i, j のいずれかが正の値（0ではない値）であることを規則4適用の条件とし，$m = \max(i, j) - 1$ であるとする．これらの4規則は排他的かつ網羅的であることに注意されたい．すなわち，2つの状態で構成されるいかなる順序対も，上記4規則のうちただひとつの規則の左辺に合致する．

本プロトコルの実行において，リーダどうしの交流が発生した場合，一方の個体がリーダであり続け，他方の個体が非リーダに変化する（規則1）．リーダと非リーダが交流した場合，両個体のリーダビットは変化しない（規則1,2）．リーダ個体が参加する交流が発生するとき，両個体のタイマの値は最大値 $s(= 96N)$ に初期化される（規則1,2）．この初期化を**タイマリセット**と呼ぶ．非リーダがリーダに変化するのはタイマの値が0である非リーダどうしで交流が発生したとき，かつそのときのみである（規則3）．この規則3が適用される交流が発生することを以降では**タイムアウト**と呼ぶ．タイムアウトの発生時，一方の個体が非リーダからリーダに変化するとともに，両個体のタイマの値は最大値 s に初期化される．いずれかのタイマの値が0でないような2つの非リーダが交流するとき，少なくともひとつの個体がタイマの値を1減じられる（規則4）．規則4には，大きなタイマの値を伝播させるという別の側面もある．すなわち，タイマの大きな個体と小さな個体が交流して規則4が適用された場合，後者

[1] O, Ω はランダウの記号である．$O(f(n))$ は $f(n)$ よりも漸近的に大きくない関数（の集合）を表し，$\Omega(g(n))$ は $g(n)$ よりも漸近的に小さくない関数（の集合）を表す．

のタイマは大きな値（前者のタイマの値から1減じた値）に設定される．この事象を**タイマの大値伝播**と呼ぶ．

少なくともひとつのリーダが存在する状況においては，タイマの大値伝播とタイマリセットによって全個体のタイマは比較的大きな値に保たれるため，タイムアウトは滅多に発生しない（Nに関して指数関数的に小さい確率でしか発生しない）．一方で，リーダがひとつも存在しない状況においては，タイマリセットが発生しないため，個体群中のタイマの最大値は単調非増加かつ（簡単な計算によって分かるとおり）$0\,(n \log n)$の期待ステップ数で1減少する．したがって，$s = 96N$であるから，$0\,(nN \log n)$期待ステップ数でタイムアウトが発生する．このため，リーダ個体が2つ以上存在する状況から実行を開始した場合は規則1によるリーダ個体の削減によって，リーダがひとつも存在しない状況から実行を開始した場合には規則3のタイムアウトによるリーダの創出によって，やがて唯一のリーダ個体が個体群中に誕生する．

期待収束時間および期待維持時間の厳密な解析は複雑なものとなるため割愛するが（文献[9]を参照されたい），個体群中にただひとつのリーダが存在し全個体のタイマの値が$s/2$以上である状況の集合を安全状況の集合Sとして定義すると，本プロトコルは$(0\,(nN \log n), \Omega\,(Ne^N)$-緩安定リーダ選挙プロトコルであることがいえる．すなわち，任意の初期状況からいずれかの安全状況に収束するまでの期待並列時間は$0\,(N \log n)$であり，任意の安全状況から開始する実行において唯一のリーダを維持する期待並列時間は$\Omega\,(e^N)$である．

5．最新の研究成果

2015年に泉[10]は前述の緩安定リーダ選挙を改良し，期待収束時間を$0\,(N \log n)$から$0\,(N)$に短縮することに成功した．同時に，$\Omega\,(e^N)$の期待維持時間を実現する任意の緩安定リーダ選挙プロトコルの期待収束時間は$\Omega\,(N)$であることを証明した．ゆえに，指数関数的な期待維持時間と線形時間より漸近的に小さい期待収束時間を両立することは不可能であることがいえる．

一方で，3節冒頭で紹介した，個体群中に唯一のリーダが存在することを仮定してPresburger算術で表現可能な任意の述語を対数多項式時間で計算するプロトコルと組み合わせて用いることを考えると，緩安定リーダ選挙プロトコルの期待収束時間を対数多項式時間まで削減することが極めて重要である．そこで筆者ら[11]は，2018年に，期待維持時間を十分に大きな多項式時間に短縮することと引き換えに対数多項式時間の期待収束時間を実現することに成功した．具体的には，プロトコルの設計者が自由に調整可能な定数パラメータcを用いて，期待収束時間が$0\,(c \log^3 n)$，期待維持時間が$\Omega\,(cn^{10c})$である緩安定リーダ選挙プロトコルを考案した．たとえばパラメータcに10を代入すると，期待維持時間は$\Omega\,(n^{100})$となり，指数時間ではないものの，実用上十分な維持時間を実現する．この新しいプロトコルは，複数のリーダが存在するときに高速にリーダを削減する仕組みを導入しており，また，その期待収束時間・期待維持時間の証明にはより精緻な確率的解析を必要とするが，カウントダウンタイマを用いてリーダの不在を検知し新たなリーダを生成する仕組みについては前述の緩安定プロトコルと本質的な差異はない．

6．おわりに

本稿では，分散システムを表現する計算モデルのひとつとして近年活発に研究されている個体群プロトコルモデルにおいて，代表的な問題であるリーダ選挙問題がどのように解決されるかについてやや詳細に紹介した．現時点で，特定の初期状況を仮定する通常の（非安定な）リーダ選挙プロトコルでもっとも高速なものの期待収束時間は$0\,(\log^2 n)$であり，特定の初期状況を仮定しない緩安定リーダ選挙プロトコルでもっとも高速なものの期待収束時間は$0\,(\log^3 n)$である．今後の課題としては，より高速なプロトコルの実現可否を探求することが挙げられる．

参考文献

1) D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. Distributed Computing, 18(4):235-253, 2006.
2) D. Angluin, J. Aspnes, D. Eisenstat. Stably computable predicates are semilinear.

Proceedings of the 25th annual ACM Symposium on Principles of distributed computing, 292-299, 2006.

3) M. J. Fischer and M. O. Rabin. Super-exponential complexity of Presburgerarithmetic. In Complexity of Computation, SIAM-AMS Proceedings, vol. VII, pp. 27-41. American Mathematical Society, 1974.

4) D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. Distributed Computing, 21(3):183-199, 2008.

5) D. Alistarh and R. Gelashvili. Polylogarithmic-time leader election in population protocols. In Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming, pages 479-491, 2015.

6) D. Doty and D. Soloveichik. Stable leader election in population protocols requires linear time. CoRR, abs/1502.04246, 2015.

7) P. Berenbrink, D. Kaaser, P. Kling, and L. Otterbach. Simple and efficient leader election. In OASIcs-OpenAccess Series in Informatics (Vol. 61). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

8) D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. ACM Transactions on Autonomous and Adaptive Systems, 3(4):13, 2008.

9) Y. Sudo, J. Nakamura, Y. Yamauchi, F. Ooshita, H. Kakugawa, and T. Masuzawa. Loosely-stabilizing leader election in a population protocol model. Theoretical Computer Science, 444:100-112, 2012.

10) T. Izumi. On space and time complexity of loosely-stabilizing leader election. In the proceedings of International Colloquium on Structural Information and Communication Complexity, 299-312, 2015.

11) Y. Sudo, F. Ooshita, H. Kakugawa, and T. Masuzawa, "Brief Announcement: Loosely-stabilizing Leader Election with Polylogarithmic Convergence Time", In Proceedings of the 32th International Symposium on Distributed Computing, DISC'18, 52:1-52:3, 2018.