数理解析研究所講究録
第833巻 1993年 153-163

双方向リングネットワーク上での自己安定2-相互排除
Self-Stabilizing 2-Mutual Exclusion on Bidirectional Rings

広島大学大学院工学研究科情報工学専攻　角川裕次　(Hirotsugu Kakugawa)
広島大学工学部第二類　山下雅史　(Masafumi Yamashita)

Abstract— Self-stabilizing algorithms are algorithms that work correctly even if states of processes are destroyed by errors; such algorithms can be considered as ideal fault tolerant algorithms. In this paper, we propose a uniform self-stabilizing 2-mutual exclusion algorithm, i.e., an algorithm such that exact two processes have privileges in legitimate configurations. The proposed algorithm works on bidirectional uniform ring networks under cental deamon. It is uniform in the sense that every process executes the same algorithm. We show that if n, the number of processes in a network, is prime, then the proposed algorithm correctly works. We also show that if n is composite and is greater than or equal to 6, then there exists no algorithm for solving the problem.

1 はじめに

複数のプロセスより構成される、分散システムを考えよう。各プロセスは状態を持ち、隣接するプロセス達から情報を集めながら状態を遷移してゆく。各プロセスの状態を組にしたものを、システムの状況と呼ぶことにする。ここで、「システムの初期状況に関わらず、システムを正常な状況に遷移させる」という問題を考える。この問題は、一過性のエラーによってプロセスの状態が変わってしまっても、再び(大域的な意味での)正常な状況にもってゆこうとする問題と考えることができる。このような性質を持つシステムは、「自己安定システム (self-stabilizing system)」と呼ばれ、耐故障性の観点から興味のあるシステムと考えられる。

この問題に対するトリビアルな解決策としては、サーバープロセスを1つ用意し、これが他のプロセスに対する指示を出すことによって再び正常な状況に遷移させるというものが考えられる。しかし、このためには、唯一の特別なプロセッサの存在や一意なプロセス識別子などを仮定する必要がある。そこで、同一のプロセスよりなるネットワークでの解が存在するか否かは興味ある問題である。

自己安定アルゴリズムは、近年活発な研究がなされてきている。自己安定の概念は1974年Dijkstraによって提案された[1]、特別なプロセスが一つだけ存在するリングネットワークでの自己安定アルゴリズムが提案された。[2]では、一様な(プロセス識別子がなく、全てのプロセスが同一のアルゴリズムを実行する)単方向リングネットワークにおいて、プロセス数$n \geq 2$が素数なら自己安定1-相互排除アルゴリズムが存在し、nが合成数ならば自己安定アルゴリズムが存在しないことが示されている。また、[3]では、[2]で提案されたアルゴリズムのための、各プロセスが安定検出を行なうアルゴリズムを提案している。

本稿では、n個(nは3以上の素数)の、同一なプロセスより構成される双方向リングネットワークを考える。各プロセスは状態を持ち、それ自身の状態と左右のプロセスの状態の上で定義された(幾つかの)述語をもつ。述語が真の時、そのプロセスは特権(privilege)を持つといわれ、真である述語に対応した動作が可能である。これまでは、正常な状態での特権の数が1である、自己安定1-相互排除アルゴリズムが提案されているが[2]、本稿では特権の数が2であるような自己安定2-相互排除アルゴリズムを提案し、その正当性について示す。また、nが6以上の合成数ならば、アルゴリズムが存在しないことも示す。

2 自己安定 k-相互排除問題

双方向リング上の連続した n 個のプロセスを $p_0, p_1, \ldots, p_{n-1}$、それぞれの状態を $q_0, q_1, \ldots, q_{n-1}$, $(q_i \in Q)$ とする。Q をプロセスの状態の有限集合とする。システムの状況 (configuration) γは、各プロセスの状態を組にしたもので、$\gamma = (q_0, q_1, \ldots, q_{n-1})$ である。可能な状況の集合を$\Gamma \subseteq Q^n$とし、正常な状況 (legitimate configuration) の集合を$\Lambda \subseteq \Gamma$とあらわす。各プロセスのアルゴリズムは "If $C(p,q,r)$ then $p\ \underline{q}\ r \rightarrow F(p,q,r)$" という (一般に複数個の) 規則で与える。ここで q,p,rは、それぞれ、リングでの左、それ自身、右のプロセスの状態を示す。ある状況$\gamma \in \Gamma$において、プロセス i が規則を適用して状況γ'になる時、$\gamma \xrightarrow{i} \gamma'$と書く。また、規則を適用するプロセス名が大切でない時は単に$\gamma \rightarrow \gamma'$と書く。更に、零回以上の規則の適用の連鎖の結果、状況がγからγ'に遷移する時、$\gamma \rightarrow^* \gamma'$と書く。

システムΣが *k self-stabilizing* であるとは、次の条件を満たすことである:

1. (No Deadlock) 任意の状況$\gamma \in \Gamma$に対し、少なくとも一つのプロセスが特権を持つ。
2. (Closure) $\forall \lambda, \lambda' (\lambda \in \Lambda, \lambda' \in \Gamma)[\lambda \rightarrow \lambda' \Rightarrow \lambda' \in \Lambda]$
3. (No Livelock) 任意の$\gamma \in \Gamma$に対しても、有限の遷移で正常な状況に達する。
4. (k-Mutual Exclusion) どの正常な状況においても、丁度 k個のプロセスが特権を持つ。
5. (Faireness) 正常な状況を含む無限の遷移列において、どのプロセッサも無限回特権を持つ。

一般に特権を持つプロセスは複数存在する。本稿では、C デーモン (central daemon) と呼ばれるモデルを取り扱う。C デーモンとは、特権を持つプロセスのうちどれか一つのみが選ばれ、対応した動作がアトミックに実行されるモデルである。なお、特権を持ったプロセスは有限時間内に実行される、という制約を付け加える。(この制約がないと、$k \geq 2$ に対して k-相互排除アルゴリズムが存在しないことは容易に示すことが出来る。)

3 自己安定 2-相互排除アルゴリズム

3.1 プロセス数が合成数の場合

補題 1 ある $a \geq 3$ に対し $n = 2a$ のとき、自己安定 2-相互排除アルゴリズムは存在しない。
(証明) 自己安定 2-相互排除アルゴリズムは存在すると仮定する。全てのプロセスの状態を任意のある状態 $q \in Q$ にした場合を考えると、仮定より全てのプロセスが特権を持っている。プロセスを、長さが 2 の a 個のブロックに分けて考える。各ブロック i $(1 \leq i \leq a)$ での左のプロセスを $p_{i,1}$、右のプロセスを $p_{i,2}$とする。$p_{1,1}, p_{2,1}, \ldots p_{a,1}$の順で実行する。$p_{i,1}, p_{i+1,1}$は一つのプロセッサを挟んで位置しているので、$p_{i+1,1}$の特権の有無は $p_{i,1}$の実行の影響を受けないことに注意せよ。a 個のプロセスの実行が終了した後は、a 個もしくは $2a = n$ 個のプロセスが特権を持つ。次に、もし $p_{1,2}$が特権を持っていれば、$p_{1,2}, p_{2,2}, \ldots p_{a,2}$の順で実行する。先程と同様に、$a$ 個のプロセスの実行が終了した後は、a 個もしくは $2a = n$ 個のプロセスが特権を持つ。以降、上記の順で実行を無限に続ける。このスケジューリングにおいては、$a \geq 3$ なので、有限時間内に特権の数は 2 にならない。これは矛盾。 □

補題 2 ある 2 つの奇数 $a, b \geq 3$ に対し $n = ab$ のとき、自己安定 2-相互排除アルゴリズムは存在しない。
(証明) 本補題は補題 1と同様な考え方で証明できる。長さ b のブロックが a 個存在することに注意せよ。 □

定理 1 プロセス数 $n \geq 6$ が合成数のとき、自己安定 2-相互排除アルゴリズムは存在しない。
(証明) 補題 1,2より、本定理が成立。 □

プロセス数 $n = 4$ に対して自己安定 2-相互排除アルゴリズムが存在するか否かは不明である。

3.2 プロセス数が 3 以下の場合

補題 3 プロセス数 $n = 2$ に対する自己安定 2-相互排除アルゴリズムが存在する。
(証明) 明らか。 □

$n = 3$ に対するアルゴリズムを以下に示す。アルゴリズムの記法であるが、"If C then $p\ \underline{q}\ r \rightarrow q'$" は、プロセスの左、それ自身、右の状態が各々$p, q, r$ であるとき、条件 Cが成立すれば、新しい状態を q'となることを表している。

The algorithm S_3
状態集合を $Q = \{0, 1\}$ とする。各プロセス Pは次のアルゴリズムを実行する。b を Pの状態、a (c) を Pの左プロセス (右プロセス) の状態である。なお、$\overline{0} = 1$, $\overline{1} = 0$ である。

Rule a If $b \neq a \wedge b = c$ then

$$a\ \underline{b}\ c \longrightarrow \overline{b}$$

Rule b If $b \neq a \wedge a = c$ then

$$a\ \underline{b}\ c \longrightarrow b$$

Rule c If $a = b = c$ then

$$a\ \underline{b}\ c \longrightarrow \overline{b}$$

補題 4 アルゴリズム S_3は自己安定 2-相互排除アルゴリズムである。
(証明) 容易に示すことが出来るので、証明は省略する。なお、正常な状況の集合は

$$\Lambda_3 = \{100,\ 110,\ 010,\ 011,\ 001,\ 101\}$$

である。 □

3.3 プロセス数 5 以上の素数に対するアルゴリズム

ここでは、$n \geq 5$ に対するアルゴリズムを与え、その正当性を証明する。各プロセスの状態は 3 つ組 $l.t.s$ で表す。$l \in \{0, 1, \ldots, n-3\}$ をラベル部、$t \in \{0\} \cup \{2, 3, \ldots, n-3\}$ をタグ部、$s \in \{0, 1\}$ を対称部と呼ぶ。$b.u.y$を Pの状態、$a.t.x$ $(c.v.z)$ を Pの左プロセス (右プロセス) の状態である。なお、タグ部及びラベル部に対する演算は、$n-2$ を法として行なう。

システムが次の状況にある時が、正常な状況である。

$$\ldots, a-1.0.0, a.0.0, \underline{a.0.*}, \underline{a.0.0}, a+1.0.0, \ldots \text{ or,}$$
$$\ldots, a-1.0.0, a.0.0, \underline{a.0.*}, a+1.0.0, \ldots, b-1.0.0, b.0.0, \underline{b.0.1}, b+1.0.0, \ldots$$

但し、$*$は 0,1 任意のものとする。なお、下線を引いた状態を持つプロセスが特権を持っている。$n = 7$ に対する正常な状況の例を示す。

0.0.0, $\underline{0.0.0}$, 1.0.0, 2.0.0, $\underline{2.0.0}$, 3.0.0, 4.0.0

0.0.0, $\underline{0.0.1}$, $\underline{0.0.0}$, 1.0.0, 2.0.0, 3.0.0, 4.0.0

3.4 アルゴリズム

The algorithm S_n for $n \geq 5$:

Rule A If $b \neq a+1 \wedge (b \neq 0 \vee t = 0 \vee t = n-3 \vee t \neq b-a \vee t < u)$
$\wedge \neg(a = c = 0 \wedge b = n-3 \wedge t = n-3)$
$\wedge \neg(a = b = c = 0 \wedge t = n-3 \wedge u = 0)$
$\wedge \neg(a = b = c \wedge x = 0)$ then

$$a.t.x\ \underline{b.u.y}\ c.v.z \longrightarrow (a+1).(b-a).0$$

Rule B If $b = a+1 \wedge t \neq u \wedge b \neq 0$ then

$$a.t.x\ \underline{b.u.y}\ c.v.z \longrightarrow b.t.y$$

Rule C If $a = c = 0 \wedge b = n-3 \wedge t = n-3$ then

$$a.t.x\ \underline{b.u.y}\ c.v.z \longrightarrow a.0.1$$

Rule D If $a = b = c \wedge x = 0$ then

$$a.t.x\ \underline{b.u.y}\ c.v.z \longrightarrow b.0.1$$

3.5 アルゴリズムの正当性

Γを可能な状況の集合、Λを正常な状況の集合とする。はじめに、本稿で用いる用語を定義する。

定義 1 連続したプロセスを $p_0, p_1, \ldots, p_{n-1}$とし、それぞれの状態のラベル部を $l_0, l_1, \ldots, l_{n-1}$とする。**セグメント** (*segument*) とは、連続したプロセスの列 $p_i, \ldots, p_j$で、$l_{k+1} = l_k + 1$ $(i \leq k < j)$ で、かつ $l_i \neq l_{i-1}+1$, $l_{j+1} \neq l_j + 1$ であるものをいう。s を、プロセス $p_i, \ldots, p_j$よりなるセグメントとするとき、p_iを s の**ヘッドプロセス** (*head process*)、p_jを s の**テイルプロセス** (*tail process*) と呼ぶ。s, s'を 2 つの連続したセグメントとし、p_i (p_{i+1}) を s のテイルプロセス (s'のヘッドプロセス) とするとき、s と s'の間に**ギャップ** (*gap*) があるといい、セグメント s の**ギャップサイズ** (*gap size*) とは、$l_{i+1} - l_i$である。ギャップサイズが gであるギャップを**gギャップ** (g *gap*) といい、特にギャップサイズが $n-3$ のとき**-1 ギャップ** (-1 *gap*) いう。セグメント s に属している全てのプロセスの状態のタグ部が s のギャップサイズであるとき、s は *well formed* である、という。

補題 5 (Closure) $\forall\lambda(\lambda \in \Lambda)[\lambda \rightarrow \lambda', \lambda' \in \Gamma \Rightarrow \lambda' \in \Lambda]$
(証明) 任意の$\lambda \in \Lambda$に対して、Rule B, C が適用できないことは明らか。以下の証明においては、対称部 (状態の第三要素) は省略して書く。

- 場合 1: 状況が$\ldots, a-1.0, a.0, \underline{a.0}, a+1.0, \ldots, b-1.0, b.0, b+1.0, \underline{b.0}, \ldots$ のとき。(即ち、特権プロセスが隣合わない時。) 適用可能なものは、Rule A のみである。適用前の状況を

$$\ldots, a-1.0, a.0, \underline{a.0}, a+1.0, \ldots.$$

とすれば、適用後の状況は、

$$\ldots, a-1.0, a.0, a+1.0, \underline{a+1.0}, \ldots$$

となる。実行を行なったプロセスの右のプロセスに特権が移る。この状況はΛに属している。

- 場合 2: 状況が$\ldots, a-1.0.0, a.0, \underline{a.0}, \underline{a.0}, a+1.0, \ldots$ のとき。(即ち、特権プロセスが隣合う時。) 左側 (右側) の特権プロセスを P_1 (P_2) とする。P_1は Rule D、P_2は Rule A による特権を持っている。P_1が実行されても、P_1の対称部が 1 になるだけで、他の状況は変わらない。P_2が実行されると、

$$\ldots, a-1.0, a.0, \underline{a.0}, a+1.0, \underline{a+1.0}, \ldots$$

となる。この状況はΛに属している。 □

補題 6 (2-Mutual Exclusion) $\forall\lambda(\lambda \in \Lambda)$[λにおいて特権プロセスは 2 つのみ存在する。]
(証明) 以下の証明においては、対称部 (状態の第三要素) は省略して書く。$\forall\lambda \in \Lambda$に対して、

- 場合 1: 状況が$\ldots, a-1.0, a.0, \underline{a.0}, a+1.0, \ldots, b-1.0, b.0, \underline{b.0}, \ldots$ のとき。(即ち、特権プロセスが隣合わない時。) このとき、ちょうど 2 つのプロセスが Rule A による特権を持っていることは明らか。

- 場合 2: 状況が$\ldots, a-1.0, a.0, \underline{a.0}, \underline{a.0}, a+1.0, \ldots$ のとき。(即ち、特権プロセスが隣合う時。) このとき、1 つのプロセスが Rule A による特権を、それと異なる 1 つのプロセスが Rule D による特権を持っていることは明らか。 □

補題 7 (Fairness) $\forall\lambda(\lambda \in \Lambda)$[λ以降、どのプロセスも無限回の特権を持つ。]
(証明) 任意の$\lambda \in \Lambda$において、2 つのプロセスが特権を持ち、少なくとも一つのプロセスは Rule A による特権を持っている。C-Deamon に対する条件より、Rule D による特権を持つプロセスのみが実行されることはなく、有限時間内に Rule A による特権を持つプロセスが実行される。Rule A が適用されると、特権は右のプロセスに移る。 □

補題 8 (No Deadlock) $\forall\gamma(\gamma \in \Gamma)$[γにおいて、少なくとも一つのプロセスは特権を持つ。]
(証明) ある$\gamma \in \Gamma$において、どのプロセスも特権を持たないと仮定する。Rule C,D の条件が全プロセスで不成立なので、Rule A の条件が不成立となるためには、

$$(b = a+1) \vee (b = 0 \wedge t \neq 0 \wedge t \neq n-3 \wedge t = b-a \wedge t \geq u)$$
$$\vee (a = b = c = 0 \wedge t = n-3 \wedge u = 0)$$

が全プロセスにおいて成立する。プロセス数は n, 状態のラベルは $0, 1, \ldots, n-3$ であることより、全てのプロセスで $b = a+1$ は成立し得ない。
同一のラベルを持つ連続したプロセスが 2 つ以上存在する場合を考える。

- 全てのプロセスのラベル部が等しい場合。各プロセスのラベルが等しいので、全てのプロセスにおいて節 $a = b = c = 0 \wedge t = n-3 \wedge u = 0$ のみが真である。Pを任意のプロセスとすると、Pのタグは 0 で、Pの左隣のプロセス P_Lのタグは $n-3$ である。P_Lにおいても $a = b = c = 0 \wedge t = n-3 \wedge u = 0$ が成立するので、P_Lのタグは 0 である。これは矛盾。

- その他、即ち、ラベル部が$i-1, i, \ldots, i, i+1, \ldots$である場合。$i$をラベルとする連続するプロセスで一番右のものを$P$とする。$P$において、$a=i$, $b=i$, $c=i+1$である。すなわち、$b \neq a+1$, $\neg(t \neq 0 \wedge t = b-a)$, $\neg(a=b=c=0)$である。ゆえに、Pでは上の条件が成立せず、矛盾。

故に、同一のラベルを持つ連続したプロセスは2つ以上連続して存在しない。よって、上の条件は、

$$(b = a+1) \vee (b = 0 \wedge t \neq 0 \wedge t \neq n-3 \wedge t = b-a \wedge t \geq u)$$

に等価である。

- 場合1: セグメント数が1の時。ギャップは1つしかないので、ギャップサイズは$b-a=n-3$である。よって、$t \neq n-3 \wedge t = b-a$は常に偽であり、矛盾。
- 場合2: セグメント数が2の時。
  - セグメント長が1と$n-1$のものよりなる時。この時のラベル列は、$0, 0, 1, \ldots n-3, 0$となるが、同一のラベルを持つプロセスが3つ連続しているので、この場合は生じない。
  - その他の時。以下の、場合3と同様に示せる。
- 場合3: セグメント数が3以上の時。このとき、どのセグメントも左端のプロセスのみがラベルとして0を持つ。Rule Bが適用できないので、各セグメント内でのプロセスのタグは等しい。$t \geq u$なので、全てのプロセスは同一のタグを持つ。故に、セグメントの長さはどれも等しい。nが素数であることより、各セグメントは長さが1である。すなわち、どのプロセスも等しいラベル0を持つ。これは矛盾。 □

以上の補題により、self-stabilizingであるため条件のうちの4つが示された。以下で、残りの条件、No Livelock、即ち、任意の初期状態および任意のスケジューリングに対しても、有限時間内に正常な状況に到達することを示す。

まず、セグメント数が2以下である状況であれば、有限時間内に正常な状況に到達することを示す。

補題 9 $\gamma \in \Gamma$において、セグメント数がLとする。このとき、ギャップサイズの総和は、$G_L = L-2 \,(\mathrm{mod} n-2)$である。

(証明) γにおいて、連続したL個のセグメントを$s_0, s_1, \ldots, s_{L-1}$とする。各$s_i$でのヘッド, テイルプロセスのラベルをそれぞれ$H_i, T_i$とする。$s_i$と$s_{i-1}$のギャップサイズは、$g_i = H_{i+1} - T_i$である。各セグメント$s_i$の長さを$l_i$とすれば、$T_i = H_i + l_i - 1$, $\sum_{i=0}^{L-1} l_i = n$である。以上より、次式を得る。

$$\begin{aligned}
G_L &= \sum_{i=0}^{L-1} g_i \,(\mathrm{mod} n-2) \\
&= \{\sum_{i=0}^{L-1} H_{i+1} - \sum_{i=0}^{L-1} T_i\} \,(\mathrm{mod} n-2) \\
&= \{\sum_{i=0}^{L-1} H_{i+1} - \sum_{i=0}^{L-1} (H_i + l_i - 1)\} \,(\mathrm{mod} n-2) \\
&= \{-\sum_{i=0}^{L-1} l_i + \sum_{i=0}^{L-1} 1\} \,(\mathrm{mod} n-2) \\
&= \{-n + L\} \,(\mathrm{mod} n-2) \\
&= \{L-2\} \,(\mathrm{mod} n-2)
\end{aligned}$$

□

補題 10 任意の$\gamma \in \Gamma$に対して、γ以降 Rule B のみが連続して無限回適用されない。
(証明) γ以降 Rule B のみが連続して無限回適用されるとする。γでの (連続した) セグメント数を$s_0, s_1, \ldots, s_{L-1}$とおく。仮定より、プロセスの状態のラベル部は変わらない。すなわち、各セグメントを構成しているプロセスは永久に変化しない。また、少なくとも一つのセグメントsが存在して、sに属しているプロセスが無限回 Rule B を適用する。以下、このsについて考える。

sでの時刻を、以下のように定める。γにおいては、時刻 0 とする。時刻$t \geq 0$以降、sのプロセスが Rule B を適用した時を時刻$t+1$と定める。(システム全体において Rule B のみしか適用されないことと、Rule B の適用の結果はセグメントの外には影響を与えないことに注意せよ。) sを構成するプロセスを、左から右に向かって、$P_1, P_2, \ldots, P_l$とし、時刻$t(\geq 0)$でのP_iの状態のタグ部をx_i^tとおく。各時刻tでのsに対して、ベクトル$V_t = (v_1^t, v_2^t, \ldots, v_l^t) \in \{0,1\}^l$を次のように定める。

$$v_i^t = \begin{cases} 0 & \text{if } x_{i-1} = x_i \ \vee \ i = 0, \\ 1 & \text{otherwise.} \end{cases}$$

すなわち、V_tはセグメント内の各プロセスが Rule B による特権を持っているか否かを表している。このベクトルの比較は、辞書式順で行なう。時刻tにおいてP_iが Rule B の適用を行なうと、$v_i^{t+1} = 0$となる。$v_i^t = 1$に注意すれば、全ての$t \geq 0$に対し、$V_t > V_{t+1}$である。ベクトル$(0,0,\ldots,0)$が最小の元なので、Rule B が無限回適用されることに矛盾する。 □

補題 11 状況γ以降、セグメント数が一定とする。有限時間内に、全てのセグメントが well formed となる。
(証明) 任意のセグメントを一つ固定し、sとする。仮定より、有限時間内でsのテイルプロセスのラベル部が 0 となることは明らか。このプロセスをPとおく。明らかに、Pは有限時間内にsのヘッドプロセスとなる。Rule B はラベル 0 を越えてタグを使えないことに注意すれば、sはこのとき well formed となっている。セグメント数は有限なので、有限時間内に全てのセグメントが well formed となる。 □

補題 12 γをセグメント数が 1 である任意の状況とする。任意のスケジューリングに対し、有限時間内のある時刻において Rule C が適用される。
(証明) γ以降 Rule C が適用されないと仮定する。Rule B はラベル 0 を越えてタグを伝えないので、永久に Rule B のみが適用されることはない。即ち、γ以降のどの時点においても、有限時間内に Rule A が適用される。Rule A が適用される度に、セグメントのテイルプロセスのラベルは 1 増える。よって、ある時刻が存在して、あるプロセスPが Rule A を実行してセグメントのテイルプロセスとなり、かつPのラベルが 0 となる。このとき、Pのタグ部は、$(n-4)-(n-3) = n-3$である。ラベルが 0 のときは、Rule B は実行できないので、Pのタグは次に Rule A を実行するまで変わらない。そして、更にある時刻が存在して、Pがそのセグメントのヘッドプロセスとなる。このとき、全プロセスのタグは$n-3$となる。その後、有限時間内に、Pのラベルが$n-3$となり、セグメントのヘッドプロセスとなる。(全プロセスのタグは$n-3$であることに注意せよ。) このとき、Pのみが特権を持っており、それは Rule C によるものである。これは矛盾。 □

補題 13 関数$f : \Gamma \mapsto \mathbf{N}$を、$f(\gamma) = L_\gamma + z_\gamma$なる写像とする。ここで、$L_\gamma \geq 1$ $(z_\gamma \geq 0)$をγにおけるセグメント数 (-1 ギャップの数) とする。また、$\mathbf{N}$は自然数の集合$\{0,1,2,\ldots\}$である。任意の$\gamma \in \Gamma$に対し、$\gamma \to \gamma'$なる全ての$\gamma' \in \Gamma$に対し、$2 \leq f(\gamma') \leq f(\gamma)$が成立。
(証明) 任意のγに対し、$L_\gamma \geq 1$、$z_\gamma \geq 0$であり、$L_\gamma = 1$のとき$z_\gamma = 1$なので、$f(\gamma) \geq 2$が成立。

セグメント数または-1 ギャップの数が変化するのは、Rule A でセグメントが消滅する場合もしくは Rule C でセグメント数が増加する場合である。

- Rule A により、セグメントが消滅する場合。セグメント数は 1 減少する。新たに得られたギャップサイズが-1 であれば、$f(\gamma) = f(\gamma')$ が、それ以外なら、$f(\gamma) > f(\gamma')$ が成立。
- Rule C でセグメントが増加する場合。Rule C は-1 ギャップを 1 つ減少させて、セグメントを 1 つ増加させるので、$f(\gamma) = f(\gamma')$ が成立。

以上より、本補題が成立。 □

補題 14 ラベル部が $a-1, a, a, \underbrace{a+1, \ldots, b-1, b}_{0\text{個以上}}, b, b+1, \ldots a-2$、すなわちセグメント数が 2 で、ギャップサイズが共に 0 で、タグ部、対称部は任意であるような$\gamma \in \Gamma$を考える。無限にセグメント数が 2 であるような任意のスケジューリングに対して、ある$\lambda \in \Lambda$が存在し、$\gamma \rightarrow^* \lambda$である。
(証明) 補題 11により、有限時間内に 2 つのセグメントは well formed となる。これは正常な状況である。 □

補題 15 $\gamma \in \Gamma$において、ラベル部がある a に対して $a, a+1, a, a+1, a+2, \ldots, a-1$ とし、タグ部、対称部は任意とする。このとき、任意のスケジューリングに対して、ある$\lambda \in \Lambda$が存在し、$\gamma \rightarrow^* \lambda$である。
(証明) 補題 12により、γ以降のある時点で Rule C が適用される。Rule C が適用された直後での、2 つのギャップサイズはともに 0 である。また、セグメント数が 2 でギャップサイズが共に 0 のときは Rule C は適用されないので、補題 13により、γ以降の全ての状況でのセグメント数は 2 以下である。Rule C をあるプロセス Pが適用した後、Pの左右のプロセスが Rule A を適用するまでは、Pは Rule A の条件 "$\neg(a = b = c = 0 \wedge t = n-3 \wedge u = 0)$" により、Rule A を適用できない。また、$P$の右または左のプロセスが Rule A を適用した後では、$a = b = c$の成立するプロセスの左のプロセスの状態の対称部は必ず 0 である。(Rule A の実行にともない、対称部は 0 となる。) このときは、Rule A の条件 "$\neg(a = b = c \wedge x = 0)$" により、セグメントの吸収は生じない。即ち、セグメント数は永久に 2 である。補題 14により、有限時間内に正常な状況に達する。 □

補題 16 $\gamma \in \Gamma$において、セグメント数が 2 で、かつ 2 つのギャップサイズがともに 0 でないとする。このとき、任意のスケジュールに対し、ある$\lambda \in \Lambda$が存在し、$\gamma \rightarrow^* \lambda$である。
(証明) 任意のγを固定する。γにおいて、次のような状況にあるとする。

$$H_0, H_0+1, \ldots, T_0, H_1, H_1+1, \ldots, T_1$$

ここで、H_i (T_i) は、セグメント s_iのヘッド (テイル) プロセスのラベルである。2 つのギャップサイズを $g_0 = H_1 - T_0 \neq 0$, $g_1 = H_0 - T_1 \neq 0$ とおく。補題 9より、$G_2 = g_0 + g_1 = 0$。$g_0 \geq g_1$とする。

- $g_0 = n-3$ のとき。$g_1 = 1$ となり、セグメント数が 2 であることに矛盾。
- $\lfloor (n-3)/2 \rfloor \leq g_0 \leq n-4$ のとき。γ以降セグメント数が変化しない限りは、Rule C を適用できないことは明らか。γ以降、セグメント数が 2 のままであると仮定すると、ある時刻において、2 つのセグメントは well formed となり、かつセグメント s_1のヘッドプロセス Pのラベル部が 0 となる。このとき、Pにおいて、次が成立する。$\neg b \neq 0$ since $b = 0$, $\neg t \neq 0$ since $t = g_0 \neq 0$, $\neg t = n-3$ since $t = g_0 \neq n-3$, $\neg t \neq b-a$ since $t = g_0 = b-a$, $\neg t < u$ since $t = g_0 \geq g_1 = u$。ゆえに、Pは Rule A を (Rule B も) 適用できず、セグメント s_0のヘッドプロセスのみが特権を持ち、Rule A を適用してゆく。そして有限時間内にセグメント s_0が吸収され、セグメント数が 1 となり、矛盾。したがって、ある時刻にセグメント数が 1 となる。このことと補題 15より、本補題が成立。 □

補題 17 セグメント数が2以下である任意の$\gamma \in \Gamma$と任意のスケジューリングに対し、システムは有限時間内に正常な状況に達する。
(証明) γにおいてセグメント数が1ならば、補題15により有限時間内に安定する。γ以降セグメント数が永久に2なら、補題14により有限時間内に安定する。γにおいて、セグメント数が2で、ある時点でセグメント数が1となれば、補題15が適用できる。 □

以下では、セグメント数が$L > 2$である任意の状況に対しても、有限時間内にセグメント数が$L-1$となり、ゆえに、有限時間内にセグメント数が2になることを示す。

補題 18 セグメント数nではlivelockは生じない。
(証明) n個のプロセスを$P_0, P_1, \ldots, P_{n-1}$とする。セグメント数が$n$のままでのlivelockが生じたと仮定すると、Rule B,Cの条件部は全プロセスにおいて不成立である。もしRule Aによる特権しか存在しないとすれば、Rule Aが適用され、セグメントが減少し、矛盾。あるプロセスがRule Aのみで、他のプロセスがRule Dのみで特権を持っているとすれば、有限時間内にRule Aが適用され、セグメントが減少し、矛盾。ゆえに、同一のプロセスでRule AとDの条件部が同時には成立し得ないことに注意すれば、Rule Aの条件部が成立するプロセスは存在せず、Rule Dによる特権のみしか存在しない。
Rule Aの条件部が成立しないことより、任意の時刻において全プロセスで次が成立。

$$(b = 0 \wedge t \neq 0 \wedge t \neq n-3 \wedge t = b-a \wedge t \geq u)$$
$$\vee (a = b = c = 0 \wedge t = n-3 \wedge u = 0) \vee (a = b = c \wedge x = 0)$$

ゆえに、全プロセスのラベル部は等しい。従って、この条件は、全てのプロセスのラベル部が等しいという仮定のもとで$(x = 0) \vee (t = n-3 \wedge u = 0)$と等価である。Rule Dを実行したプロセスの右隣のプロセスPでは、Rule Dの定義より、$x = 1 \wedge t = 0$となり、上の条件は成立しない。よって、PはRule Aを適用し、セグメント数が$n-1$となる。 □

補題 19 ある時刻τ以降Rule Cが適用されず、かつセグメント数が3以上と仮定する。すると、ある時刻$t > \tau$が存在し、時刻tでのセグメント数は減少する。
(証明) τ以降、セグメント数は一定の$L \geq 3$と仮定し、矛盾を導く。補題18より、$L \leq n-1$と仮定できる。補題9より、ギャップサイズの総和は$G_L = L-2$である。
L個のセグメントを$s_0, s_1, \ldots, s_{L-1}$とおく。セグメント$s_i$のヘッドプロセス(テイルプロセス)のラベルを$H_i$ (T_i)とおく。s_iのギャップサイズg_iは、$g_i = H_{i+1} - T_i$である。補題11により、ある時刻が存在して、全セグメントがwell formedとなる。
Rule Aの条件部を考えれば、各セグメントのヘッドプロセスでは任意の時点において次が成立する。

$$b \neq 0 \vee t = 0 \vee t = n-3 \vee t \neq b-a \vee t < u$$

セグメントがwell formedなので、$t = b-a$であること、各ヘッドプロセスは無限にしばしば0をラベルとして持つこと及び、各セグメント内のプロセスのタグ部は永久に一定であることに注意すれば、この条件は$t = 0 \vee t = n-3 \vee t < u$に等価である。

- 全てのヘッドプロセスで$t = 0$の時。$G_L = L-2$に反するので、これは生じない。
- その他の時。全てのヘッドプロセスにおいて$t < u$は生じないので、あるセグメントのヘッドプロセスにおいて$t = n-3$となる。以下では、$t = n-3$の成立するプロセスをヘッドプロセスとするセグメントsについて考える。

sのヘッドプロセスPがRule Aを適用する時、sは少なくとも2つのプロセスより成る。(sに一つしかプロセスがないと、Rule Aの適用にともなってsが消滅し、セグメント数が減少する。) よって、次のような状況がある時点で生じる:

$$\ldots, 0.n-3.x, n-3.u.y, 0.u.z, \ldots$$

ここで、$n-3.u.y$を状態として持っているのがPである。この状況では、PにおいてRule Cの条件部が成立し、Rule Aのそれは不成立となる。仮定により、Rule Cは適用されないので、有限時間内にsの左隣または右隣のセグメントが消滅する。これは矛盾。 □

補題 20 任意の$\gamma \in \Gamma$と任意のスケジューリングに対して、有限時間内に全てのプロセスはRule Aを適用する。

(証明) ある$\gamma \in \Gamma$と、γに対するあるスケジュールリングに対してあるプロセスPがRule Aを適用しないと仮定する。Rule Aは、実行にともなって特権を失い、(もし可能なら)右隣へ特権を移す働きをする。従って、仮定により、PがRule Aによる特権をブロックしてしまうので、Rule Aはγ以降、高々有限回しか適用されない。最後にRule Aが適用された直後の状況をγ_1とおく。(もしγ以降Rule Aが適用されない場合は、$\gamma_1 = \gamma$とする。) 明らかに、γ_1以降は有限回しかRule Cが適用可能ではない。最後にRule Cが適用された直後の状況をγ_2とおく。(もしγ_1以降Rule Cが適用されない場合は、$\gamma_2 = \gamma_1$とする。) γ_2以降は各プロセスの状態のラベル部は変わらないので、Rule Bの適用は有限回である。最後にRule Bが適用された直後の状況をγ_3とおく。(もしγ_2以降Rule Bが適用されない場合は、$\gamma_3 = \gamma_2$とする。)

γ_3におけるセグメントを$s_1, s_2, \ldots, s_L$とおく。($3 \leq L \leq n-1$と仮定できることに注意せよ。) 補題8により、少なくとも一つのプロセスはRule Dによる特権を持っている。Rule Dの適用を行なっても特権の移動は生じないので、もしRule D以外で特権を持ち、かつRule Dで特権を持っていないようなプロセスが存在すると、C-Daemonのスケジューリングに対する制約により、有限時間内にそれが実行される。故に、γ_3以降任意の時点で、どのプロセスにおいてもRule A,B,Cの条件部は不成立である。(Rule Dと他のRuleとが同時には成立し得ないことに注意せよ。) あるプロセスがRule Dを適用したとする。このプロセスの右隣プロセスをP_Rとすると、P_Rにおいて$b \neq a+1 \wedge t = 0 \wedge x = 1$が成立するので、Rule Aの条件部が成立し、かつRule Dの条件部が不成立。故に、有限時間内にP_RはRule Aを適用することとなり、矛盾。 □

補題 21 関数fを、補題13の中で定義されたものとする。任意の$\gamma \in \Gamma$と任意のスケジューリングに対し、$\gamma_t \rightarrow \gamma_{t+1}, t \geq 0, \gamma_0 = \gamma$とする。ある$\tau$が存在し、$f(\gamma_\tau) < f(\gamma_0)$となる。

(証明) もしそのようなτが存在しないと仮定すれば、livelockが生じていることになる。補題18,19より、セグメント数が一定のままでのlivelockは生じない。よって、セグメントの消滅と生成が繰り返されるようなlivelockが生じる場合を考える。このためには、補題13により、セグメントの消滅に伴って得られるギャップが必ず-1ギャップであり、かつ、Rule Cの適用が無限回繰り返される。Rule Cが無限回適用されると、γ以降の(有限時間内の)ある時刻において、セグメントの消滅の結果得られるギャップサイズが-1 ($= n-3$)でないものが存在することを示す。

補題17より、セグメント数は任意の時刻において3以上と仮定できる。どのプロセスもRule Aを少なくとも一度適用した時刻以降を考える。(補題20より、そのような時刻が存在することに注意せよ。) あるプロセスPがRule Cが適用されると、1つの-1ギャップが消滅し、2つの0ギャップが生じてセグメント数が1増える。この時の状況は、

$$\ldots, 0.n-3.x, 0.0.1, 0.v.z, \ldots$$

である。どのプロセスも少なくとも一度は Rule A を適用したので、$x = 0$ であることに注意せよ。s_{-1}, s_0, s_1を、それぞれ、Pの左のセグメント、Pのみからなるセグメント、Pの右のセグメントとする。Rule A の定義より、s_{-1}または s_1が消滅しない限りは、s_0は消滅しない。ところが、s_{-1}も s_1も消滅し得ないことを示す。

- s_{-1}が消滅すると仮定する時。s_{-1}の左のセグメントを s_{-2}とし、s_{-2}と s_{-1}とのギャップサイズを g_1とおく。s_{-1}が消滅して得られるギャップサイズは、$g = g_1 + 0 - 1 = g_1 - 1$ である。$g = -1 = n - 3$ となるためには、$g_1 = 0$。しかし、s_{-1}は 2 つの 0 ギャップではさまれたセグメントであるので、s_{-1} は、Rule A の定義により、消滅し得ない。

- s_1が消滅する時。s_{-1}が消滅する時と同様に、2 つの 0 ギャップではさまれたセグメントが消滅することになり、Rule A の定義に矛盾。

以上のことが Rule C を適用して得られたセグメント全てに対して成立する。よって、セグメントの吸収の結果得られたギャップサイズが-1 でないものが存在する。この時刻をτとすれば、$f(\gamma_\tau) < f(\gamma_0)$ である。 □

定理 2 プロセス数が $n \geq 2$ なる各素数に対し、自己安定 2-相互排除アルゴリズムが存在する。
(証明) $n \geq 5$ なる素数の場合を考える。Closure, 2-Mutual Exclusion, Fairness, No Deadlock はすでに示された。任意の$\gamma \in \Gamma$に対して、$f(\gamma)$ は有限なので、これまでの一連の補題より、任意の状態から有限時間内でセグメント数が 2 以下となる。更に、セグメント数が 2 以下である任意の状況から有限時間内で正常な状況に達することが示された。故に、No Livelock が成立する。よって、S_nは自己安定 2-相互排除アルゴリズムである。このことと、補題 3,4により、本定理が成立する。 □

4 結論

本稿では、プロセス数が 2 以上の素数であれば自己安定 2-相互排除アルゴリズムが存在し、プロセス数が 6 以上の合成数であれば、アルゴリズムが存在しないことを示した。しかし、プロセス数が $4 = 2 \cdot 2$ の時にアルゴリズムが存在するか否かは未解決である。本稿で用いた技法を拡張することでより一般的な自己安定 k-相互排除アルゴリズムが構成できるものと思われ、これは今後の課題である。

参考文献

[1] E.W.Dijkstra. Self-Stabilizing Systems in Spite of Distributed Control. *Communications of the ACM*, **17** (11), Nov. 1974, 643–644.

[2] J.E.Burns and J.Pachl. UniformSelf-Stabilizing Rings. *ACM Trans. on Programming Languages and Systems*, **11**(2), Apr. 1989, 330–344.

[3] C.Lin and J.Simon. Observing Self-Stabilization. *Proc. PODC*, 113–123 (1992).

[4] J.L.W.Kessels. An Exercise in Proving Self-Stabilization with a Variant Function. *Inform. Proc. Lett.*, 39–42 (1988).