SAT 技術を用いた正規ペトリネットのデッドロック検出手法の提案

寸田 智也　宋 剛秀　番原 睦則　田村 直之

本論文では，SAT 技術を用いた正規ペトリネット (各プレイスのトークン数が整数値，アークの多重度が 1) のデッドロック検出について述べる．与えられたペトリネットは整数上の制約充足問題として表現され，順序符号化法を用いて SAT 問題に変換される．また，デッドロックの検出には有界モデル検査の手法を用いる．ペトリネットの制約モデルとしては，まず基本モデルを提案し，次にトークン数の変化量に制限を加えた制限モデルと，多重発火を可能にした多重発火モデルを提案する．制限モデルは，基本モデルに比べデッドロック検出までの遷移回数が多くなるが，より少ない節数で表現することができる．一方，多重発火モデルは，基本モデルに比べ節数が多くなるが，より少ない遷移回数でデッドロックを検出できる．これら 3 種類の制約モデルに関し，Model Checking Contest 2015 デッドロック検出部門の問題を用いて提案手法を評価した．既存のツールがいずれもデッドロック検出に失敗した問題でデッドロックを検出でき，提案手法の有効性が示せた．

1　はじめに

C. A. Petri によって提唱されたペトリネットは，コンピュータ・通信システムなどの並行的，非同期的，分散的，非決定的，確率的な動作をする一般的な情報・制御システムの記述・設計・解析・検証に有用である [13].

命題論理式の充足可能性判定 (SAT) は，与えられた命題論理式が真になる値割り当てが存在するかどうかを判定する問題である [7]．この SAT を解くアルゴリズムおよびそれを実装したソルバーに関して，半世紀前からこれまでに膨大な研究がなされており，近年，SAT ソルバーの性能は飛躍的に向上した．それを受けて，SAT ソルバーはさまざまな分野に応用されるようになっており，ペトリネットの検証にも用いられるようになった．SAT ソルバーでは，直接的には整数変数を扱うことができないため，既存の SAT 型ペトリネット検証手法の多くは，各プレイスのトークン数が 1 以下の安全ペトリネットのみを対象にしている [12] [14].

そこで本論文では，順序符号化 [17] を用いて整数上の制約を SAT 符号化することで，各プレイスのトークン数が整数値となる正規ペトリネットのデッドロック検出を行う．順序符号化は，整数変数上の制約充足問題を SAT 問題に符号化する手法の 1 つで，多くの問題に対してよい性能を示している．

デッドロック検出には有界モデル検査の手法を用いる [1] [2] [3]．有界モデル検査では，動的なシステムについて，遷移長 k の状態遷移と検証すべき性質を命題論理式で表し，その命題論理式を SAT ソルバーで検証する．この手法でデッドロックを効率的に検出するために考慮すべき点として，SAT 問題として表した場合の節数と検証する遷移長 k の長さがある．そこで本論文では，ペトリネットを制約充足問題として定式化した制約モデルを 3 種類提案し比較する．最初に基本モデルを提案し，次に節数を抑えるように基本モデルを改良した制限モデルと，遷移長 k を抑え

Proposal of a SAT-based Method to Detect Deadlocks of Ordinary Petri Nets.

Tomoya Sunda, 神戸大学大学院システム情報学研究科, Graduate School of System Informatics, Kobe University.

Takehide Soh, Mutsunori Banbara, Naoyuki Tamura, 神戸大学情報基盤センター, Information Science and Technology Center, Kobe University.

るように基本モデルを改良した多重発火モデルを提案する．制限モデルは，基本モデルにトークン数の変化量に制限を加えた制約モデルである．多重発火モデルは，基本モデルに多重発火を可能にした制約モデルである．

以下，2 節では，SAT と SAT 型制約ソルバーについて説明する．3 節では，ペトリネット，およびそのデッドロック検出，Model Checking Contest について説明する．4 節では，ペトリネットの 3 つの制約モデルと，デッドロック状態の制約式，デッドロックの探索手法について説明する．5 節では，Model Checking Contest 2015 デッドロック検出部門の問題を用いて，3 種類の制約モデルの特徴を比較する．また，既存手法との比較を行う．既存のツールがいずれもデッドロック検出に失敗した問題でデッドロックを検出でき，提案手法の有効性が示せた．最後に，6 節で結論をまとめる．

2 SAT と SAT 符号化

2.1 SAT

充足可能性判定 (Satisfiability testing problem; SAT) 問題とは，与えられた命題論理式を真にするような命題変数の値割り当てが存在するかどうかを判定する問題である [7]．与えられた命題論理式を真にするような値割り当てが存在する場合，命題論理式は充足可能 (`SAT`) であるといい，一方命題論理式を真とする値割り当てが存在しないとき，充足不能 (`UNSAT`) であるという．

SAT 問題を解くソルバーは SAT ソルバーと呼ばれる．SAT ソルバーは連言標準形 (CNF) の命題論理式を受け取って，`SAT` か `UNSAT` かを判定する．近年の SAT ソルバーは求解過程で発生する矛盾から得られる学習節を利用し，解を探索する CDCL アルゴリズムなどの高速化技術が取り入れられており，その性能は飛躍的に向上している [10]．これに伴い，与えられた問題を SAT 問題に変換し，SAT ソルバーを用いて求解する SAT 型手法が用いられるようになっており，システム検証にも用いられている [1]．

2.2 SAT 符号化

与えられた制約式をすべて満たすことができるような変数の値の割当てを決定する問題を, 制約充足問題 (Constraint Satisfaction Problem; CSP) という．SAT 型制約ソルバーは与えられた制約充足問題を SAT に符号化し，SAT ソルバーを用いて与えられた制約充足問題の解を求めるシステムである．

制約充足問題から SAT への符号化法には以下のようなものがある．

- 直接符号化法 [4] [18]
- 支持符号化法 [6] [9]
- 対数符号化法 [5] [8]
- 順序符号化法 [17]

中でも順序符号化法はショップ・スケジューリング問題および 2 次元ストリップパッキング問題で未知だった最適値決定に成功する等，有望な手法であることが示されている [16] [15]．

3 ペトリネットのデッドロック検出

本節では，正規ペトリネットとデッドロックについて説明する．また，ペトリネットの検証のコンテストである Model Checking Contest や，既存の SAT 型ペトリネット検証手法について述べる．

3.1 ペトリネット

正規ペトリネットは，トークン数が整数値，アークの多重度が 1 のペトリネットであり，組 $(\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{M}_0)$ で表される．ここで，$\mathbf{P} = \{p_1, p_2, ..., p_m\}$ はプレイスの集合，$\mathbf{T} = \{t_1, t_2, ..., t_n\}$ はトランジションの集合，$\mathbf{A} \subseteq (\mathbf{P} \times \mathbf{T}) \cup (\mathbf{T} \times \mathbf{P})$ はアークの集合，$\mathbf{M}_0 : \mathbf{P} \to \mathbb{N}$ は初期マーキングである．また，$\bullet p$ はプレイス p の入力トランジションの集合を表し，$p\bullet$ はプレイス p の出力トランジションの集合を表す．同様に，$\bullet t$ はトランジション t の入力プレイスの集合，$t\bullet$ はトランジション t の出力プレイスの集合を表す．

ペトリネットの状態はマーキング $\mathbf{M} : \mathbf{P} \to \mathbb{N}$ で表される．また，常に $\mathbf{M} : \mathbf{P} \to \{0, 1\}$ となるペトリネットを安全ペトリネットという．図 1 にペトリネットの例を示す．

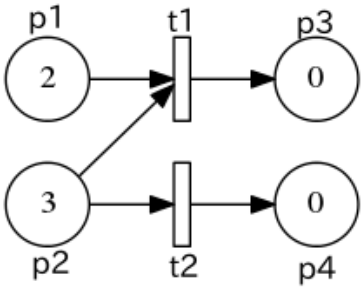

図 1　ペトリネットの例

この例は，$\mathbf{P} = \{p_1, p_2, p_3, p_4\}$，$\mathbf{T} = \{t_1, t_2\}$，

$$\mathbf{M}_0(p_1) = 2 \quad \mathbf{M}_0(p_2) = 3$$
$$\mathbf{M}_0(p_3) = 0 \quad \mathbf{M}_0(p_4) = 0$$

となる．

3.2　デッドロック検出

あるマーキングにおいて，トランジション t のすべての入力プレイスがトークンを持つとき，トランジション t は発火可能になる．

ペトリネットにおけるデッドロックとは，すべてのトランジションが発火不可能である状態を言う．つまり，すべてのトランジション $t \in \mathbf{T}$ について，$M(p) = 0$ となるプレイス $p \in \bullet t$ が存在することを言う．

あるペトリネット PN でデッドロックが検出できるとは，PN がデッドロック状態になるような発火系列が存在することを言う．

3.3　Model Checking Contest

Model Checking Contest †1 は同期システムの検証ツールの評価を行うコンテストである．同期システムのモデルはすべてペトリネットとして与えられる．Model Checking Contest は以下の 4 つの部門からなる．

1. State Space generation
2. Reachability analysis
3. LTL analysis
4. CTL analysis

†1　Model Checking Contest `http://mcc.lip6.fr/`

State Space generation では，初期状態から到達可能な状態数やトランジションの発火回数，各プレイスのトークン数の最大値，各マーキングのトークン数の最大値を求める．Reachability analysis では初期マーキングから，与えられた条件を満たすような状態に到達できるかどうかを検証する．LTL (CTL) analysis では，LTL (CTL) の形式で与えられた性質をペトリネットが満たすかどうかを検証する．

また，これらの部門はさらに細かいいくつかのサブカテゴリーに分かれる．Reachability analysis 部門は以下のサブカテゴリーからなる．

- ReachabilityComputeBounds
- ReachabilityBounds
- ReachabilityCardinality
- ReachabilityFireability
- ReachabilityFireabilitySimple
- ReachabilityDeadlock

ReachabilityComputeBounds では，各プレイスのトークン数の上限に関する制約を満たすかどうかを調べる．ReachabilityBounds では，プレイスのトークン数が有界であるかどうかを調べる．ReachabilityCardinality では，プレイスのトークン数に関する制約が常に成り立つ，または成り立つ可能性があるかどうかを調べる．ReachabilityFireability では，トランジションの発火に関する制約が常に成り立つ，または成り立つ可能性があるかどうかを調べる．ReachabilityFireabilitySimple では，あるトランジションが常に発火する，または発火する可能性があるかどうかを調べる．本論文では，このうち，ReachabilityDeadlock での評価を行う．

4　提案手法

本節では，ペトリネットのデッドロック検出の提案手法について述べる．提案手法は，正規ペトリネット PN($\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{M}_0$) を入力とし，デッドロックが見つかれば処理を終了する半決定的手続きである．

提案手法では，有界モデル検査 [1] [2] [3] の手法を用いて，与えられたペトリネットのデッドロックを検出する．あるトークン数の上限 N と，遷移長 k の中で，正規ペトリネット PN が初期状態からデッドロック

状態に到達するという条件を制約充足問題として表し，それを順序符号化を用いて SAT 問題に変換した後，SAT ソルバーで求解する．SAT ソルバーが `SAT` を返した場合，与えられたペトリネットのトークン数の上限 N と，デッドロックを探索する遷移回数 k の中で，デッドロック状態に到達する発火系列があることを示す．`UNSAT` を返した場合，N, k の上限の中では，デッドロックを検出できないことを表す．このとき，N, k の値を増加させた SAT 問題を生成し，再び SAT ソルバーで求解する．一般的には N と k を共に増加させるべきだが，多くのペトリネットではトークン数が有限個となるため，後述する本論文での実装は k のみを増加させている．

4.1 正規ペトリネットの制約モデル

あるトークン数の上限 N と，遷移長 k の中で，正規ペトリネット PN が初期状態からデッドロック状態に到達することを表す制約モデルについて説明する．

まず，基本モデルについて説明し，次にトークン数の変化量に制限を加えた制限モデルと，多重発火を可能にした多重発火モデルについて説明する．

4.1.1 基本モデル

ステップ i でのペトリネットの状態はマーキングで表される．ステップ i でのプレイス p のトークン数を表す変数 m_p^i を導入する．

$$m_p^i \in \{0..N\} \qquad (p \in \mathbf{P}, 0 \leq i \leq k) \tag{1}$$

また，ステップ i でトランジション t が発火することを表す変数 f_t^i を導入する．

$$f_t^i \in \{0,1\} \qquad (t \in \mathbf{T}, 0 \leq i < k) \tag{2}$$

ステップ i で，プレイス p の出力トランジションのうち発火するトランジション数を表す変数 α_p^i と，入力トランジションのうち発火するトランジション数を表す β_p^i を導入する．

$$\begin{aligned} \alpha_p^i &\in \{0..|p\bullet|\} \qquad (p \in \mathbf{P}, 0 \leq i < k) \\ \beta_p^i &\in \{0..|\bullet p|\} \qquad (p \in \mathbf{P}, 0 \leq i < k) \end{aligned} \tag{3}$$

これらの元で制約モデルを考える．

プレイス p の発火する出力トランジション数 α_p^i は，$f_t^i \quad (t \in p\bullet)$ の和に等しい．また，プレイス p の発火する入力トランジション数 β_p^i も，$f_t^i \quad (t \in \bullet p)$ の和に等しい．このことを表す以下の制約を加える．

$$\begin{aligned} \alpha_p^i &= \sum_{t \in p\bullet} f_t^i \qquad (p \in \mathbf{P}, 0 \leq i < k) \\ \beta_p^i &= \sum_{t \in \bullet p} f_t^i \qquad (p \in \mathbf{P}, 0 \leq i < k) \end{aligned} \tag{4}$$

ステップ i から $i+1$ への遷移で p から減るトークン数は，ステップ i での p のトークン数以下である必要があるので，以下の制約を加える．

$$\alpha_p^i \leq m_p^i \qquad (p \in \mathbf{P}, 0 \leq i < k) \tag{5}$$

ステップ i からステップ $i+1$ への遷移があるとき，すべてのプレイス p に対して，以下が成り立つ．

1. ステップ i で発火した p の出力トランジション数は，p から減るトークン数と等しい．
2. ステップ i で発火した p の入力トランジション数は，p で増えるトークン数と等しい．

したがって，ペトリネット PN が初期マーキングから k 回状態遷移することは，以下の制約で表される．

$$\bigwedge_{p \in \mathbf{P}} (m_p^i = M_0(p)) \wedge \bigwedge_{i=0}^{k-1} \bigwedge_{p \in \mathbf{P}} (m_p^{i+1} = m_p^i - \alpha_p^i + \beta_p^i) \tag{6}$$

例として，図 1 のペトリネットに対応する制約モデルを示す．$N = 3, k = 1$ での制約モデルは以下のようになる．

$$\begin{aligned} &(m_{p_1}^0 = 2) \wedge (m_{p_2}^0 = 3) \wedge (m_{p_3}^0 = 0) \wedge (m_{p_4}^0 = 0) \\ &\wedge (\alpha_{p_1}^0 = f_{t_1}^0) \wedge (\alpha_{p_2}^0 = f_{t_1}^0 + f_{t_2}^0) \\ &\wedge (\alpha_{p_3}^0 = 0) \wedge (\alpha_{p_4}^0 = 0) \\ &\wedge (\beta_{p_1}^0 = 0) \wedge (\beta_{p_2}^0 = 0) \\ &\wedge (\beta_{p_3}^0 = f_{t_1}^0) \wedge (\beta_{p_4}^0 = f_{t_2}^0) \\ &\wedge (\alpha_{p_1}^0 \leq m_{p_1}^0) \wedge (\alpha_{p_2}^0 \leq m_{p_2}^0) \\ &\wedge (\alpha_{p_3}^0 \leq m_{p_3}^0) \wedge (\alpha_{p_4}^0 \leq m_{p_4}^0) \\ &\wedge (m_{p_1}^1 = m_{p_1}^0 - \alpha_{p_1}^0 + \beta_{p_1}^0) \\ &\wedge (m_{p_2}^1 = m_{p_2}^0 - \alpha_{p_2}^0 + \beta_{p_2}^0) \\ &\wedge (m_{p_3}^1 = m_{p_3}^0 - \alpha_{p_3}^0 + \beta_{p_3}^0) \\ &\wedge (m_{p_4}^1 = m_{p_4}^0 - \alpha_{p_4}^0 + \beta_{p_4}^0) \end{aligned} \tag{7}$$

図 1 のペトリネットの基本モデルでのデッドロック状態までの状態遷移は，図 2 の実線または点線で表した遷移が許される．この制約モデルでは，多

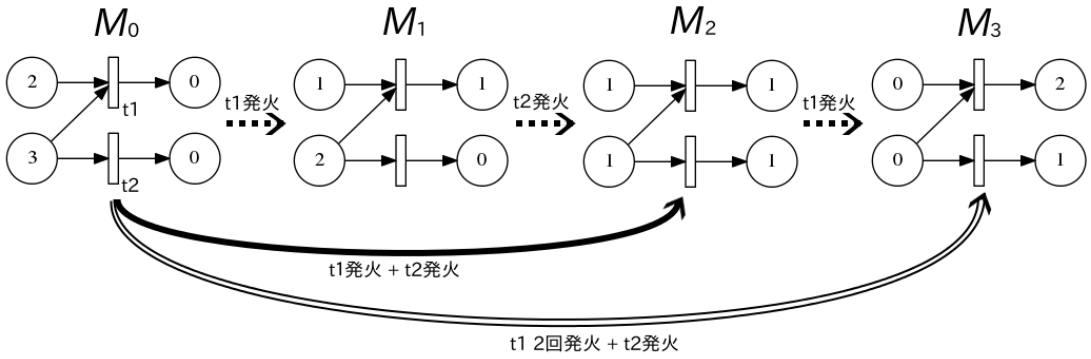

図 2　各制約モデルでの状態遷移

重発火ができないため，二重線で表した遷移 (トランジション t_1 が 2 回発火) はできない．すなわち，$M_o \to M_1 \to M_2 \to M_3$ あるいは $M_0 \to M_2 \to M_3$ の遷移が許される．したがって，$k = 2$ でデッドロックを検出できるが，$k = 1$ では検出できない．

4.1.2　制限モデル

制限モデルでは，各プレイスのトークン数の変化量を 1 個以内に制限する．制限モデルは，基本モデルに，以下の制約を加えることで表せる．

$$\alpha_p^i \leq 1 \qquad (p \in \mathbf{P}, 0 \leq i < k) \tag{8}$$

制限モデルは，基本モデルに比べて一度の遷移で発火できるトランジションが限られる．そのため，デッドロック状態に到達するには，基本モデルより長い遷移長 k が必要となる場合がある．例えば，図 1 のペトリネットの制限モデルでのデッドロック状態までの状態遷移は，図 2 の点線で表した遷移のみが許される．この制約モデルでは，各プレイスのトークン数の変化量が 1 以下に制限しているため，実線や二重線で表した遷移はできない．すなわち，$M_0 \to M_1 \to M_2 \to M_3$ の遷移のみが許される．したがって，$k = 3$ でデッドロックを検出できるが，$k = 2$ では検出できない．

4.1.3　多重発火モデル

多重発火モデルでは，トランジションの多重発火を許す．多重発火とは，一つのトランジションが同時に複数回発火することである． 多重発火モデルは，基本モデルの変数 f_t^i のドメインを $0 \leq f_t^i \leq N$ にすることで表せる．

$$f_t^i \in \{0..N\} \qquad (t \in \mathbf{T}, 0 \leq i < k) \tag{2'}$$

多重発火を許したことにより，デッドロック状態に到達するまでの遷移長 k は少なく済む可能性がある．図 1 のペトリネットの制限モデルでのデッドロック状態までの状態遷移は，図 2 の実線または点線または二重線で表した遷移が許される．この制約モデルでは，多重発火を許したことにより，t_1 が一度の遷移で 2 回発火することができる．すなわち，基本モデルでの遷移に加えて，$M_0 \to M_3$ の遷移が許される．したがって，$k = 1$ でデッドロックを検出できる．

各制約モデルを制約充足問題として表し，順序符号化を用いて SAT 問題に変換した時の節数を表 1 に示す． 表 1 での節数は，各ステップごとの節数である．同じ遷移長 k での節数では，制限モデルが最も少なくなり，多重発火モデルが最も多くなる．

表 1　各制約モデルでの SAT 問題の節数

制約モデル	節数
基本モデル	$O(\lvert\mathbf{A}\rvert \cdot N + \lvert\mathbf{P}\rvert \cdot N^2)$
制限モデル	$O(\lvert\mathbf{A}\rvert + \lvert\mathbf{P}\rvert \cdot N)$
多重発火モデル	$O(\lvert\mathbf{A}\rvert \cdot N^2 + \lvert\mathbf{P}\rvert \cdot N^2)$

4.2　デッドロック状態の制約モデル

i ステップ目でデッドロック状態となる条件は以下

表 2　各制約モデルに対応する式

制約モデル	対応する制約式
基本モデル	(1)-(6), (10)
制限モデル	(1)-(6), (8), (10)
多重発火モデル	(1), (3)-(6), (2′), (10)

のように表せる．

$$\bigwedge_{t\in T}\bigvee_{p\in\bullet t}(m_p^i=0) \tag{9}$$

この条件は，すべてのトランジションが発火不可能，すなわち，トークンを持たない入力プレイスが存在することを表している．

この条件を用い，k 回以内の遷移でデッドロック状態に到達する発火系列があることを表す制約を与える．i 回以内の遷移でデッドロック状態に到達する発火系列があることを表す命題変数 D_i を導入する．このとき，k 回以内の遷移でデッドロック状態に到達する制約は以下のように表すことができる．

$$\begin{aligned}&(D_0\rightarrow\bigwedge_{t\in T}\bigvee_{p\in\bullet t}(m_p^0=0))\\&\wedge\bigwedge_{i=1}^{k}\{D_i\rightarrow(D_{i-1}\vee\bigwedge_{t\in T}\bigvee_{p\in\bullet t}(m_p^i=0))\}\\&\wedge D_k\end{aligned} \tag{10}$$

4.3　デッドロックの探索手法

有界モデル検査の手法を用いて，与えられたペトリネットのデッドロックを検出する．Algorithm 1 にデッドロック検出の手順を示す．$\phi_k(\mathrm{PN},N)$ は，ペトリネット PN が各プレイスのトークン数上限 N，遷移長 k のなかでデッドロック状態となることを表す制約式である ($\phi_k(\mathrm{PN},N)$ の詳細は表 2 を参照)．

Algorithm 1 デッドロック検出手続き

入力: ペトリネット PN，トークン数の上限 N

```
1: k = 1
2: loop
3:     ψ = φ_k(PN, N) を順序符号化した CNF 式
4:     if ψ が充足可能 then
5:         break /* デッドロック検出 */
6:     end if
7:     k = next(k)
8: end loop
```

まず，$k=1$ と与えられた PN と N に対する制約式 $\phi_k(\mathrm{PN},N)$ を生成し，それを順序符号化を用いて SAT 問題 ψ に変換する．その ψ を SAT ソルバーに与え，充足可能かどうかを判定する．もし，ψ が充足可能ならば，k，N のなかでデッドロックが検出できたとして，この手続きを終了する．もし，ψ が充足不能ならば，k，N のなかではデッドロックを検出できなかったので，k の値を増加させ，再度同様の処理を行う．ここで，$next(k)$ は次にデッドロックを探索すべき遷移長を返す関数である．なお，実装では N の値は初期状態のマーキングにおける最大のトークン数とした．

$next(k)$ の定義について，以下の 3 通りを考えた．

1. $next(k)=k+1$
2. $next(k)=1.5k$
3. $next(k)=2k$

予備実験として，この 3 通りについて実験をしたところ，$next(k)=2k$ のときが最も良い結果となった．以下では，$next(k)=2k$ を用いる．

5　性能評価

提案手法の有効性を検証するため，Model Checking Contest 2015 のデッドロック検出部門の問題 267 問のうち，正規ペトリネットでデッドロックがあることが分かっている問題 83 問を用いて性能評価を行った．ベンチマークは Ubuntu 14.04 (Xeon 3.5GHz, メモリ 8GB) 上で計測した．

5.1 提案する制約モデル 3 つの比較

提案した 3 つの制約モデル (基本モデル，制限モデル，多重発火モデル) がデッドロックを検出できた問題数を表 3，4 に示す．表 3 では，各制約モデルでデッドロックを検出できた問題数と，CPU 時間の平均を N の値ごとに示す．表 3 での CPU 時間の平均は，時間内に解けなかった問題の CPU 時間を 600 秒として計算した平均である．表 4 では，各制約モデルでデッドロックを検出できた問題数と，デッドロック検出に必要な遷移長 k の平均を N の値ごとに示す．なお， 表 4 の k の平均値は，デッドロックを検出できた問題のみに対する平均値である．

まず，基本モデルと制限モデルについて比較する．表 3 を見ると，$N = 11 \sim 20$，$N = 21 \sim 49$ では基本モデルがデッドロックを検出できなかった問題を 1 問ずつ制限モデルで検出している．この理由についてはまだ考察が必要であるが，デッドロック検出に必要な遷移長が同程度の時は制限モデルの方がよい性能を示す可能性がある．

次に，多重発火モデルと他の制約モデルについて比較する．表 3 を見ると，N が大きくなるにつれて，多重発火モデルが他の制約モデルより短い CPU 時間でデッドロックを検出できている．特に，$N \geq 50$ の問題では，多重発火モデルのみがデッドロックを検出できている．表 4 を見ると，基本モデルや制限モデルでは，N が大きくなるにつれ，デッドロック検出に必要な遷移長 k が大きくなっているが，多重発火モデルでは N が大きくなっても比較的小さい k でデッドロックを検出できており，それが CPU 時間の短縮につながっている．多重発火モデルは各ステップでの節数は最も多くなるが，k が短いことで `UNSAT` の判定回数が少なくて済むことが理由と考えられる．

本実験では多重発火モデルが最もよい性能を示したが，多重発火が起こりにくいようなペトリネットの場合，制限モデルの方がよい性能を示す可能性がある．

5.2 既存手法との比較

提案手法 (多重発火モデル) と既存手法を比較する．比較に用いる既存手法としては，SAT 型検証手法である SAT-based Verification of Safe Petri Nets (Ogata+) [12] の手法と Cunf [14]，そして，Model Checking Contest 2015 デッドロック検出部門で優勝したツール LoLA2 [11] を用いた．提案手法と既存手法で，デッドロックを検出できた問題数を比較した結果を表 5 に示す．提案手法は LoLA2 がデッドロックを検出できなかった問題 2 問 (Angiogenesis-PT-50, Philosophers-PT-010000) のデッドロック検出に成功した．このうち Angiogenesis-PT-50 は Model Checking Contest 2015 に参加したいずれのツールもデッドロックを検出できなかった問題である．表 6 は Angiogenesis-PT での提案手法と LoLA2 の結果 (CPU 時間と使用メモリサイズ) を示す．

表 5　既存手法との比較

	提案	Ogata+	Cunf	LoLA2
安全 (52 問)	40	44	27	47
安全でない (31 問)	29	—	—	30
合計 (83 問)	69	44	27	77

Angiogenesis-PT-50 は初期状態から到達可能な状態数が非常に多い問題である．LoLA2 では被覆グラフを用いてデッドロックに到達するかどうかを調べているが，Angiogenesis-PT-50 では状態数多いためメモリーオーバーを起こしている．一方，SAT 型検証手法では，他の方法に比べ状態空間爆発が起こりにくいという特徴があり，Angiogenesis-PT-50 のデッドロック検出に成功している．

6 まとめと今後の課題

本論文では，SAT 技術を用いて，正規ペトリネットのデッドロック検出を行った．正規ペトリネットの 3 つの制約モデル，基本モデル，制限モデル，多重発火モデルを提案し，実験により，それぞれの特徴を確かめた．制限モデルは，他 2 つの制約モデルに比べ，少ない節数で表現できるため，大きなサイズのペトリネットに対して有効であることが確かめられた．一方，多重発火モデルは，他 2 つの制約モデルに比べ，一度の遷移でプレイス中のトークン数を大きく変化できるため，トークン数の上限が大きい問題に対し，より少ない遷移回数でデッドロックを検出できることが確かめられた．

表 3　各制約モデルで検出できた問題数と CPU 時間の平均値

		検出できた問題数			CPU 時間の平均 (秒)		
N		基本モデル	制限モデル	多重発火モデル	基本モデル	制限モデル	多重発火モデル
1	(54)	41	41	41	62.99	55.50	63.35
2〜5	(8)	8	8	8	6.15	8.38	5.87
6〜10	(6)	6	6	6	19.24	60.75	19.98
11〜20	(6)	5	6	6	119.32	69.89	15.20
21〜49	(4)	3	4	4	154.76	30.00	15.86
50〜500	(5)	0	0	4	—	—	153.59
合計	(83)	63	65	69	93.95	81.83	51.21

表 4　各制約モデルで検出できた問題数と k の平均値

		検出できた問題数			k の平均		
N		基本モデル	制限モデル	多重発火モデル	基本モデル	制限モデル	多重発火モデル
1	(54)	41	41	41	11.7	11.7	11.7
2〜5	(8)	8	8	8	8.3	10.0	8.3
6〜10	(6)	6	6	6	13.0	20.0	7.3
11〜20	(6)	5	6	6	14.0	24.0	9.0
21〜49	(4)	3	4	4	4.7	26.0	7.5
50〜500	(5)	0	0	4	—	—	16.0
合計	(83)	63	65	69	11.17	14.25	10.67

表 6　Angiogenesis-PT での比較

		提案手法		LoLA2	
instance	N	CPU 時間	メモリ	CPU 時間	メモリ
Angiogenesis-PT-10	10	11.37	4979764	1.90	176952
Angiogenesis-PT-15	15	18.94	4986748	15.86	1020700
Angiogenesis-PT-20	20	15.38	5094012	165.39	7925784
Angiogenesis-PT-25	25	22.21	5092920	66.43	3703676
Angiogenesis-PT-50	50	178.92	5115492	M.O.	M.O.

既存手法との比較では，SAT 型検証手法である SAT-based Verification of Safe Petri Nets の手法と Cunf，そして，Model Checking Contest 2015 デッドロック検出部門で優勝したツール LoLA2 と比較した．デッドロックを検出できた問題数では LoLA2 が最も良い結果となったが，既存手法がいずれもデッドロックを検出できなかった問題でのデッドロック検出に成功した．

今後の研究課題としては，制限モデルと多重発火モデルを組み合わせたデッドロックの探索手法が挙げられる．また，P/T ネット (アークの多重度が整数値) のペトリネットでのデッドロック検出や，デッドロック検出よりも一般的な性質の検証である，LTL 式で表された性質の検証が挙げられる．

参 考 文 献

[1] 番原睦則, 田村直之: SAT によるシステム検証, 人工知能学会誌, Vol. 25, No. 1(2010), pp. 122–129.

[2] Biere, A.: Bounded Model Checking, *Handbook of Satisfiability*, IOS Press, 2009, pp. 457–481.
[3] Biere, A., Cimatti, A., Clarke, E. M., and Zhu, Y.: Symbolic Model Checking without BDDs, *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 1999), LNCS 1579*, 1999, pp. 193–207.
[4] de Kleer, J.: A Comparison of ATMS and CSP Techniques, *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI 1989)*, 1989, pp. 290–296.
[5] Gelder, A. V.: Another Look at Graph Coloring via Propositional Satisfiability, *Discrete Applied Mathematics*, Vol. 156, No. 2(2008), pp. 230–243.
[6] Gent, I. P.: Arc Consistency in SAT, *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2002)*, 2002, pp. 121–125.
[7] 井上克巳, 田村直之: SAT ソルバーの基礎, 人工知能学会誌, Vol. 25(2010), pp. 57–67.
[8] Iwama, K. and Miyazaki, S.: SAT-Variable Complexity of Hard Combinatorial Problems, *Proceedings of the IFIP 13th World Computer Congress*, 1994, pp. 253–258.
[9] Kasif, S.: On the Parallel Complexity of Discrete Relaxation in Constraint Satisfaction Networks, *Artificial Intelligence*, Vol. 45, No. 3(1990), pp. 275–286.
[10] 鍋島英知, 宋剛秀: 高速 SAT ソルバーの原理, 人工知能学会誌, Vol. 25, No. 1(2010), pp. 68–76.
[11] Niewiadomski, A. and Wolf, K.: LoLA as Abstract Planning Engine of PlanICS, *Proceedings of the International Workshop on Petri Nets and Software Engineering, Tunis, Tunisia, June 23-24, 2014.*, 2014, pp. 349–350.
[12] Ogata, S., Tsuchiya, T., and Kikuno, T.: SAT-Based Verification of Safe Petri Nets, *Automated Technology for Verification and Analysis: Second International Conference, ATVA 2004, Taipei, Taiwan, ROC, October 31-November 3, 2004. Proceedings*, 2004, pp. 79–92.
[13] 奥川峻史: ペトリネットの基礎, 共立出版, 1995.
[14] Rodríguez, C. and Schwoon, S.: Cunf: A Tool for Unfolding and Verifying Petri Nets with Read Arcs, *Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*, 2013, pp. 492–495.
[15] Soh, T., Inoue, K., Tamura, N., Banbara, M., and Nabeshima, H.: A SAT-based Method for Solving the Two-dimensional Strip Packing Problem, *Fundam. Inform.*, Vol. 102, No. 3-4(2010), pp. 467–487.
[16] 田村直之, 多賀明子, 番原睦則, 宋剛秀, 鍋島英知, 井上克巳: ショップ・スケジューリング問題の SAT 変換による解法, スケジューリング・シンポジウム *2007* 講演論文集, 2007, pp. 97–102.
[17] Tamura, N., Taga, A., Kitagawa, S., and Banbara, M.: Compiling Finite Linear CSP into SAT, *Constraints*, Vol. 14, No. 2(2009), pp. 254–272.
[18] Walsh, T.: SAT v CSP, *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP 2000)*, 2000, pp. 441–456.