

Title	ペトリネットの発火系列問題についての研究
Author(s)	田中博英
Citation	
Issue Date	2001-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/1431
Rights	
Description	Supervisor:平石 邦彦, 情報科学研究科, 修士

修 士 論 文

**ペトリネットの発火系列問題
についての研究**

指導教官 平石邦彦 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

田中博英

2001 年 2 月 15 日

目次

1	はじめに	1
2	諸定義	3
2.1	ペトリネット	3
2.2	ペトリネットの発火系列問題 (LFS)	5
3	LFS の NP 困難性 (問題の複雑さ)	7
4	パーシステントペトリネットの LFS	9
4.1	パーシステンシィの基本的性質	10
4.2	アルゴリズム	11
5	パーシステント集合を利用した解法	15
5.1	パーシステント集合	15
5.2	パーシステント集合の計算方法	19
6	実験	20
6.1	深さ優先探索 (DFS)	20
6.2	パーシステント集合を利用した探索 (PSS)	20
6.3	5 人の哲学者問題	21
6.4	実験結果	23
6.4.1	実験 1	23
6.4.2	実験 2	24
6.4.3	実験 3	25
6.5	考察	25
6.5.1	実験 1	25
6.5.2	実験 2	25

6.5.3 実験 3	27
7 おわりに	28
謝辞	29

第 1 章

はじめに

ペトリネットは離散事象システムのモデル化やシミュレーションに広く用いられているモデルであり、組み立てライン、プラントの運転過程、OS、通信プロトコル、非同期回路など数多くの応用例がある。ペトリネットの発火系列問題はシステムの挙動解析における基本問題の 1 つであり、NP-困難であることが示されている。そのため従来研究では、ペトリネットの発火系列問題および、最適発火系列問題に対するヒューリスティックな解法について議論されてきた。

ペトリネットの発火系列問題 (LFS: Legal Firing Sequence Problem of Petri nets) とは、ペトリネット PN 、初期マーキング M 、発火回数ベクトル X (各成分 $X(t)$ がトランジション t の発火回数を表す) が与えられたとき、各トランジション t がその中に合計でちょうど $X(t)$ 回現れ、かつ M から順に発火可能であるようなトランジションの系列が存在するか否かを判定し、さらに存在するならばそのような系列の 1 つを求める問題であり、システムの動的な性質の解析を行なう際に重要な問題となる。ペトリネットの発火系列問題 (LFS) は、ペトリネット理論でよく知られたマーキング可到達問題、最小初期資源配置問題、活性問題、スケジューリング問題、等において、部分問題として、あるいは単純化された問題として表れる基本的な問題である。しかし LFS の解法は簡単ではなく、単純な構造を持つペトリネットのクラスに対してさえ LFS は NP-困難であることが Watanabe, Mizobata, Onaga らによって証明されている [6]。従って実用規模の問題を現実的な時間で解くことができるような、ヒューリスティックな解法の研究が行われてきた。パーシステントなペトリネットに限定して LFS を考えると、その LFS を解くアル

ゴリズムが知られている [1].

本研究では効率的な発火系列問題の解法を提案することを目的とし, さらに実装を行ない実験的に評価する. 具体的にはパーシステント集合の考えを用いて, 探索において障害となる実行系列のインターリービングによる状態数の爆発を抑える.

本論文の全体の章構成を説明する. 本章では, 本研究の背景および目的を述べたが, 2章では, ペトリネットとペトリネットの発火系列問題 (LFS) の諸定義を与える. 3章では, LFS の NP-困難性について, Watanabe, Mizobata, Onaga らの証明する際に使用したペトリネットモデルを用いて述べる. 4章では, パーシステントペトリネットについて述べ, 5章では, 過去に考えられた発火系列問題のアルゴリズムを参考にしてパーシステント集合を利用した解法の提案をし, 最後に6章で実験を行ない, 得られた結果について考察をする.

第 2 章

諸定義

2.1 ペトリネット

ペトリネットは次の 3 つの側面を持つモデルである.

(1) 数学的モデル (2) 図的モデル (3) 実行可能なモデル

すなわち, 記述されたモデルがそのままシミュレーションにより実行でき, その動的な過程が図で表現できるという点でうまく利用すれば, ソフトウェア・ツール化に適した強力なモデルとなり得る.

定義 2.1

ペトリネットとは 4 項組 $PN=(P, T, A, M_0)$ である. ここで P はプレース (place) の有限集合, T はトランジション (transition) の有限集合 ($P \cap T = \emptyset$), $A: (P \times T) \cup (T \times P) \rightarrow \mathbf{N}$ はプレースとトランジションの接続を表す写像, $M_0: P \rightarrow \mathbf{N}$ は初期マーキング (initial marking) である. 初期マーキングを除いた $N=(P, T, A)$ をネット構造 (net structure) という. ■

ペトリネットはプレースとトランジションという 2 種類の節点を持つ有向 2 部グラフとして見ることができ, 図 2.1 のように描く.

プレースは $\bigcirc$ で, トランジションは $\square$ または $|$ で, 写像 A はアーク (arc) $\rightarrow$ により表す. $A(x, y) = n \geq 1$ のとき n 本のアーク (または本数を枝の上に書いた 1 本のアーク) を接点 x から接点 y へ向けて描く. 初期マーキングは各プレース p 内に $M_0(p)$ 個の $\bullet$ (こ

れをトークン (token) と呼ぶ) を描くことにより表す. プレースにトークンが存在するとき, そのプレースはマークされている (marked) という. $A(x, y)$ の値をアークの重みという. 節点 $x \in P \cup T$ について, $\bullet x = \{y \mid (y, x) \in A\}$, $x^\bullet = \{y \mid (x, y) \in A\}$ をそれぞれ x の前集合, 後集合と呼ぶ. また $X \subset P \cup T$ について, $\bullet X = \bigcup_{x \in X} \bullet x$, $X^\bullet = \bigcup_{x \in X} x^\bullet$ とする.

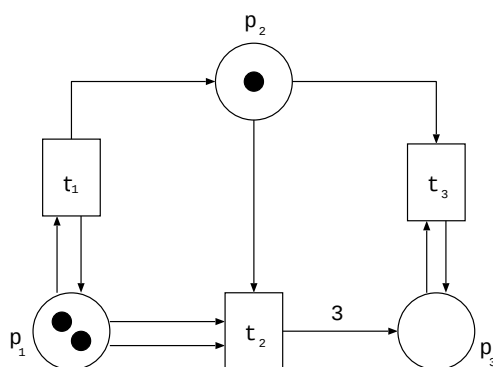


図 2.1: ペトリネットの図的表現

各プレースに存在するトークンの個数を表す写像 $M : p \rightarrow \mathbb{N}$ をマーキング (marking) と呼ぶ. マーキングはオートマトンにおける状態に対応し, 初期値 M_0 からトランジションの発火 (firing) により推移していく. トランジション t は各入力プレース p 上にその間のアークの重みである $A(p, t)$ 個以上のトークンが存在するとき発火可能 (enabled) であるという. 発火可能なトランジションは発火させることができる. トランジション t の発火により次のマーキング M' に変化する.

$$M'(p) = M(p) + A(t, p) - A(p, t) (p \in P) \quad (2.1)$$

トランジションの各入力プレースからアークの重みのトークンが失われ, 出力プレースにはアークの重みのトークンが加えられる. マーキング M からトランジション t の発火によりマーキング M' に推移することを $M[t > M'$, または $M \xrightarrow{t} M'$ と書く. また単に発火可能のみを表すときには $M[t >$, または $M \xrightarrow{t}$ と書く.

発火させたトランジションを並べた系列を発火系列 (firing sequence) という. 空系列 ϵ を含む有限の長さの発火系列全体の集合を T^* により表す. 発火系列 $\sigma = t_1 t_2 \cdots t_n \in T^*$

について, $M[t_1 > M_1[t_2 \cdots M_{n-1}[t_n > M'$ が成り立つとき, マーキング M において σ は発火可能であるという. 空列 ϵ は任意のマーキングで発火可能であり, 発火によりマーキングを変化させないものとする. すなわち $M[\epsilon > M$ である. また $M[\sigma > M'$ であるような発火系列が存在するとき, マーキング M' はマーキング M から到達可能 (reachable) であるといい, これを $M[> M'$ で表す.

写像 $\psi : T^* \rightarrow \mathbb{N}^T$ は発火系列中に各トランジションが出現する回数を与える. 例えば $\psi(abaabbc)(a)=3$ である. $\psi(\sigma)$ を発火系列 σ の発火回数ベクトル (firing count vector) と呼ぶ. T の部分集合 T' が与えられたとき, $PN|T'$ はトランジション T' とそれにアークにより接続される全てのプレースから成る部分ネットを表す. $PN|_{\text{supp}(x)}$ は, あるマーキングにおいて, 残りの発火回数ベクトルに対して, その値が 1 以上のトランジションとそれにアークにより接続される全てのプレースから成る部分ネットを表す. T^* は T のすべての有限列の集合を表す. (空列 λ も含む) $M' = \text{succ}(M, t)$ はマーキング M からトランジション t の発火によって M' に遷移したことを表している. 系列 $\sigma \in T^*$ に対し $T_\sigma = \{t \in T | \psi(\sigma)(t) > 0\}$ となる. ベクトル $x \in \mathbb{N}^T$ に対し $T_x = \{t \in T | x(t) > 0\}$ となる. $\bar{\delta}(t)$ は系列 δ に含まれる t の個数を表す. Y は残りの発火回数ベクトルを表している. T_Y は残りの発火回数ベクトル Y に対して, その値が 1 以上のトランジションの集合. $\text{en}(M)$ はマーキング M において発火可能なトランジションの集合を表している.

2.2 ペトリネットの発火系列問題 (LFS)

ペトリネットの発火系列問題 (LFS: Legal Firing Sequence Problem of Petri nets) とは, ペトリネット PN , 初期マーキング M_0 , 発火回数ベクトル X (各成分 $X(t)$ がトランジション t の発火回数を表す.) が与えられたとき, 各トランジション t がその中に合計でちょうど $X(t)$ 回現れ, かつ M_0 から順に発火可能であるようなトランジションの系列 δ が存在するか否かを判定し, さらに存在するならばそのような系列の 1 つを求める問題である. 発火系列 δ を X を実現する M からの発火系列と呼ぶ. LFS の入力を (PN, M, X) , 又 X が明らかな場合は省略して (PN, M) と表す.

ペトリネットの発火系列問題は, システムの動的な性質の解析を行なう際に重要な問題となる. ペトリネットの発火系列問題は, ペトリネット理論でよく知られたマーキング可

到達問題，最小初期資源配置問題，活性問題，スケジューリング問題，等において，部分問題として，あるいは簡単化された問題として表れる基本的な問題である．しかし LFS の解法は簡単ではなく，単純な構造を持つペトリネットのクラスに対してさえ LFS は NP-困難であることが示されている．従って実用規模の問題を現実的な時間で解くことができるような，ヒューリスティックな解法の研究が行なわれてきた．

第 3 章

LFS の NP 困難性 (問題の複雑さ)

NP-困難である 3-satisfiability problem (3-充足可能性問題) を多項式時間で LFS に還元可能であることを示すことにより LFS の NP-困難性が証明された [6].

以下に例を挙げる. ここでペトリネット PN は $|M| = 1$, $X = \bar{1}$ で T -インバリアントとする.

例 3.1

3-SAT において, 変数 $L = \{v_1, v_2, v_3\}$, 節 $C = \{c_1 = \{v_1, \bar{v}_2, v_3\}, c_2 = \{\bar{v}_1, v_2, v_3\}\}$ であるならば, これに相当するペトリネット PN は図 3.1 に示したものである.

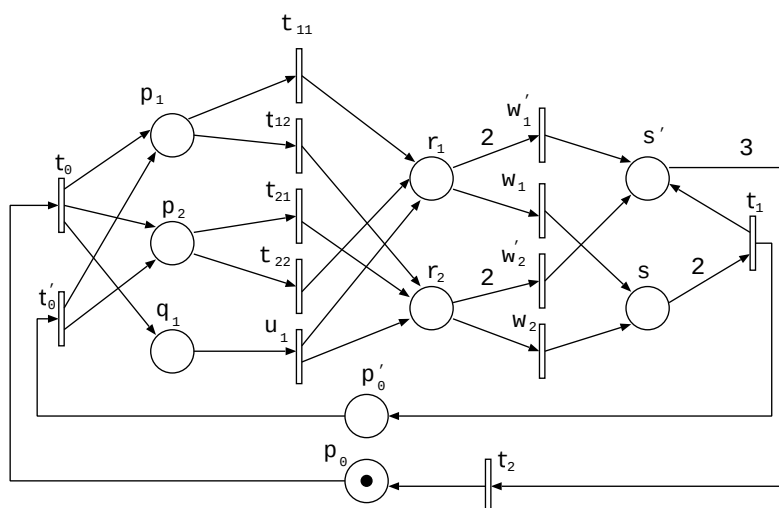


図 3.1: 例 1 の 3-SAT から構成された PN

この場合 **3-SAT** は以下のようになる．

リテラル v_1, v_2, v_3 が与えられ，変数 $L = \{v_1, v_2, v_3\}, \bar{L} = \{\bar{v}_1, \bar{v}_2, \bar{v}_3\}$ ，リテラルの集合である節 $C = \{c_1 = \{v_1, \bar{v}_2, v_3\}, c_2 = \{\bar{v}_1, v_2, v_3\}$ が与えられたとき，変数 (v_1, v_2, v_3) に値 $true(T)$ または $false(F)$ を割り当て，どの節にも値が T である変数が少なくとも 1 つ含むようにすることができるか．

$v_1 = T, v_2 = T, v_3 = T$ のとき図 3.1 のペトリネットで発火系列問題を実際に解く．まず t_0 を発火させると $t_{11}, t_{12}, t_{21}, t_{22}, u_1$ が発火可能となる．ここで v_1 と $t_{11}, \bar{v}_1$ と t_{12}, v_2 と $t_{21}, \bar{v}_2$ と t_{22}, v_3 と u_1 が対応しており， T のリテラルと対応するトランジションを発火させる． t_{11}, t_{21}, u_1 の発火後， w_1, w_2, t_1, t_0' と発火させていき，ここでは F のリテラルと対応するトランジション t_{12}, t_{22} を発火させる．その後， w_1', w_2', t_2 と発火させれば，すべてのトランジションを各 1 回ずつ発火させ，もとの状態に戻るような発火系列が存在する．他の場合についても発火系列問題を解くと以下の表のようになる．

表 3.1: 発火系列の有無

v_1	v_2	v_3	発火系列の有無
T	T	T	○
F	F	F	○
T	T	F	○
F	F	T	○
T	F	F	×
F	T	T	○
T	F	T	○
F	T	F	×

これは 3-SAT の答えと同じ結果になっており，発火系列の有無を調べることにより 3-SAT を解くことができる．

第 4 章

パーシステント ペトリネットの LFS

ペトリネット PN , マーキング M において任意の二つの発火可能なトランジション t に対して, 一方のトランジションが発火することにより他方が発火不可能となることがなければ, そのペトリネットはパーシステント (persistent) であると呼ばれる. つまりパーシステントならばマーキング M において以下が成り立つ. t_i, t_j は任意の発火可能なトランジションである. またこれが成り立つ性質をパーシステンシィ (persistency) という.

$$M[t_i > \wedge M[t_j > \Rightarrow M[t_i t_j > \quad (4.1)$$

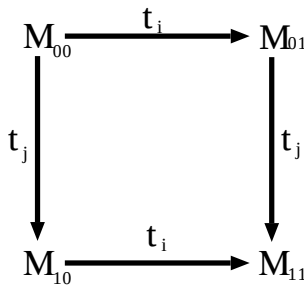


図 4.1: パーシステンシィの性質

マーキング M_{00} においてパーシステントで t_i, t_j が発火可能ならば, t_i, t_j の順に発火させることも t_j, t_i の順に発火させることも可能であり, 2つのトランジション t_i, t_j の発

火により得られるマーキングは共に同じマーキング M_{11} となる. (図 4.1)

4.1 パーシステンシの基本的性質

δ_1, δ_2 をマーキング M から発火可能な発火系列で $\delta_1 = t_0 \cdots t_{s-1}$ であるとする. $\delta_2 \odot \delta_1$ を新しい発火系列とし, 以下のように定義する. はじめに $\pi_0 = \delta_2$ として, $\pi_i (i = 0, \dots, s-1)$ を次のように再帰的に定義する. もし t_i が π_i に現れたなら π_{i+1} は π_i の中で一番はじめ (一番左) に現れる t_i を取り除いて得られた系列とし, t_i が π_i になければ $\pi_{i+1} = \pi_i$ とする. 最終的に $\delta_2 \odot \delta_1 = \pi_s$ となる.

例 4.1

$\delta_1 = t_0 t_1 t_2 (s = 3)$, $\delta_2 = u_1 u_2 t_1 t_0 t_1$ ならば

$$\pi_0 = u_1 u_2 t_1 t_0 t_1,$$

$$\pi_1 = u_1 u_2 t_1 t_1,$$

$$\pi_2 = u_1 u_2 t_1,$$

$$\pi_3 = u_1 u_2 t_1 = \delta_2 \odot \delta_1 \text{ となる.}$$

あるトランジションが δ_1, δ_2 の両方に現れたとき, かつそのときに限って $\delta_2 \odot \delta_1$ と $\delta_2 \delta_1$ は異なる.

定理 4.2

δ_1, δ_2 を M から発火可能な発火系列とする. もしペトリネット PN がパーシステントなら $\delta = \delta_1(\delta_2 \odot \delta_1)$ はマーキング M から発火可能な発火系列となり, どんなトランジション t に対しても $\bar{\delta}(t) = \max\{\bar{\delta}_1(t), \bar{\delta}_2(t)\}$ となる. ■

図 4.2 は例 4.1 の δ_1, δ_2 を用いて, 定理 4.2 を図的に証明したもので

$$\delta = \delta_1(\delta_2 \odot \delta_1) = t_0 t_1 t_2 u_1 u_2 t_1 \quad (4.2)$$

は M から発火可能な発火系列となる.

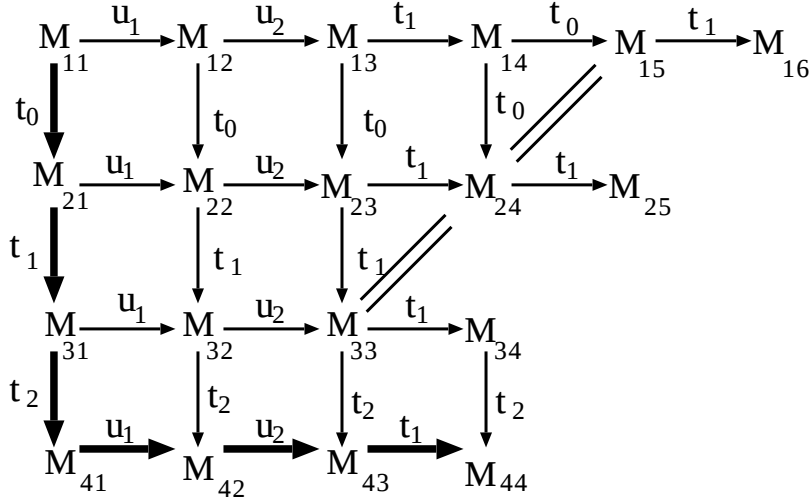


図 4.2: 定理 4.2 の図的証明

4.2 アルゴリズム

任意のマーキングでパーシステンシィであるペトリネット (パーシステントペトリネット) に限定して LFS を考えると, 与えられた発火回数ベクトル $x \in \mathbb{N}^T$ に対し, $M_0[\sigma > \wedge \psi(\sigma) = x]$ となる発火系列の存在を次のアルゴリズムによって判定することができる [1].

以下のアルゴリズムを考察してみる.

Algorithm :

Input: PN, X and M ; **Output:** “yes” or “no”

Step 1. $M' \leftarrow M$; $\delta \leftarrow \emptyset$; $F \leftarrow \{t \in T \mid x(t) \neq 0\}$; $Y \leftarrow X$;

Step 2. **While** $F \neq \emptyset$ かつ M' において発火可能な $t \in F$ が存在する。 **do**

$M' \leftarrow M'[t >]$; $\delta \leftarrow \delta t$; $Y(t) \leftarrow Y(t) - 1$;

if $Y(t) = 0$ **then** $F \leftarrow F - \{t\}$;

Step 3. **if** $F = \emptyset$ **then** output “yes”with δ

else output “no”.

アルゴリズム中の文字の定義は以下のようになる.

M' : 現在のマーキング

σ : 今までに発火させた系列

F : これから発火させるべきトランジションの集合

Y : 残りの発火回数ベクトル

このアルゴリズムに, あるパーシステントペトリネット PN , 初期マーキング M , 発火回数ベクトル X を入力した結果 no が出力されたならば, そのペトリネットには M から発火可能かつ発火回数ベクトル X を満たす発火系列は存在しないことの証明を以下に示す.

[証明]:

対遇, すなわち

ペトリネット PN に M から発火可能かつ発火回数ベクトル X を満たす発火系列が存在するならば, アルゴリズムが yes を出力する.

を証明する. M から発火可能かつ発火回数ベクトル X を満たす発火系列は存在すると仮定し, それを δ_0 (長さが s) とする. $t_1 \in F$ かつ M から発火可能な t_1 が存在する. (なぜなら少なくとも δ_0 の最初のトランジションは M から発火可能なトランジション.) 定理 4.2 より $\sigma_1 = t_1(\delta_0 \odot t_1)$ も M から発火可能かつ発火回数ベクトル X を満たす発火系列である.

$M[t_1 > M_1]$ とし, M_1 での残りの発火回数ベクトルを Y_1 とする. $t_2 \in F$ かつ M_1 から発火可能な t_2 が存在する. $\delta_0 \odot t_1 = \delta_1$ と置いて, 定理 4.2 より $\sigma_2 = t_2(\delta_1 \odot t_2)$ も M_1 から発火可能かつ発火回数ベクトル Y_1 を満たす発火系列である.

$M_1[t_2 > M_2]$ とし, M_2 での残りの発火回数ベクトルを Y_2 とする. $t_3 \in F$ かつ M_2 から発火可能な t_3 が存在する. $\delta_1 \odot t_2 = \delta_2$ と置いて, 定理 4.2 より $\sigma_3 = t_3(\delta_2 \odot t_3)$ も M_2 から発火可能かつ発火回数ベクトル Y_2 を満たす発火系列である.

以下同様にして $M_3, M_4, \dots, M_{s-1}$ の各マーキングにおいて発火可能なトランジションが存在し, かつ残りの発火回数ベクトルを満たす発火系列が存在する. よって上記のアルゴリズムに, あるパーシステントペトリネット PN , 初期マーキング M , 発火回数ベクトル X を入力した結果 no が出力されたならば, そのペトリネットには M から発火可能

かつ発火回数ベクトル X を満たす発火系列は存在しないことが証明された.

また逆に「yes が出力されたならば, ペトリネット PN には M から発火可能かつ発火回数ベクトル X を満たす発火系列が存在する」ことを証明するためには, 対遇である「ペトリネット PN には M から発火可能かつ発火回数ベクトル X を満たす発火系列は存在しないならば, アルゴリズムが no を出力する」を証明すればいいのだが, これはアルゴリズムを見れば自明である. ■

このアルゴリズムは演算回数が各トランジションの発火回数により決定されることより準多項式時間アルゴリズムである.

第 5 章

パーシステント 集合を利用した解法

5.1 パーシステント 集合

パーシステント集合 W とはトランジション T の部分集合でマーキング M において以下を満たす集合である.

定義 5.1

$t \in W, \sigma \in (T - W)^*$ とする,

$$M \xrightarrow{\sigma t} M' \wedge M \xrightarrow{t} M' \Rightarrow M \xrightarrow{t\sigma} M' \quad (5.1)$$

つまりマーキング M において発火系列 σt により M' に遷移でき, かつマーキング M において t が発火可能であるならば, 発火系列 $t\sigma$ によっても M から M' に遷移することができる. ■

命題 5.2

ペトリネット $PN = (P, T, A)$ で, T の部分集合 W はマーキング M において

$$\forall t \in W \cap en(M) \quad (5.2)$$

$$(\bullet t)^\bullet \subset W \quad (5.3)$$

ならばパーシステント集合である. ■

[証明]:

$t \in W$ および $\sigma \in (T - W)^*$ について $M \xrightarrow{\sigma t} \wedge M \xrightarrow{t}$ であるとする. t と競合する可能性のあるトランジション $((\bullet t) \bullet$ のトランジション) はすべて W に含まれるので, t の発火可能性は $T - W$ の任意のトランジションの発火により失われることはない. よって $M \xrightarrow{t\sigma}$ が成り立つ. ■

ペトリネット PN がパーシステントなら 4 章で示したアルゴリズムが使えるが一般の PN は常にパーシステントではない. ところが部分的に注目するとパーシステントな部分ペトリネットを含んでいる. ある状態で PN においてパーシステント集合が存在するとき発火可能なトランジションを全部発火させるのではなく, 発火可能かつパーシステント集合に含まれるトランジションを発火させ, 深さ優先探索を行ない, 状態数の爆発を抑えるアルゴリズムを提案する. つまりパーシステントな部分ネットを経由し, アルゴリズムの効率化を行なう.

Algorithm PSS:

Stack is empty;

push $\langle M_0, X, \lambda \rangle$ onto *Stack*;

While *Stack* is not empty **do**

pop $\langle M, Y, \sigma \rangle$ from *Stack*;

if $Y = \mathbf{0}$ **then** *output* σ and *halt*;

 compute a $W(M, Y)$;

for all t in $W(M, Y) \cap en(M)$ **do**

$M' := succ(M, t)$;

push $\langle M', Y - \psi(t), \sigma t \rangle$ onto *Stack*;

output: “no”.

ここで $W(M, Y) \subseteq T$ は以下の条件を満たす. Y は残りの発火回数ベクトルである.

1. 部分ネット $PN|_{T_Y}$ において $W(M, Y)$ はマーキング M でパーシステント集合.
2. マーキング M において発火不可能な $t \in W(M, Y)$ が存在するとき以下が成り立つ.

$$\sigma \in (T_Y - W(M, Y))^* : M \xrightarrow{\sigma} M' \Rightarrow t \notin en(M'). \quad (5.4)$$

これはマーキング M において $W(M, Y)$ 以外の発火させるべきトランジションの列 σ の発火によりマーキング M' に変化するならば, t はマーキング M' でもやはり発火できないことを意味する. つまり σ は発火不能の t を発火可能にすることはないことを定義は示している.

上記のアルゴリズムにおいて compute a $W(M, Y)$ の部分でパーシステント集合 $W(M, Y)$ を計算し処理するが, その方法は次節で論じる.

このアルゴリズム PSS が LFS を正しく解くことの証明を以下に示す.

[証明]:

$M \xrightarrow{\sigma} M'$ であり, $\psi(\sigma) = Y$ であり, マーキング M はアルゴリズム PSS により探索されたと仮定する. M' もアルゴリズム PSS により探索されることを σ の長さで帰納法を用いて証明する. $\sigma = 0$ のときは自明だから σ が 0 より大きいときを考える. 定義より $W(M, Y) \subseteq T_Y$ だから $W(M, Y)$ のすべてのトランジションは σ に含まれている. $t \in W(M, Y)$ が σ で最初に出現すると仮定する. すなわち $\sigma = \sigma_1 t \sigma_2$ and $T_{\sigma_1} \cap W(M, Y) = \emptyset$ となる. $W(M, Y)$ の条件 2. より t はマーキング M で発火可能となる. $W(M, Y)$ の条件 1. より $M \xrightarrow{t\sigma_1\sigma_2} M', \psi(\sigma'_1) = \psi(\sigma_1)$ となる σ'_1 が存在する. よって

$$M \xrightarrow{t} M_1 \xrightarrow{\sigma'_1} M', t \in W(M, Y), \psi(\sigma') = Y' \quad (5.5)$$

となる. Y' は M_1 での残りの発火回数ベクトルである. 以下同様に

$$M_1 \xrightarrow{t'} M_2 \xrightarrow{\sigma''} M', t' \in W(M_1, Y'), \psi(\sigma'') = Y'' \quad (5.6)$$

$$M_2 \xrightarrow{t''} M_3 \xrightarrow{\sigma'''} M', t'' \in W(M_2, Y''), \psi(\sigma''') = Y''' \quad (5.7)$$

と同じことを繰り返し行ない, $W(M, Y)$ のトランジションだけを発火させて M' に到達することができる. よって M' もアルゴリズム PSS により探索されることが示された. ■

5.2 パーシステント 集合の計算方法

この小節ではパーシステント集合を計算する方法を考える。

マーキング M において、パーシステント集合は複数存在することが可能である。あるマーキング M において発火可能かつパーシステント集合に含まれるトランジションを発火させるのだから、できるだけ小さい集合を求めることが重要となる。

パーシステント集合の計算方法としては、マーキング M において、発火可能なトランジションの中から任意の1つのトランジションを選び、パーシステント集合をそれ1つからなるの集合からスタートし、そのトランジションに「干渉する (interfere)」すべてのトランジションを加えていき、新たなトランジションを加える必要がなくなるまで繰り返して決める。より小さい集合を求めるためには、パーシステント集合の1つ目のトランジションを決める際に、なるべく後集合が多くあるようなプレースからの入力が少ないトランジションを選び出すとよいと考えられる。なぜなら、トランジションに干渉するすべてのトランジションを加えていくからである。

パーシステント集合を計算する方法として以下の手順を踏む。

1. マーキング M において発火可能なトランジション t の集合 $\{en(M)\}$ から任意のトランジション t を1つ選び $t \in W(M, Y)$ とする。このとき、後集合が多くあるようなプレースからの入力が少ないトランジションを選び出すようにする。

2. 命題 5.2 より、マーキング M で $t \in W(M, Y)$ と競合する可能性のあるトランジション $(\bullet t) \bullet$ があればそれらをすべて $W(M, Y)$ に加える。

3. $W(M, Y)$ の条件 2. より、マーキング M において発火不可能な $t \in W(M, Y)$ がある場合、その t を発火可能にさせるトランジションがあればそれらをすべて $W(M, Y)$ に加える。

4. 2. と 3. を繰り返し行い、 $W(M, Y)$ が変化しなくなったところで $W(M, Y)$ を決定する。

第 6 章

実験

ペトリネットの発火系列問題 (LFS) の実装を行ない実験的に評価する．深さ優先探索とパーシステント集合を利用した探索について到達可能性の判定が終わるまでの時間を比較し考察する．

6.1 深さ優先探索 (DFS)

初期マーキング M においてまず，発火可能なトランジション t の存在を調べ発火可能なトランジションの番号をスタックにプッシュする．スタックからトランジションの番号を 1 つポップしてそれを発火させる．遷移先のマーキング M においても，発火可能なトランジション t の存在を調べるが，もしなければバックトラックをして 1 つ前のマーキングに戻り，発火可能だがまだ発火させていないトランジションを発火させる．これを繰り返し行ない発火回数ベクトルを満たすような発火系列を求める．

6.2 パーシステント集合を利用した探索 (PSS)

あるマーキング M において発火可能なトランジション t の存在を調べた上で，前章で述べた方法によりパーシステント集合を計算する．発火可能なトランジション t かつパーシステント集合に含まれるトランジションの番号をスタックにプッシュする．このように深さ優先で探索する．

6.3 5 人の哲学者問題

例として 5 人の哲学者問題 (dining philosophers problem) を扱う。5 人の哲学者問題は Dijkstra[1968] によって提案されたものであり、思考と食事を繰り返す 5 人の哲学者に関するものである。哲学者は大きな円卓についており、その円卓には数多くの料理がのっている。哲学者の間にはそれぞれ一本ずつフォークが置いてある。料理を食べるにはフォークが 2 本いるので、哲学者のそれぞれは右側と左側の両方のフォークを手にとらなければならない。問題は、すべての哲学者がいっせいに左側のフォークを取り上げ、右側のフォークが空くのを待つとすれば、未来永劫に待つことになる。これがデッドロック (deadlock) 条件である。プレースにトークンの存在する状態が条件、トランジションが事象を表すものとして、この問題をペトリネットのモデルにしたものを図に示す。

プレース f_1, f_2, f_3, f_4, f_5 にトークンのある状態をフォークが使用可能であることをあらわし、最初は全てのフォークが円卓に置かれている状態なので、初期マーキングはこれらのプレースにトークンが存在する。哲学者のそれぞれを 2 つのプレース m_i と e_i によってあらわし、それぞれのプレースは冥想状態と食事状態をあらわす。ある哲学者が冥想状態から食事状態に移行するためには、左側の 1 本と右側の 1 本の 2 つのフォークが使用可能でなければならない。このモデル化では、各哲学者が食事に入るとき左右のフォークを同時に取り上げることでデッドロックを回避している。また哲学者全員が冥想している状態を初期状態とし、初期マーキングはこれらのプレースにトークンが存在する。

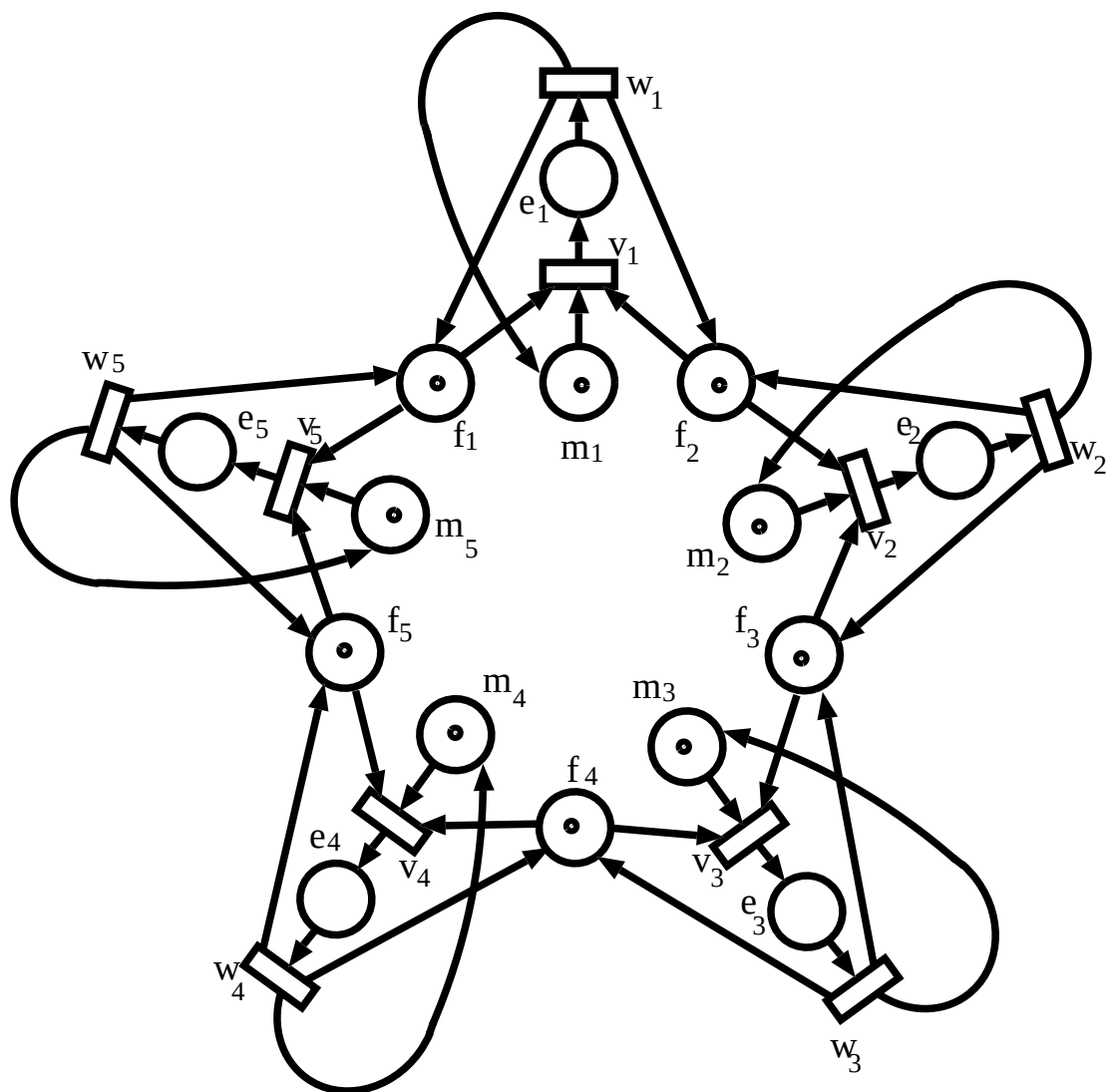


図 6.1: 5 人の哲学者問題のペトリネットモデル

6.4 実験結果

6.4.1 実験 1

まずそれぞれのトランジションの発火回数をすべて同じに設定して実験を行なった．結果は以下ようになった．この場合，与えられた発火回数ベクトルを満たす発火系列は存在し，得られた時間はその発火系列の 1 つを見つけ出すまでの時間である．

表 6.1: 実験結果 1

人数	発火回数	DFS(秒)	PSS(秒)
5	1	0.0010	0.0013
5	2	0.0011	0.0016
5	3	0.0012	0.0018
5	4	0.0014	0.0021
5	5	0.0015	0.0024

表 6.2: 実験結果 2

人数	発火回数	DFS(秒)	PSS(秒)
10	2	0.0025	0.0041
15	2	0.0064	0.0105
20	2	0.0155	0.0230
30	2	0.0941	0.1356
50	2	0.3081	0.4725

6.4.2 実験 2

次に、すべてのトランジションの発火回数を同じにするのではなく、トランジションによって発火の回数を変えてみたところ、結果は以下のようになった。この場合、与えられた発火回数ベクトルを満たす発火系列は存在せず、得られた時間は発火系列が存在しないことが判明するまでの時間である。また BT はバックトラックの回数である。

表 6.3: 実験結果 3

人数	発火回数										DFS		PSS	
	v_1	w_1	v_2	w_2	v_3	w_3	v_4	w_4	v_5	w_5	時間 (秒)	BT (回)	時間 (秒)	BT (回)
5	1	2	1	2	1	2	1	2	1	2	0.1007	6,450	0.0043	120
5	1	2	1	2	1	2	1	2	2	2	0.4612	32,934	0.0059	195
5	1	2	1	2	1	2	2	2	2	2	2.6011	177,006	0.0127	510
5	1	2	1	2	2	2	2	2	2	2	16.8407	1,301,030	0.0384	1,091
5	1	2	2	2	2	2	2	2	2	2	165.8431	12,453,830	0.0519	1,765

表 6.4: 実験結果 4

人数	発火回数										DFS		PSS	
	v_1	w_1	v_2	w_2	v_3	w_3	v_4	w_4	v_5	w_5	時間 (秒)	BT (回)	時間 (秒)	BT (回)
5	2	1	2	1	2	1	2	1	2	1	1.0091	74,535	0.0071	257
5	2	1	2	1	2	1	2	1	1	1	0.6019	53,130	0.0084	321
5	2	1	2	1	2	1	1	1	1	1	0.3050	31,725	0.0057	192
5	2	1	2	1	1	1	1	1	1	1	0.1625	18,108	0.0048	177

6.4.3 実験 3

以下のように発火回数を設定して実験を行なった．この場合，与えられた発火回数ベクトルを満たす発火系列が存在し，得られた時間はその発火系列の 1 つを見つけ出すまでの時間である．

表 6.5: 実験結果 5

人数	発火回数										DFS		PSS	
	v_1	w_1	v_2	w_2	v_3	w_3	v_4	w_4	v_5	w_5	時間 (秒)	BT (回)	時間 (秒)	BT (回)
5	5	5	6	5	5	5	5	5	5	5	0.1112	9,225	0.0025	5
5	4	4	6	5	4	4	4	4	4	4	0.0339	2,004	0.0022	4
5	3	3	6	5	3	3	3	3	3	3	0.0063	411	0.0020	3
5	2	2	6	5	2	2	2	2	2	2	0.0021	74	0.0017	2
5	1	1	6	5	1	1	1	1	1	1	0.0012	9	0.0015	1

6.5 考察

6.5.1 実験 1

表を見てわかるように，実験 1 の場合はすべてにおいて PSS の方が DFS より発火系列の発見が遅い．実験したときにバックトラックの回数もそれぞれでカウントしたが，すべての場合においてバックトラックはなかった．DFS は発火可能なトランジションから順に発火させるアルゴリズムであるのに比べて，PSS は発火可能なトランジションを調べ，さらにそのトランジションの集合からパーシステント集合を調べるというアルゴリズムであるから，PSS は各マーキングにおいて，オーバーヘッドの時間が原因となって，DFS に比べて発火させるまでの時間が余分にかかったものと思われる．

6.5.2 実験 2

いずれの場合も PSS の方が DFS より発火系列が存在しないことが判明するまでの時間が早い．トランジション v_1 の発火回数を 1，それ以外のトランジションの発火回数を 2 としたときは DFS と PSS の時間差は約 165 秒，DFS は PSS の約 3200 倍もの時間を要す

ることになる.

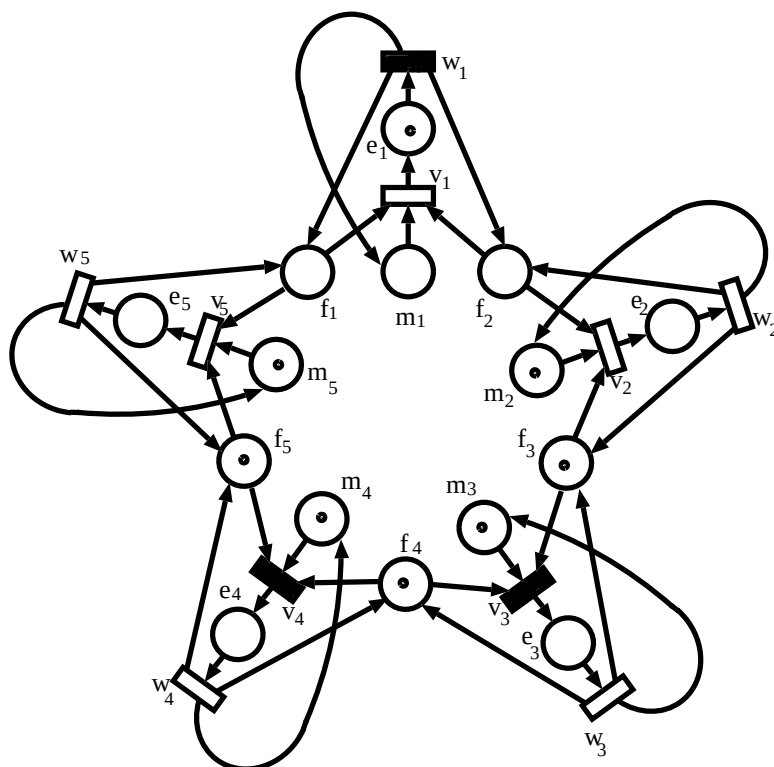


図 6.2: w_1, v_3, v_4 が発火可能な5人の哲学者問題のペトリネットモデル

図 6.2 のように黒いトランジション w_1, v_3, v_4 が発火可能であるとき, DFS においては w_1, v_3, v_4 のすべてのトランジションがスタックに入る.

PSS ではパーシステント集合の計算をする際, スタートのトランジションを w_1 以外にすればパーシステント集合が v_3, v_4 となり, パーシステント集合かつ発火可能なトランジションは v_3, v_4 になる. しかしスタートするトランジションを w_1 に決めてやればパーシステント集合は w_1 のみとなり, パーシステント集合かつ発火可能なトランジション w_1 のみがスタックに入る. このようにスタートするトランジションを後集合が多くあるようなブレース, からの入力が少ないトランジションを選び出すようにすることによって状態数の爆発をおさえることが可能となり, バックトラックの回数が DFS に比べて少なくなり結果的にペトリネットの到達可能性を判定するまでの時間が短くできたものと思われる.

6.5.3 実験 3

トランジション v_2, w_2 の発火回数をそれぞれ 6 回, 5 回, 他のトランジションを 1 回に設定したものは PSS より DFS の方が早く探索されたがそれ以外の場合は PSS が DFS より早く発火系列を見つけ出した. 実験 2 のときと同じく, DFS の場合はバックトラック数が多くなりそれが影響しているものと思われる.

第 7 章

おわりに

本研究では，ペトリネットの発火系列問題 (LFS) を解くにあたり従来から知られているパーシステントペトリネットの場合のアルゴリズムを拡張してパーシステント集合の考えを用いたアルゴリズム PSS により，実験的に評価した．

一般のペトリネットにおいて，バックトラックが発生しないように発火回数が与えられたペトリネットについては，アルゴリズム PSS を用いるとアルゴリズム DFS に比べてパーシステント集合を計算する時間が余分にかかる．このオーバーヘッドの時間が原因となり，DFS が PSS よりも早く発火系列を見つけ出すこととなる．

一方バックトラックが多く発生するように発火回数が与えられたペトリネットについては，PSS ではパーシステント集合の計算をすることにより状態数の爆発をおさえることが可能となり，バックトラックを減らすことができる．結果的にアルゴリズム PSS を用いるとアルゴリズム DFS に比べて到達可能性の判定時間または発火系列探索の時間を早くすることができた．

ペトリネットの発火系列問題について従来研究と比較すると，従来手法はデッドロックを起こすような特殊構造であるサイホンに注目したヒューリスティックな手法であったのに対して本研究で提案した手法は従来研究とは異なり，パーシステント集合に着目したものであり，従来手法と組み合わせて使用できる．すなわち，従来手法をより効率化できるものと考えられる．

謝辞

本研究を進めるにあたり本学 平石邦彦 助教授より数多くの御指示，御助言を，そして本論文の構成にあたり数多くの有益な御指導を頂きました．平石邦彦 助教授の最適な御指導に感謝いたします．

有益な御助言を頂きました宋少秋助手および高島康裕助手に感謝致します．

また本論文をまとめるにあたり，御協力いただいたシステム基礎講座の諸兄に厚く御礼申し上げます．

参考文献

- [1] 渡邊敏正, 山内雅弘, ペトリネット発火系列の概説. 1997 信学技報 Vol. 97 CAS97-74 pp15-22
- [2] 山内雅弘, 渡邊敏正, ペトリネットの発火系列問題に対する新しいヒューリスティックアルゴリズム. 1997 信学技報 CAS97-73 pp7-14
- [3] 山内雅弘, 中村伸治, 渡邊敏正, ペトリネットの発火系列問題に対する近似アルゴリズム. 1993 信学技報 Vol. 93 アルゴリズム 33-3
- [4] 山内雅弘, 橋本通高, 渡邊敏正, ペトリネットの発火系列問題に対する発見的解法 FSD. 1998 信学技報 COMP 98-6 pp41-48
- [5] Kunihiro Hiraishi, Persistency and Weak Persistency in Partial Order Based Analysis. 1997 Proc. IEEE SMC Vol. 5 pp4285-4290
- [6] T. Watanabe, Y. Mizobata and K. Onaga. Time complexity of legal firing sequence and related problems of Petri nets Trans. IEICE of Japan Vol. E72 No. 12 pp1400-1409 December 1989
- [7] K. Morita, H. Nakawaki and T. Watanabe The legal firing sequence problem of Petri nets with State Machine Structure. Technical Report COMP95-2 IEICE of Japan January 1995
- [8] K. Morita and T. Watanabe The legal firing sequence problem for weighted state machines. Technical Report CST95-33 IEICE of Japan January 1996