Vol.2012-AL-142 No.3
2012/11/2

ペトリネットの状態数の多項式時間計算について
～有界性と入れ子構造を活用したアプローチ～

洲崎 武史[1,a)]　山口 真悟[1,b)]

概要：ペトリネットの振舞いに関する問題は，その状態空間を調べることにより解くことができる．状態空間そのものを求めることは，そのサイズがペトリネットのサイズの指数関数オーダーになる場合があるので，手に負えない．しかしながら，状態空間のサイズ（状態数）であれば，多項式時間で求められる可能性がある．状態数を求めることができれば，状態空間を求めるために必要な時間やメモリを見積もることが可能となる．本稿ではペトリネットの状態数計算問題を定式化し，その複雑さを明らかにする．さらにサブクラスに対して多項式時間の計算法を提案する．

1. はじめに

ペトリネットは離散事象システムをモデル化し解析するためのグラフィカルで数学的なツールである [1]．ペトリネットの強みはトークンに由来する．トークンを使用することにより，ペトリネットはシステムの構造だけでなく振舞いも表現することができる．システムの振舞いは，それをモデル化したペトリネットの状態を調べることにより解析できるが，ペトリネットがとりうる状態の数は，ペトリネットのサイズの指数関数オーダーになる場合がある．この問題は状態空間爆発と呼ばれており，状態空間そのものを求めることは手に負えない．一方，状態空間のサイズ（状態数）であれば，状態を列挙することなしに，効率よく求めることができる可能性がある．状態数を求めることができれば，状態空間を求めるために必要な時間やメモリを見積もることが可能となる．しかしながら，そのような可能性に対する検討は，これまで積極的に行われてこなかった．

Chao ら [2] はペトリネットの基本的なサブクラスに対して，状態数を計算する方法を提案している．対象のサブクラスは状態機械とマークグラフである．ただしマークグラフはブリッジ*1 がないものに限定されている．Chao らの方法は，まずペトリネットの構造を代数式として表し，その代数式から状態数を計算する．しかしながら，ペトリネットの構造を代数式として表す体系的な方法は与えられておらず，提案方法の計算量は明らかにされていない．

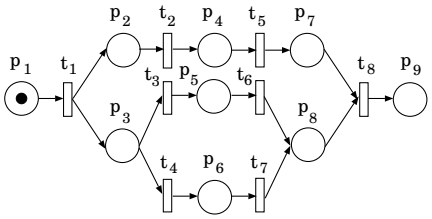


図 1　ペトリネット $(N_1, [p_1])$.

本稿ではペトリネットの状態数計算問題を定式化し，その複雑さを明らかにする．さらにサブクラスに対して多項式時間の解法を提案する．まず 2 章では，準備としてペトリネットの定義と性質を紹介する．3 章では状態数計算問題を定式化し，その可解性や計算複雑さについて，ペトリネットの性質の一つである有界性を用いて明らかにする．そして 4 章では，ペトリネットのサブクラスの一つであるアサイクリック Well-Structured ワークフローネット [3] に対して多項式時間の解法を提案する．この解法は Well-Structured ワークフローネットが入れ子構造を有するという特徴を活用する．最後に 5 章で，まとめと今後の課題を示す．

2. ペトリネット

（正規）ペトリネットは 3 項組 $N=(P,T,A)$ であり，P はプレースの有限集合，T（$\cap P \neq \emptyset$）はトランジションの有限集合，A（$\subseteq (P\times T)\cup(T\times P)$）はアークの有限集合である．ペトリネットの一例を図 1 に示す．プレース p の入力トランジションの集合 $\bullet p$ と出力トランジションの集

[1] 山口大学大学院理工学研究科
〒 755–8611 宇部市常盤台 2 丁目 16–1
[a)] s024vk@yamaguchi-u.ac.jp
[b)] shingo@yamaguchi-u.ac.jp

*1 ブリッジとは，ペトリネットのサイクル c，c とは端点以外では交わらないパス h に関して，c と h 上のノードをそれぞれ端点とし，それら以外では c や h と交わらないパスである．

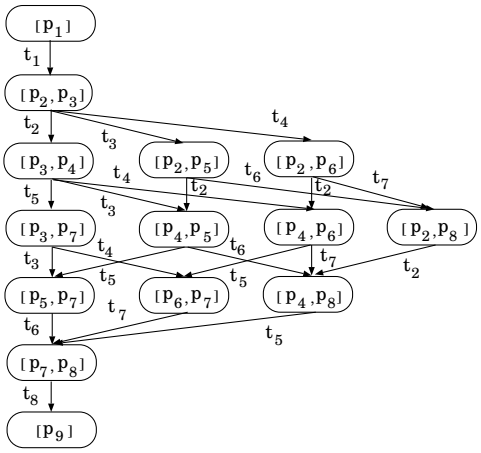


図 2 $(N_1, [p_1])$ の可達グラフ

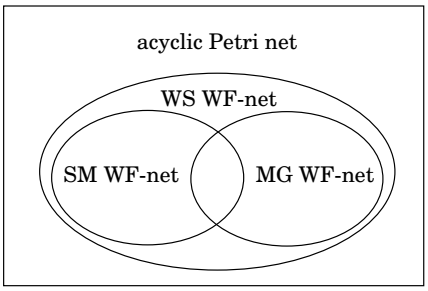


図 3 アサイクリックペトリネットのサブクラス間の関係

合 $p\bullet$ をそれぞれ次のように定義する：$\bullet p=\{t|(t,p)\in A\}$；$p\bullet=\{t|(p,t)\in A\}$．トランジション t の入力プレースの集合 $\bullet t$ と出力プレースの集合 $t\bullet$ も同様に定義される．マーキングはプレースから非負整数への写像 $M : P\to\{0,1,2,\cdots\}$ であり，多重集合の形式で $[p^{M(p)}|p\in P, M(p) > 0]$ と表記する．マーキングの初期値を初期マーキングと呼び，M_0 と表記する．マーキング M のペトリネット N を (N, M) と表記する．また M_0 から可達な全てのマーキングの集合を $R(N, M_0)$ と表記する．

M_0 から発火可能なトランジションを 1 回発火することにより，発火可能なトランジションと同数の新しい可達マーキングを得ることができる．それぞれの新しいマーキングから，またさらに新しい可達マーキングを得ることができる．このプロセスは，マーキングのグラフ表現に帰着する．そのようにして得られたグラフが，M_0 から可達な全てのマーキングを含むならば，可達グラフと呼ばれる．可達グラフのノードは M_0 とそれから可達なマーキングを表し，各アークはトランジションの発火を表す．例として図 1 のペトリネット $(N_1, [p_1])$ の可達グラフを図 2 に示す．

ペトリネットの初期マーキング M_0 から可達な任意のマーキングにおいて，各プレース内のトークンの数がある有限な数 k を超えなければ，k-有界あるいは単に有界であるという．

ペトリネットにはグラフ的な構造に制約を加えたサブクラスがある．

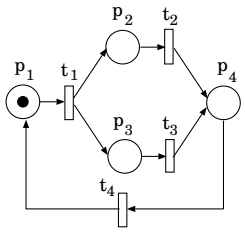


図 4 ペトリネット $(N_2, [p_1])$.

(1) 状態機械（State Machine; SM）は全てのトランジション t が高々 1 つの入力プレースと高々 1 つの出力プレースをもつペトリネットである．

(2) マークグラフ（Marked Graph; MG）は全てのプレース p が高々 1 つの入力トランジションと高々 1 つの出力トランジションをもつペトリネットである．

(3) ワークフローネット（WF ネット）は次の二つの条件を満たすペトリネットである：(i) 唯一のソースプレース p_I と唯一のシンクプレース p_O を持つ；(ii) 全てのノードが p_I から p_O へのパス上にある．

(4) Well-Structured WF ネット（WS WF ネット）は p_O から p_I へ追加のトランジション t^* を通じて接続したペトリネットに PT-ハンドルや TP-ハンドル[*2]が存在しない WF ネットである．

アサイクリックペトリネットのサブクラス間の関係をベン図の形式で図 3 に示す．アサイクリック WS WF ネットはアサイクリック SM WF ネットとアサイクリック MG WF ネットのスーパークラスであることに注意されたい．

3. 状態数計算問題とその複雑さ

状態数計算問題を以下のように定式化する．

定義 1 （状態数計算問題）
入力： ペトリネット (N, M_0)
出力： 状態数 $|R(N, M_0)|$ □

例として，図 1 のペトリネット N_1 について考えよう．初期マーキングは $[p_1]$ である．このペトリネット $(N_1, [p_1])$ の状態数計算問題は以下のように記述される．

[$(N_1, [p_1])$ の状態数計算問題]
入力： ペトリネット $(N_1, [p_1])$
出力： 状態数 $|R(N_1, [p_1])|$

$(N_1, [p_1])$ の可達グラフは図 2 のようになる．ノードの数を数えることによって，$|R(N_1, [p_1])| = 14$ と求めることができる．

もう一つの例として，図 4 のペトリネット N_2 について考えよう．初期マーキングは $[p_1]$ である．このペトリネット $(N_2, [p_1])$ の状態数計算問題は以下のように記述される．

[$(N_2, [p_1])$ の状態数計算問題]

[*2] TP-ハンドルとは，ペトリネットのサイクル c に関して，c 上のトランジションとプレースをそれぞれ始点と終点とし，それら以外では c と交わらないパスである．PT-ハンドルも同様に定義される．

入力： ペトリネット $(N_2, [p_1])$
出力： 状態数 $|R(N_2, [p_1])|$

$(N_2, [p_1])$ の可達グラフは無限に大きくなる．これは，たとえばトランジション $t_1t_3t_4$ が順に繰り返し発火すると，プレース p_2 にトークンが無限に増えるからである．

このようにペトリネットが有界でなければ，その状態空間は無限に大きくなってしまう．このため非有界なペトリネットの状態数は一見すると，計算できないように思われるが，記号 ∞ を使うと，その状態数は ∞ 個とみなして計算することができる．ここで ∞ は任意の整数 n に対し，$\infty \pm n = \infty$，$\infty \geq \infty$ という性質を満たす．したがって，$|R(N_2, [p_1])| = \infty$ と求めることができる．

状態数計算問題の可解性について考えよう．

性質 1 ペトリネット (N, M_0) の状態数計算問題は可解（solvable）である． □

証明：ペトリネットの有界性問題は決定可能である [4], [5]．(N, M_0) が有界である場合，状態数は有限なので，可達グラフをつくることができる．そのノードの数を数えれば，状態数を求めることができる．ペトリネットが有界でない場合，その状態数は上述のように ∞ とみなす．以上から状態数計算問題は可解である． **証明終わり**

状態数計算問題の複雑さについて考えよう．

性質 2 ペトリネット (N, M_0) の状態数計算問題は手に負えない（intractable）． □

証明：状態数計算問題を以下のような決定問題に変換する：

[状態数有限性問題]
入力： ペトリネット (N, M_0)
出力： 状態数 $|R(N, M_0)|$ は有限か?

この決定問題は有界性問題そのものである．有界性問題は手に負えないことが知られている．有界性問題は状態数有限性問題に多項式時間帰着できるので，状態数有限性問題は手に負えないといえる．オリジナルの状態数計算問題は，それ以上に難しいので，手に負えないといえる．

証明終わり

一般的なペトリネットに対して状態数計算問題は手に負えないが，サブクラスに限定すれば多項式時間で計算できる可能性がある．

非有界であることが多項式時間で検証できるクラスは，多項式時間でその状態数を ∞ として求めることができる．そのようなクラスの一つに MG でない無競合（conflict-free; CF）ネットがある．CF ネットは各プレースが高々 1 つの出力トランジションをもつペトリネットである．図 4 のネット N_2 は，$|\bullet p_4| > 1$ なので MG でなく，$|p_1\bullet| = |p_2\bullet| = |p_3\bullet| = |p_4\bullet| = 1$ なので CF ネットである．したがって，$(N_2, [p_1])$ は非有界であり，前述のように $|R(N_2, [p_1])| = \infty$ である．

一方，有界であることが多項式時間で検証できるクラスも，多項式時間でその状態数を求めることができる可能性がある．次節では，そのようなクラスに対する状態数計算問題を考察する．

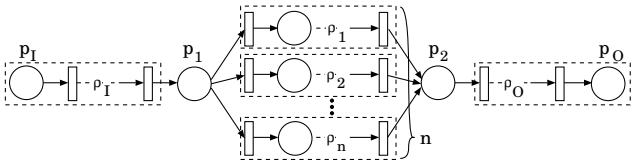


図 5 基本的な RASM WF ネット．

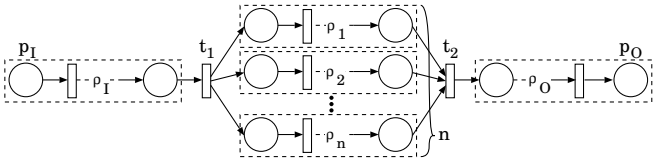


図 6 基本的な RAMG WF ネット．

4. 状態数計算問題の多項式時間解法

有界であることが多項式時間で検証できるクラスの一つにアサイクリック WS WF ネットがある．アサイクリック WS WF ネットは常に有界であることが知られている．またアサイクリック WS WF ネットはアサイクリック SM WF ネットやアサイクリック MG WF ネットを真に含む比較的大きなクラスである．さらにアサイクリック WS WF ネットは入れ子構造を有する．具体的には，アサイクリック SM WF ネットやアサイクリック MG WF ネットはアサイクリック WS WF ネットである．さらに，アサイクリック WS WF ネットの任意のプレースをアサイクリック SM WF ネットやアサイクリック MG WF ネットに置き換えたネットもアサイクリック WS WF ネットである．そこで本稿ではアサイクリック WS WF ネットの入れ子構造に注目し，それを活用した解法を提案する．

まず本解法で取り扱うネットを以下のように定義する．

定義 2 （制約付きアサイクリック WS WF ネット（RAWS WF ネット））

- 図 5 のネット（基本的な制約付きアサイクリック SM WF ネット（RASM WF ネット）と呼ぶ）は RAWS WF ネットである．
- 図 6 のネット（基本的な制約付きアサイクリック MG WF ネット（RAMG WF ネット）と呼ぶ）は RAWS WF ネットである．
- N を RASM WF ネットまたは RAMG WF ネット，N' を RAWS WF ネット，p を N のプレースとすると，$N \otimes_p N'$ は RAWS WF ネットである．ここで $N \otimes_p N'$ は N 中の p を N' で置き換える（リファインメント）操作である． □

RAWS WF ネットの制約とは，リファインメント操作をプレースに限定していることと，ブリッジがないことである．

RAWS WF ネットの一例を図 7 に示す．このネットは

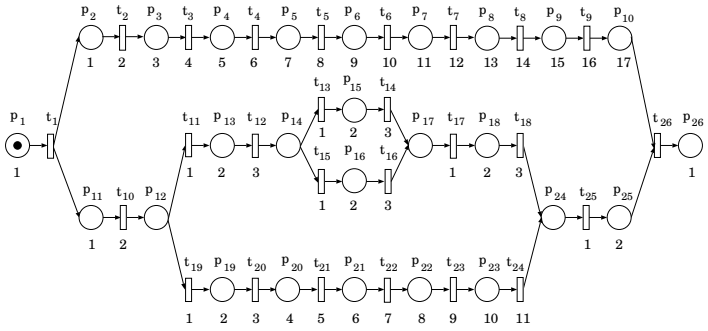


図 7 RAWS ワークフローネット N_3

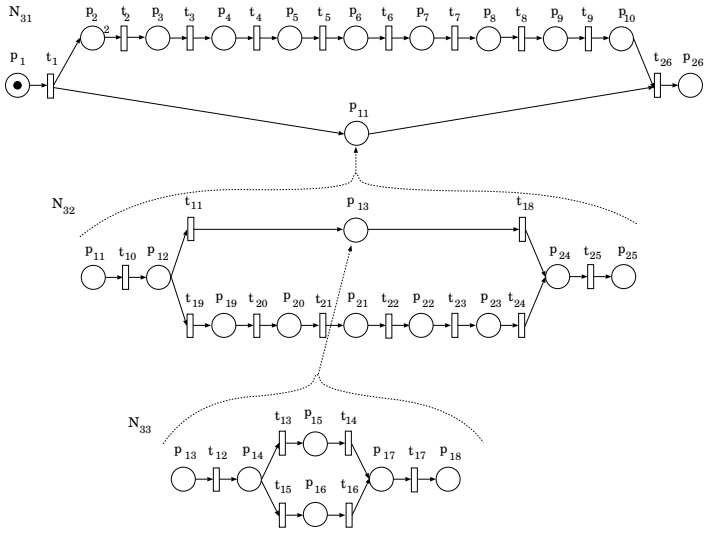


図 8 N_3 の入れ子構造：$N_{31}\otimes_{p_{11}}(N_{32}\otimes_{p_{13}}N_{33})$

次のような入れ子構造を成している（図 8 参照）：

$$N_3 = N_{31}\otimes_{p_{11}}(N_{32}\otimes_{p_{13}}N_{33})$$

この構造によって，状態数の計算を効率良く行うことが期待される．まず RASM WF ネット，RAMG WF ネットの計算法を示し，それから RAWS WF ネットの計算法を示す．

4.1 RASM WF ネットの状態数計算法

定義 3 RASM WF ネット

- 図 5 のネット（基本的な RASM WF ネット）は RASM WF ネットである．
- N を基本的な RASM WF ネット，N' を RASM WF ネット，p を N のプレースとすると，$N\otimes_p N'$ は RASM WF ネットである． □

RASM WF ネットの状態数は以下の性質を満たす．

性質 3 （RASM WF ネットの状態数）

- 基本的な RASM WF ネット（図 5 参照）$(N,[p_I])$ の状態数は

$$|R(N,[p_I])| = \lfloor|\rho_I|/2\rfloor + (\Sigma_{i=1}^{n}\lfloor|\rho_i|/2\rfloor + 2) + \lfloor|\rho_O|/2\rfloor \quad (1)$$

である．

- N を基本的な RASM WF ネット，N' を RASM WF ネット，p を N のプレースとすると，$(N\otimes_p N',[p_I])$ の状態数は，$(N',[p_I])$ の状態数を S' とすると

$$|R(N\otimes_p N',[p_I])| = \text{式 } (1) + S' - 1 \quad (2)$$

である． □

4.2 RAMG WF ネットの状態数計算法

定義 4 RAMG WF ネット

- 図 6 のネット（基本的な RAMG WF ネット）は RAMG WF ネットである．
- N を基本的な RAMG WF ネット，N' を RAMG WF ネット，p を N のプレースとすると，$N\otimes_p N'$ は RAMG WF ネットである． □

RAMG WF ネットの状態数は以下の性質を満たす．

性質 4 （RAMG WF ネットの状態数）

- 基本的な RAMG WF ネット（図 6 参照）$(N,[p_I])$ の状態数は

$$|R(N,[p_I])| = \lceil|\rho_I|/2\rceil + \Pi_{i=1}^{n}\lceil|\rho_i|/2\rceil + \lceil|\rho_O|/2\rceil \quad (3)$$

である．

- N を基本的な RAMG WF ネット，N' を RAMG WF ネット，p を N のプレースとすると，$(N\otimes_p N',[p_I])$ の状態数は，$(N',[p_I])$ の状態数を S' とすると
  - $p\in\rho_I,\rho_O$ の場合

$$|R(N\otimes_p N',[p_I])| = \text{式 } (3) + S' - 1 \quad (4)$$

  - $p\in\rho_j\in\{\rho_1,\rho_2,\cdots,\rho_n\}$ の場合

$$\lceil|\rho_I|/2\rceil+(\lceil|\rho_j|/2\rceil+(S'-1))\Pi_{i=1,i\neq j}^{n}\lceil|\rho_i|/2\rceil+\lceil|\rho_O|/2\rceil \quad (5)$$

である． □

4.3 RAWS WF ネットの状態数計算法

RAWS WF ネットの状態数は以下の性質を満たす．

性質 5 （RAWS WF ネットの状態数）

- 基本的な RASM WF ネット（図 5 参照）$(N,[p_I])$ の状態数は式 (1) である．
- 基本的な RAMG WF ネット（図 6 参照）$(N,[p_I])$ の状態数は式 (3) である．
- N を基本的な RASM WF ネット，N' を RAWS WF ネット，p を N のプレースとすると，$(N\otimes_p N',[p_I])$ の状態数は，$(N',[p_I])$ の状態数を S' とすると式 (2) である．
- N を基本的な RAMG WF ネット，N' を RAWS WF ネット，p を N のプレースとすると，$(N\otimes_p N',[p_I])$ の状態数は，$(N',[p_I])$ の状態数を S' とすると，
  - $p\in\rho_I,\rho_O$ の場合，式 (4) である．
  - $p\in\rho_j\in\{\rho_1,\rho_2,\cdots,\rho_n\}$ の場合，式 (5) である． □

4.4 RAWS WF ネットの状態数計算アルゴリズム

性質 5 に基づいて，RAWS WF ネットの状態数を計算するアルゴリズムを構築する．このアルゴリズムは深さ優先探索をベースにしている．ただし，未発見の入力ノードがある場合，バックトラックする．このアルゴリズムは二つのサブアルゴリズムを使う．直観的に，CalPathLen はネットをパスに分割し，それぞれの長さを求める．CalStateNum は，それらを性質 5 に当てはめていくことで状態数を計算する．

≪RAWS WF ネットの状態数計算 ≫

入力：RAWS WF ネット $(N, [p_I])$.
出力：状態数 $|R(N, [p_I])|$.

CalStateRAWS(N, p_I)
1 CalPathLen$(p_I, 1)$
2 Push$(S, 0)$
3 CalStateNum(p_I, S)
4 Output Pop(S) and stop

▷ ネットを構成する各パスの長さを求める
CalPathLen(n, l)
1 **if**$(|n\bullet|\geq 2)$
▷ n が複数の出力ノードを持つ場合
2 **for each** $m \in n\bullet$
3 CalPathLen$(m, 1)$
4 **else**
5 **if**$(|\bullet n|\geq 2)$
▷ n が複数の入力ノードを持つ場合
6 **if**(n の全ての入力ノードが発見済みか)
7 m を n の出力ノードとする
8 CalPathLen$(m, 1)$
9 **else**
▷ n が高々 1 つの入出力ノードを持つ場合
10 $len[n] \leftarrow l$
11 **if**$(n \neq p_O)$
12 m を n の出力ノードとする
13 CalPathLen$(m, l+1)$

▷ ネットの長さを性質 5 に当てはめて状態数を計算する
CalStateNum(n, S)
1 **if**$(|n\bullet|\geq 2)$
▷ n が複数の出力ノードを持つ場合
2 m を n の入力ノードとする
3 **if**$(n \in P)$ ▷ 式（1）の右辺第 1 項
4 Push$(S, \lfloor len[m]/2 \rfloor$ + Pop$(S))$
5 **if**$(n \in T)$ ▷ 式（3）の右辺第 1 項
6 Push$(S, \lceil len[m]/2 \rceil$ + Pop$(S))$
7 **for each** $m \in n\bullet$
8 Push$(S, 0)$
9 CalPathLen(m, S)
10 **else**
11 **if**$(|\bullet n|\geq 2)$
▷ n が複数の入力ノードを持つ場合
12 m を n の入力ノードとする
13 **if**$(n \in P)$
▷ 式（1）の右辺第 2 項
14 Push$(S, \lfloor len[m]/2 \rfloor$ + Pop$(S))$
15 **if**(n の全ての入力ノードが発見済みか)
16 Push$(S, \Sigma_{i=1}^{|\bullet n|}Pop(S) + 2 +Pop(S))$
17 m を n の出力ノードとする
18 CalPathLen(m, S)
19 **if**$(n \in T)$
▷ 式（3）の右辺第 2 項
20 Push$(S, \lceil len[m]/2 \rceil$ + Pop$(S))$
21 **if**(n の全ての入力ノードが発見済みか)
22 Push$(S, \Pi_{i=1}^{|\bullet n|}Pop(S)+Pop(S))$
23 m を n の出力ノードとする
24 CalPathLen(m, S)
25 **else**
26 **if**$(n = p_O)$
▷ 式（1）または式（3）の右辺第 3 項
27 Push$(S, \lceil len[n]/2 \rceil$ + Pop$(S))$

このアルゴリズムは深さ優先探索をベースにしており，多項式時間で動作する．

4.5 計算例

提案したアルゴリズムを次の問題に適用し，その動作を説明する．

[$(N_3, [p_1])$ の状態数計算問題]
入力： RAWS WF ネット $(N_3, [p_1])$（図 7 参照）
出力： 状態数 $|R(N_3, [p_1])|$

CalPathLen は p_I から各ノード n を順に深さ優先探索していき，分岐ノードや合流ノードの間にあるノードの数を $len[n]$ としてラベリングする．その結果を図 7 に示す．$len[n]$ の値は各ノードの下に図示される．分岐ノードや合流ノードの直前のノードの値がパスの長さになる．

CalStateNum は p_I から各ノード n を順に深さ優先探索していき，各パスの長さを性質 5 に当てはめていくことで状態数を計算していく．スタック S は初期値 0 をもつ．

トランジション分岐 t_1 が発見されたとき，その入力プレース p_1 の長さ $len[p_1] = 1$ を式 (3) の第 1 項に当てはめ，その値 $\lceil len[p_1]/2 \rceil + Pop(\mathrm{S}) = 1$ を S にプッシュする（図 9（a）参照）．

トランジション合流 t_{26} が最初に発見されたとき，その

入力プレース p_{10} の長さ $len[p_{10}] = 17$ を式 (3) の第 2 項に当てはめ，その値 $\lceil len[p_{10}]/2 \rceil + \textsc{Pop}(S) = 9$ を S にプッシュする（図 9（b）参照）．入力プレース p_{25} はまだ発見されていないので，バックトラックする．

プレース分岐 p_{12} が発見されたとき，その入力トランジション t_{10} の長さ $len[t_{10}] = 2$ を式 (1) の第 1 項に当てはめ，その値 $\lfloor len[t_{10}]/2 \rfloor + \textsc{Pop}(S) = 1$ を S にプッシュする（図 9（c）参照）．

プレース合流 p_{17} が最後に発見されたとき，その入力トランジション t_{16} の長さ $len[t_{16}] = 3$ を式 (1) の第 2 項に当てはめ，その値 $\lfloor len[t_{16}]/2 \rfloor + \textsc{Pop}(S) = 1$ を S にプッシュする．全ての入力トランジションが発見済みなので，CalStateNum の 16 行目以降の処理として，$\Sigma_{i=1}^{|\bullet p_{17}|} \textsc{Pop}(S) + 2 + \textsc{Pop}(S) = 5$ を S にプッシュする（図 9（d）参照）

トランジション合流 t_{26} が最後に発見されたとき，その入力プレース p_{25} の長さ $len[p_{25}] = 2$ を式 (3) の第 2 項に当てはめ，その値 $\lceil len[p_{25}]/2 \rceil + \textsc{Pop}(S) = 1$ を S にプッシュする．全ての入力プレースが発見済みなので，CalStateNum の 22 行目以降の処理として，$\Pi_{i=1}^{|\bullet t_{26}|} \textsc{Pop}(S) + \textsc{Pop}(S) = 136$ を S にプッシュする（図 9（e）参照）．

シンクプレース p_{26} が発見されたとき，$len[p_{26}] = 1$ を式 (3) の第 3 項に当てはめ，その値 $\lceil len[p_{26}]/2 \rceil + \textsc{Pop}(S) = 137$ を S にプッシュする．

最後に S の値 137 を $|(N_3, [p_1])|$ として出力する．

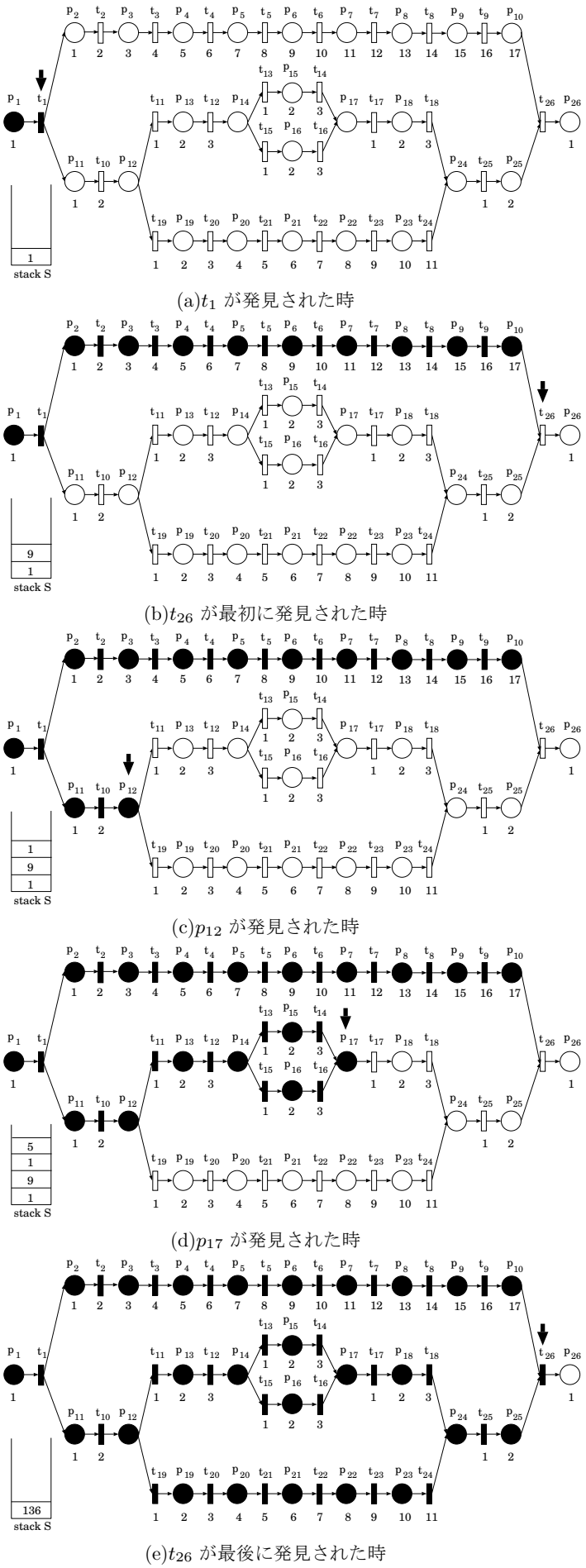
5. おわりに

本稿では，まず状態数計算問題を定式化し，それは可解であるが，手に負えないことを示した．また RAWS WF ネットに対する多項式時間の解法を与えた．今後の課題として，より一般的なクラスに対する解法を提案すること，ならびに計算した状態数の活用法を考案することなどが挙げられる．

謝辞 本研究の一部は（株）インタフェース社の援助を受けたことを記して謝意を表する．

参考文献

[1] T. Murata, “Petri nets: properties, analysis and applications,” Proc. IEEE, vol.77, no.4, pp.541–580, 1989.
[2] D.Y. Chao, “Number of reachable states for simple classes of Petri nets” Proc. IECON 2011, pp.3788–3791, 2011
[3] W.M.P. van der Aalst, “The application of Petri nets to workflow management,” J. Circuits, Systems, Computers, vol.8, no.1, pp.21–65, 1998.
[4] J. Esparza and M. Nielsen, “Decidability issues for Petri nets,” Bulletin of the EATCS 52, pp.244–262, 1994.
[5] R.M. Karp and R.E. Miller, “Parallel program schemata,” Journal of Computer and System Sciences vol.3, issue.2, pp.147–195, 1969.



(a)t_1 が発見された時
(b)t_{26} が最初に発見された時
(c)p_{12} が発見された時
(d)p_{17} が発見された時
(e)t_{26} が最後に発見された時

図 9 $(N_3, [p_1])$ に対する提案アルゴリズムの動作