SAT技術を用いた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
井上，克巳

(Citation)
情報処理学会論文誌,59(9):1749-1760

(Issue Date)
2018-09-15

(Resource Type)
journal article

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/90006798

情報処理学会論文誌 Vol.60 No.1 1–11 (Jan. 2019)

SAT技術を用いたペトリネットのデッドロック検出手法の提案

寸田 智也[1] 宋 剛秀[2] 番原 睦則[2] 田村 直之[2] 井上 克巳[3]

受付日 2018年1月1日, 採録日 2018年1月1日

概要：本論文では，トークン数が整数値である一般のペトリネットを対象とし，そのデッドロック検出についてSAT技術を用いた手法を提案する．提案手法では，非負整数値であるトークン数を表現するために順序符号化を採用することで，既存のSAT型手法では対応できなかった一般のペトリネットでのデッドロック検出を実現した．また，既存SAT型手法で採用されていたモデル（連続発火モデルと呼ぶ）よりも短いステップ長でデッドロック検出が可能となる多重発火モデルを提案し，性能向上を実現した．評価実験では，Model Checking Contest 2017のベンチマーク問題を用いて，連続発火モデルと多重発火モデルを比較した．ほぼすべての問題で多重発火モデルのほうが優れていたが，特に初期マーキングのトークン数が比較的多い問題に対する効果が大きかった．また，Model Checking Contest 2017 デッドロック検出部門での優勝ソルバー LoLA および準優勝ソルバー Tapaal との比較を行った．提案手法は，LoLA および Tapaal を含めたすべてのソルバーがデッドロック検出に失敗した問題での検出に成功し，提案手法の有効性が確認できた．

キーワード：ペトリネット, デッドロック検出, SAT技術, 有界モデル検査

Proposal of SAT-Based Method to Detect Deadlock of Petri Nets

Tomoya Sunda[1] Takehide Soh[2] Mutsunori Banbara[2] Naoyuki Tamura[2] Katsumi Inoue[3]

Received: January 1, 2018, Accepted: January 1, 2018

Abstract: In this paper, we proposed a SAT-based method to detect deadlock of general Petri nets in which more than one tokens are allowed for each place. In our approach, the transition relation of a Petri net is represented as constraints on integers and they are translated into SAT by order encoding, so that the deadlock of general Petri nets can be detected by a SAT solver, while existing SAT-based methods cannot be applied for them. Furthermore, in order to improve performance, we introduced multiple firing model, which can detect deadlock with shorter steps than the model used in an existing SAT-based method called successive firing model. We evaluated the successive firing model and the multiple firing model through a benchmark set of Model Checking Contest 2017. The multiple firing model showed better performance for almost all instances, and was especially effective for the instances in which there are many tokens at the initial marking. Through the comparison with the winner tool LoLA and the second place tool Tapaal of the contest, we confirmed the effectiveness of the proposed method with some instances for which all tools including LoLA and Tapaal failed to detect deadlock.

Keywords: Petri nets, Deadlock detection, SAT technologies, Bounded model checking

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

[1] 神戸大学大学院システム情報学研究科
Graduate School of System Informatics, Kobe University
[2] 神戸大学情報基盤センター
Information Science and Technology Center, Kobe University
[3] 国立情報学研究所情報学プリンシプル研究系
Principles of Informatics Research Division, National Institute of Informatics

的，非決定的な動作のモデルであり，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 [14]．本論文で対象とするデッドロックの検証を含め，その検証は重要な研究課題となっている．2011 年からペトリネットの検証を行うソフトウェアに関する国際コンテスト Model Checking Contest[*1]が毎年開催されている．

命題論理の充足可能性判定（Satisfiability testing; **SAT**）は，与えられた命題論理式が真になる値割当てが存在するかどうかを判定する問題である [1], [7]．与えられた SAT 問題を解くプログラムである **SAT ソルバー**の性能が近年飛躍的に向上し [10], [11], [12]，システム検証 [2], [3], [4] を含め多くの分野への応用が広がっている．ペトリネットのデッドロック検証への応用も存在するが [13], [15]，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，これらで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**のみが対象となっていた．

そこで本論文では，SAT ソルバーを用いて，トークン数が非負整数値である**一般のペトリネットでのデッドロック検出**を行う方法を提案する．

しかし，ペトリネットのモデルを単純に SAT 問題に変換（**SAT 符号化**という）した場合，デッドロック検出までのステップ長が長くなると，SAT 符号化の結果である命題論理式のサイズが大きくなりすぎ，SAT ソルバーでの求解が行えないという問題点が生じる．そこで，本論文では，デッドロック検出までのステップ長を短くするための工夫として，複数回トランジションが発火する遷移を許すモデル（**多重発火モデル**と呼ぶ）を導入した．すなわち，本論文で提案する手法の特徴は以下の通りである．

(1) SAT ソルバーによる一般のペトリネットでのデッドロック検出の実現
非負整数値であるトークン数を表現するために**順序符号化** [19], [20] を用い，ペトリネットの可達性を命題論理式の充足可能性として表現する．これにより，既存の SAT 型手法である [13], [15] では対応できなかった一般のペトリネットでのデッドロック検出が可能となった．

(2) SAT ソルバーによる効率良いデッドロック検出を可能とするための多重発火モデルの導入
既存の SAT 型手法である [13] で採用されていたモデル（連続発火モデルと呼ぶ）よりも短いステップ長でデッドロック検出が可能となるモデルを提案した．これにより全般に性能が向上したが，特にトークン数が比較的多い問題に対する効果が大きいことがわかった．

以下，2 節でペトリネットとそのデッドロックを定義し，3 節で本論文で利用する SAT 技術について説明する．4 節では提案する多重発火モデルを導入する．まず，安全ペトリネットを対象とした既存 SAT 型手法を一般のペトリネットに拡張した連続発火モデルを述べ，さらに多重発火モデルに拡張する．次に 5 節で多重発火モデルの SAT 符号化方法について説明し，6 節で SAT 型手法の実装方法を述べる．7 節では，Model Checking Contest 2017 のベンチマーク問題を用い，実装した 2 種類の手法および既存手法の比較実験を行い，提案手法について評価する．最後に，8 節で結論を述べる．

[*1] http://mcc.lip6.fr

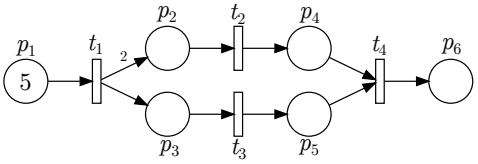


図 1 ペトリネットの例
Fig. 1 Example of a Petri net

2. ペトリネットとデッドロック

$\mathbb{N}$ で 0 以上の整数の集合を表す．ペトリネットは，組 $(\mathbf{P}, \mathbf{T}, F, M_0)$ で与えられる [14]．ここで，$\mathbf{P}$ は**プレイス**の有限集合，$\mathbf{T}$ は**トランジション**の有限集合，$F : (\mathbf{P} \times \mathbf{T}) \cup (\mathbf{T} \times \mathbf{P}) \to \mathbb{N}$ は**アーク**の多重度（アークがない場合は 0）を与える関数，$M_0 : \mathbf{P} \to \mathbb{N}$ は初期**マーキング**である．ペトリネットの状態は，各プレイスでの**トークン**の個数で定まり，マーキング $M : \mathbf{P} \to \mathbb{N}$ により表される．また，$\bullet p$ でプレイス p の入力トランジションの集合，$p\bullet$ で出力トランジションの集合，$\bullet t$ でトランジション t の入力プレイスの集合，$t\bullet$ で出力プレイスの集合を表す．**図 1** にペトリネットの例を示す．ただし，アークの多重度が 1 の場合，その記載を省略している．

ペトリネットはトランジションの**発火**により次のマーキングへと遷移する．トランジション t はすべての入力プレイスが対応するアークの多重度以上のトークンを持つときのみ発火可能となる．t の発火でマーキング M から M' に遷移することを $M \xrightarrow{t} M'$ で表す．$M \xrightarrow{t} M'$ は以下のように定義できる．

$$M \xrightarrow{t} M' \overset{\text{def}}{\Longleftrightarrow} \forall p \in \mathbf{P}\ (M(p) \geq F(p,t) \wedge M'(p) = M(p) + F(t,p) - F(p,t)) \quad (1)$$

また，$M \to M'$ を以下のように定義し，これを標準遷移あるいは $\to$ 遷移と呼ぶ．

$$M \to M' \overset{\text{def}}{\Longleftrightarrow} \exists t \in \mathbf{T}\ (M \xrightarrow{t} M') \quad (2)$$

初期マーキング M からマーキング M' へ遷移可能であるようなトランジションの発火系列が存在するとき，マーキング M' は M から**可達**であるといい，$M \to^* M'$ で表す．このとき，初期マーキングでデッドロックしている場

合もありうるので，$M = M'$ の場合も含む．すなわち，$\rightarrow^*$ は $\rightarrow$ の反射的推移的閉包である．

デッドロックとは，すべてのトランジションが発火可能でないようなマーキングを意味する．デッドロックが初期マーキングから可達であるとき，そのペトリネットはデッドロックを持つという．ペトリネットがデッドロックを持つことは以下のように表せる．

$$\exists M \left(M_0 \rightarrow^* M \wedge \forall t \in \mathbf{T} \, \exists p \in \bullet t \left(M(p) < F(p,t) \right) \right) \quad (3)$$

3. 利用する SAT 技術

以下では，本論文で利用する SAT 技術について述べる．

3.1 SAT ソルバーとインクリメンタル SAT 解法

SAT ソルバーに与える命題論理式は，**連言標準形**（Conjunctive Normal Form; CNF）で記述する必要がある [1], [7]．連言標準形の式（**CNF 式**）は，複数の**節**の連言（AND）であり，各節は複数の**リテラル**の選言（OR）である．各リテラルは，命題変数あるいは命題変数の否定である．

SAT ソルバーは，与えられた CNF 式が充足可能（SAT）か充足不能（UNSAT）を判定し，充足可能ならその値割当てを出力する．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**を利用して探索の枝刈りを行う 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 [11], [12]．

また，本論文で使用する GlueMiniSat [10] を含め多くの SAT ソルバーは**インクリメンタル SAT 解法**と呼ばれる方法に対応している [5]．インクリメンタル SAT 解法では，SAT ソルバーを起動したまま，関連した複数の SAT 問題を連続して解くことが可能になる．したがって，SAT ソルバーが獲得した学習節が保持され，効率良い解探索が実現できる．本論文では，[16] で提案された iSAT Library[*2] を用い，インクリメンタル SAT 解法を導入する．

3.2 SAT 符号化

与えられた制約をすべて満たすことができるような変数の値の割当てが存在するかどうかを判定する問題を**制約充足問題**（Constraint Satisfaction Problem; CSP）という．特に，制約中に現れる変数が有限範囲の整数値だけを取る CSP を有限領域 CSP という．後述のように，ペトリネットのステップ長およびトークン数などの上限が与えられた場合，その範囲内でデッドロックが生じるかどうかは，有限領域 CSP の充足可能性の判定問題として定式化できる．

有限領域 CSP については，与えられた問題を SAT 問題に符号化する手法が活発に研究開発されている [21]．特に，加減乗除等の演算が含まれている制約に対しては，順序符号化 [19], [20] が有効であることが知られている [9]．順序符号化では，整数変数 x とその取り得る値 a に対し，$x \leq a$ を意味する 1 つの命題変数を割り当てる．取り得る値が d 個であるような整数変数を符号化した結果の節数は $O(d)$ であり，2 変数を含む制約（たとえば $x \geq y$）に対する節数は $O(d)$，3 変数を含む制約（たとえば $z \geq x + y$）に対する節数は $O(d^2)$ となる．順序符号化では整数変数の取り得る範囲の推論が SAT ソルバーの単位伝播で実現されており，他の SAT 符号化より高速な実行が可能である [21]．実際に，順序符号化を採用した **SAT 型制約ソルバー**である Sugar[*3] [20] 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，多くの問題に対し有望な手法であることが示されている．

3.3 有界モデル検査

SAT ソルバーを用いて動的なシステムの検証を行う手法として，**有界モデル検査**（Bounded Model Checking; BMC）[2], [3], [4] がある．この方法では，システムにおける状態遷移の**ステップ長**の上限 k を与え，時刻 0 から時刻 k までの状態遷移の系列 $M_0 \rightarrow M_1 \rightarrow \cdots \rightarrow M_k$ において，ある性質 P を満たす状態 M_i が存在するかどうかを SAT ソルバーを用いて判定する．有界モデル検査は，特にハードウェアの検証において，それまで用いられていた二分決定木（Binary Decision Diagram; BDD）による方法より，大規模な問題に対処可能な手法として有効性が示されている [4]．

今，状態 M_0 が初期状態であることを式 $I(M_0)$ で，状態 M_i から M_i への遷移関係を式 $T(M_i, M_{i+1})$ で，状態 M_i が性質 P を持たすことを式 $P(M_i)$ で表すと，上記の系列の存在は

$$I(M_0) \wedge \bigwedge_{i=0}^{k-1} T(M_i, M_{i+1}) \wedge \bigvee_{i=0}^{k} P(M_i)$$

という式の充足可能性判定問題と同等になる．すなわち，上記を CNF 式として表したものが SAT であれば時刻 k までに性質 P を満たす状態が存在し，UNSAT であれば存在しない．

有界モデル検査では，ステップ長の上限 k の値を徐々に増やしながら判定することが通常である．この時，前述のインクリメタル SAT 解法と組合せることで，性能が向上することが知られている [5]．本論文でも，前述の GlueMiniSat および iSAT Library を用い，インクリメンタル SAT 解法を導入した有界モデル検査を実現する．

4. 多重発火モデルの提案

本節では，安全ペトリネットを対象とした既存 SAT 型

[*2] http://bach.istc.kobe-u.ac.jp/iSATLib/

[*3] http://bach.istc.kobe-u.ac.jp/sugar/

手法を一般のペトリネットに拡張した連続発火モデルを述べ，さらに多重発火モデルに拡張する．

4.1 連続発火モデル

ペトリネットのデッドロック検出に関し，SAT 技術を用いた既存手法として Ogata らの方法 [13] がある．各プレイスのトークン数が 1 以下である安全ペトリネットだけを対象としているが，そのアイデアは一般のペトリネットに適用できる．ここでは，それを**連続発火モデル**と呼ぶ．

連続発火モデルでは，トランジションの順序を定める必要がある．すなわち，$l = |\mathbf{T}|$ とし，トランジションを適当な順序で並べたものを $t_1, t_2, \ldots, t_l$ とする．この順序は任意で良いが，例えば，6.1 節で述べるようにペトリネット内のプレイスを深さ優先で辿った順序が考えられる．

この時，$M \xrightarrow{t} M'$ を以下のように定義する．

$$M \xrightarrow{t} M' \stackrel{\text{def}}{\Longleftrightarrow} (M \xrightarrow{t} M') \vee (M = M') \qquad (4)$$

さらに，$M \to_c M'$ をマーキング M から M' への連続発火遷移あるいは $\to_c$ 遷移と呼び，以下のように定義する．

$$M \to_c M' \stackrel{\text{def}}{\Longleftrightarrow} M = M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \cdots \xrightarrow{t_l} M_l = M' \qquad (5)$$

すなわち，$\to_c$ 遷移では，その中で複数の別々のトランジションが連続して発火することが許されている．

可能な $\to_C$ 遷移の例を**図 2** に示す．トランジションの順序は t_1, t_2 としている．まず，左端のマーキング $(p_1, p_2, p_3, p_4) = (1, 0, 0, 0)$ で t_1 は発火可能だから，$\xrightarrow{t_1}$ により $(0, 1, 1, 0)$ または $(1, 0, 0, 0)$ に遷移できる．次に $(0, 1, 1, 0)$ で t_2 は発火可能だから $(0, 1, 0, 1)$ または $(0, 1, 1, 0)$ に遷移する．一方で $(1, 0, 0, 0)$ では t_2 は発火できないから，$\xrightarrow{t_2}$ により $(1, 0, 0, 0)$ にのみ遷移する．すなわち，マーキング $(1, 0, 0, 0)$ から可能な $\to_c$ 遷移先は，右端に示す $(0, 1, 0, 1)$, $(0, 1, 1, 0)$, $(1, 0, 0, 0)$ の 3 通りになる．

$\to_c{}^*$ を $\to_c$ の反射的推移的閉包とする．明らかに $M \to^* M'$ と $M \to_c{}^* M'$ は同値であり，可達性は変わらない．また，M から k ステップの $\to$ 遷移で M' へ到達できるとき，k ステップ以下の $\to_c$ 遷移で到達できる．

4.2 多重発火モデル

同じトランジションが，連続して複数回発火することを多重発火と呼ぶ．連続発火モデルの遷移 $M \xrightarrow{t} M'$ において，t の多重発火を認めた**多重発火モデル**を導入する．多重発火モデルでは，連続発火モデルよりさらに短いステップ長でデッドロックに到達できる．なお，以下では連続発火モデルと同様にトランジションの順序を $t_1, t_2, \ldots, t_l$ のように定める．

マーキング M から t が n $(n \geq 0)$ 回発火して M' に遷移することを $M \xrightarrow{t^n} M'$ で表し，以下のように定義する．

$$M \xrightarrow{t^n} M' \stackrel{\text{def}}{\Longleftrightarrow} \forall p \in \mathbf{P}\ (M(p) \geq nF(p,t) \wedge M'(p) = M(p) + n(F(t,p) - F(p,t))) \qquad (6)$$

また，$M \to_m M'$ をマーキング M から M' への多重発火遷移あるいは $\to_m$ 遷移と呼び，以下のように定義する．

$$M \to_m M' \stackrel{\text{def}}{\Longleftrightarrow} \exists n_1, n_2, \ldots, n_l \in \mathbb{N}\ (M = M_0 \xrightarrow{t_1^{n_1}} M_1 \xrightarrow{t_2^{n_2}} \cdots \xrightarrow{t_l^{n_l}} M_l = M') \qquad (7)$$

$n_j = 0$ の場合には $M_{j-1} = M_j$ となることに注意すると，$\to_m$ 遷移中の各 t_j の多重発火回数 n_j を 1 以下に制限した場合，連続発火での $\to_c$ 遷移に相当することがわかる．

$\to_m{}^*$ を $\to_m$ の反射的推移的閉包とする．明らかに $M \to_m{}^* M'$ は，$M \to^* M'$ および $M \to_c{}^* M'$ と同値であり，可達性は変わらない．また，M から k ステップの $\to_c$ 遷移で M' へ到達できるとき，k ステップ以下の $\to_m$ 遷移で到達できる．

図 1 に示したペトリネットについて，連続発火モデルと多重発火モデルの各々に対し，デッドロック検出に必要なステップ長を示す．なお，トランジションの順序は t_1, t_2, t_3, t_4 の順とする．また，初期マーキングは $(p_1, p_2, p_3, p_4, p_5, p_6) = (5, 0, 0, 0, 0, 0)$ である．

連続発火モデルの場合，最初の $\to_c$ 遷移で t_1, t_2, t_3, t_4 が発火することでマーキング $(4, 1, 0, 0, 0, 1)$ に遷移できる．その後，$(3, 2, 0, 0, 0, 2) \to_c (2, 3, 0, 0, 0, 3) \to_c \cdots \to_c (0, 0, 0, 5, 0, 5)$ と遷移した場合の計 9 ステップが最短である．

一方，多重発火モデルの場合，最初の $\to_m$ 遷移で $t_1^5, t_2^{10}, t_3^5, t_4^5$ の発火により，1 ステップでデッドロック $(0, 0, 0, 5, 0, 5)$ に遷移できる．なお，トランジションの順序を t_1, t_2, t_4, t_3 とした場合には，最短で 2 ステップが必要となる．このように，トランジションの順序に依存して変化するが，連続発火モデルよりかなり短いステップ長でデッドロックに到達することが可能になる．

5. 多重発火モデルの SAT 符号化

Sugar などの既存の SAT 型制約ソルバーの利用を前提とすると，多重発火モデルの SAT 符号化には，$\to_m$ 遷移やデッドロックの条件などを有限領域 CSP の制約として記述すれば十分であり，その後の SAT 符号化自体は SAT 型制約ソルバーに行わせることが可能である．そのため，ここでは対応する制約の説明を中心に行う．

5.1 整数変数の導入と初期マーキングの制約

多重発火モデルでの状態遷移の系列 $M_0 \to_m M_1 \to_m \cdots \to_m M_k$ において，各 $\to_m$ は，さらにトランジション t_1,

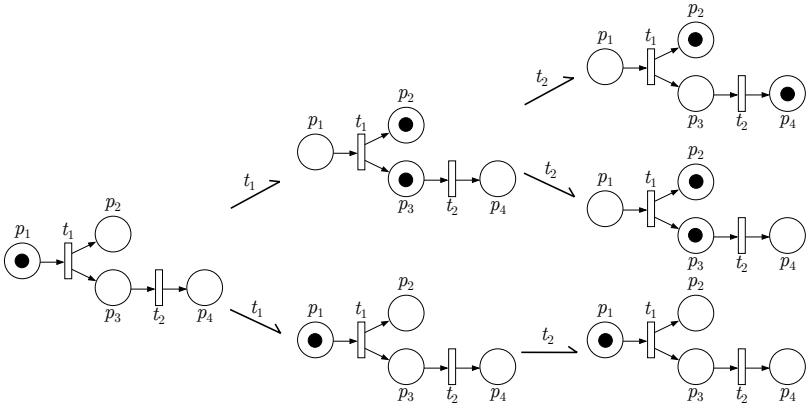


図 2 可能な $\rightarrow_c$ 遷移の例
Fig. 2 Example of possible $\rightarrow_c$ transition

$t_2, \ldots, t_l$ に対する遷移 $\xrightarrow{t_j^{n_j}}$ の列である（$n_j \geq 0$）．すなわち遷移 $M_i \rightarrow_m M_{i+1}$ は，$M_i = M_{i,0}$ および $M_{i+1} = M_{i,l}$ とすると，遷移の系列

$$M_{i,0} \xrightarrow{t_1^{n_{i,1}}} M_{i,1} \xrightarrow{t_2^{n_{i,2}}} \cdots \xrightarrow{t_l^{n_{i,l}}} M_{i,l}$$

として表現できる．

そこで，制約で使用する整数変数を以下のように導入する．

$$\begin{aligned} &m_p^{il+j} \quad (p \in \mathbf{P},\ 0 \leq i < k,\ 0 \leq j < l) \\ &f_t^i \quad (t \in \mathbf{T},\ 0 \leq i < k) \end{aligned} \tag{8}$$

ここで，変数 m_p^{il+j} は上記の $M_{i,j}(p)$ に対応する．変数 $f_{t_j}^i$ は上記の $n_{i,j}$ に対応し，M_i から M_{i+1} への $\rightarrow_m$ 遷移中で t が多重発火する回数を表している．どちらも 0 以上の値を取るが，その上限の定め方については 6.3 節で述べる．なお，$f_{t_j}^i$ の上限を 1 に制限した場合，連続発火モデルが対応する．

この時，M_0 すなわち $M_{0,1}$ が初期マーキングであるという条件 I は，以下のような制約で記述できる．

$$I \overset{\text{def}}{\Longleftrightarrow} \bigwedge_{p \in \mathbf{P}} m_p^0 = M_0(p) \tag{9}$$

5.2 $\rightarrow_m$ 遷移の制約

遷移 $M_{i,j-1} \xrightarrow{t_j^{n_{i,j}}} M_{i,j}$ は以下のような制約で表すことができる．

$$\begin{aligned} &m_p^{il+j-1} \geq F(p,t_j) f_{t_j}^i \quad (p \in {\bullet}t_j) \\ &m_p^{il+j} = m_p^{il+j-1} - F(p,t_j) f_{t_j}^i \quad (p \in {\bullet}t_j \backslash t_j{\bullet}) \\ &m_p^{il+j} = m_p^{il+j-1} + F(t_j,p) f_{t_j}^i \quad (p \in t_j{\bullet} \setminus {\bullet}t_j) \\ &m_p^{il+j} = m_p^{il+j-1} + (F(t_j,p) - F(p,t_j)) f_{t_j}^i \quad (p \in {\bullet}t_j \cap t_j{\bullet}) \\ &m_p^{il+j} = m_p^{il+j-1} \quad (p \in \mathbf{P} \backslash ({\bullet}t_j \cup t_j{\bullet})) \end{aligned} \tag{10}$$

これらはトランジション t_j が $n_{i,j}$ 回発火した時のマーキング $M_{i,j-1}$ と $M_{i,j}$ の関係を表している．1 行目により，$M_{i,j-1}$ で t_j が発火可能な最大回数以下の任意の値が $n_{i,j}$ に対して許され，特に 0 であれば発火しないことを意味する．

この制約（10）を $T_{i,j}$ と置くと，$M_i \rightarrow_m M_{i+1}$ すなわち $M_{i,0} \rightarrow_m M_{i,l}$ に対する制約 T_i は以下のように表せる．

$$T_i \overset{\text{def}}{\Longleftrightarrow} \bigwedge_{j=1}^{l} T_{i,j} \tag{11}$$

ここで，変数 m_p^{il+j} は $|\mathbf{P}| \cdot k \cdot l$ 個が導入されている．しかし，$p \in \mathbf{P} \backslash ({\bullet}t_j \cup t_j{\bullet})$ の場合，m_p^{il+j} と m_p^{il+j-1} は同じ値になる．そこで，これらを同じ変数とみなし m_p^{il+j} を m_p^{il+j-1} で置き換えれば省略することが可能である [13]．これにより，$M \rightarrow_m M'$ を SAT 符号化した時に必要な節数は，$M \rightarrow M'$ を SAT 符号化した時に必要な節数と同程度に抑えられる．

5.3 デッドロック検出の制約

マーキング M_i がデッドロックである条件は以下の制約 D_i で表現できる．

$$D_i \overset{\text{def}}{\Longleftrightarrow} \bigwedge_{t \in T} \bigvee_{p \in {\bullet}t} (m_p^{il} < F(p,t)) \tag{12}$$

多重発火モデルでの $\rightarrow_m$ 遷移はトランジションが一つも発火しない場合も含んでいる．すなわち，i ステップ目より前にデッドロックが発生する場合にも，D_i は充足可能となる．したがって，k ステップ目以内でデッドロックする条件は D_k として表せる．

以上で，デッドロック検出のための制約が得られた．多重発火モデルでの $\rightarrow_m$ 遷移のステップ長 k およびトークン数と多重発火回数の上限が与えられた時，以下の制約は，その範囲内でデッドロックがある時，かつその時に限り充足可能である．

$$I \wedge \bigwedge_{i=0}^{k-1} T_i \wedge D_k \tag{13}$$

5.4 SAT 符号化の例

制約（13）を Sugar 等の SAT 型制約ソルバーに与えることで SAT 符号化は自動的に行える．したがって，ここでは簡単な例に対する順序符号化の結果のみを示す．例えば，図 1 のペトリネットでの遷移 $M_{0,0} \xrightarrow{t_1^{n_{0,1}}} M_{0,1}$ で，プレイス p_1 に対する制約は以下のようになる．

$$m^0_{p_1} \geq f^0_{t_1}, \qquad m^1_{p_1} = m^0_{p_1} - f^0_{t_1}$$

順序符号化では各変数 x と各値 a に対し，$x \leq a$ を意味する命題変数を導入する．今，トークン数および多重発火回数の上限を 3 とする．この時，$m^0_{p_1}$ に対して $m^0_{p_1} \leq 0$, $m^0_{p_1} \leq 1$, $m^0_{p_1} \leq 2$ を表す 3 つの命題変数が導入される．また，制約 $m^0_{p_1} \geq f^0_{t_1}$ の符号化で以下の 3 つの節が得られる．

$$\neg(m^0_{p_1} \leq 0) \vee (f^0_{t_1} \leq 0), \qquad \neg(m^0_{p_1} \leq 1) \vee (f^0_{t_1} \leq 1)$$
$$\neg(m^0_{p_1} \leq 2) \vee (f^0_{t_1} \leq 2)$$

これらは，それぞれ $m^0_{p_1}$ が a 以下ならば $f^0_{t_1}$ も a 以下であるという条件を表している．

制約 $m^1_{p_1} = m^0_{p_1} - f^0_{t_1}$ は $m^1_{p_1} \geq m^0_{p_1} - f^0_{t_1}$ と $m^1_{p_1} \leq m^0_{p_1} - f^0_{t_1}$ の 2 つの制約に分解され，各々が SAT 符号化される．$m^1_{p_1} \geq m^0_{p_1} - f^0_{t_1}$ に対する節は以下のようになる．

$$(m^0_{p_1} \leq 0) \vee \neg(f^0_{t_1} \leq 0) \vee \neg(m^1_{p_1} \leq 0)$$
$$(m^0_{p_1} \leq 1) \vee \neg(f^0_{t_1} \leq 0) \vee \neg(m^1_{p_1} \leq 1)$$
$$(m^0_{p_1} \leq 1) \vee \neg(f^0_{t_1} \leq 1) \vee \neg(m^1_{p_1} \leq 0)$$
$$(m^0_{p_1} \leq 2) \vee \neg(f^0_{t_1} \leq 0) \vee \neg(m^1_{p_1} \leq 2)$$
$$(m^0_{p_1} \leq 2) \vee \neg(f^0_{t_1} \leq 1) \vee \neg(m^1_{p_1} \leq 1)$$
$$(m^0_{p_1} \leq 2) \vee \neg(f^0_{t_1} \leq 2) \vee \neg(m^1_{p_1} \leq 0)$$

例えば 5 行目は，$f^0_{t_1}$ と $m^1_{p_1}$ が 1 以下ならば，$m^0_{p_1}$ は 2 以下であるという条件を表している．

6. 多重発火モデルの SAT 解法の実装

前節で提案した多重発火モデルの制約では，トランジションの順序，トークン数および多重発火回数の上限が未定であり，実装においてはそれらを定める必要がある．また，3.3 節で述べたように，有界モデル検査では時刻 0 から時刻 k までの状態遷移の系列 $M_0 \to_m M_1 \to_m \cdots \to_m M_k$ において，デッドロック条件 D を満たす状態 M_i が存在するかどうかを判定し，デッドロックが検出できなければステップ長 k を増加させる必要がある．

そこで，本節ではこれらを含め以下の項目について，採用した実装方法を説明する．

(1) トランジションの順序の決定方法
(2) ステップ長の上限 k の増加方法
(3) トークン数および多重発火回数の上限の決定方法
(4) インクリメンタル SAT 解法の導入方法

6.1 トランジションの順序の決定方法

本論文の実装では，アルゴリズムの詳細は異なっているが，論文 [13] と同様の深さ優先探索を採用した．

深さ優先探索を用いる方法では，ペトリネットを有向グラフとみなし，初期マーキングでトークンを持つプレイスを 1 つ選び，そこから深さ優先探索でペトリネットを辿る．新たなプレイス（またはトランジション）を辿れなくなった場合，まだ辿っていないプレイスのうち，初期マーキングでトークンを持つプレイスを 1 つ選び，再度深さ優先探索で辿る．そこでトランジションを辿った順に順序を設定する．例えば，図 1 の場合，t_1, t_2, t_4, t_3 の順となる．

幅優先探索による方法も考えられるが，深さ優先探索のほうがより遠いプレイスまでトークンが移動する可能性が高くなり，デッドロック検出までのステップ長が短くなると考えられる．

なお，論文 [13] の方法では，入力側のプレイスがすでに辿られた場合にのみ，その先へ進む．したがって，図 1 の場合，t_1, t_2, t_3, t_4 の順となり，違いが生じる．しかし本論文での実装方法のほうが単純で，それほど差は大きくないが予備実験の結果でも良かったため，上記の方法を採用した．

6.2 ステップ長の上限 k の増加方法

一般に，SAT ソルバーでは SAT の判定よりも UNSAT の判定のほうが時間を要する．有界モデル検査では，デッドロックを検出できるまで複数回 UNSAT の判定が生じる．そのため，できるだけ UNSAT の判定回数が少なくなるようにステップ長 k を増加させるほうが好ましい．

予備実験として，ステップ長 k を 1 ずつ増やす場合，1.5 倍ずつ増やす場合，2 倍ずつ増やす場合の 3 通りを比較した．その結果，2 倍ずつ増やす場合の結果が比較的良かったため，本論文の実装ではその方法を採用した．

6.3 トークン数および多重発火回数の上限の決定方法

5 で述べた制約を SAT 符号化するためには，各プレイスのトークン数を表す変数 m_p^{il+j} および各トランジションの多重発火回数を表す変数 f_t^i の上限を定める必要がある．上限を小さく設定すると，デッドロックを検出できない可能性があり，逆に大きく設定すると SAT 符号化後の節数が増大し，求解速度が低下する恐れがある．

そこで，本研究ではこれらの上限値をおおまかに見積もって初期値とし，一定のステップ長以内でデッドロックを検出できない場合は，その上限を増加させる方法を採用した．

まず，各プレイス p のトークン数の上限を $N(p)$ とし，その初期値の決め方について説明する．初期マーキング M_0 でのトークン数の最大値 $\max_{p \in \mathbf{P}} M_0(p)$ を N_0 で表し，アークの多重度の最大値 $\max_{(u,v) \in \{\mathbf{P}\times\mathbf{T}\} \cup \{\mathbf{T}\times\mathbf{P}\}} F(u,v)$ を W

Algorithm 1 インクリメンタル SAT 解法を用いたデッドロック検出手続き

```
1: k' = 0; k = 1
2: 各プレイス p に対し，N(p) を式 (14) から計算
3: φ = I を SAT 符号化した節集合
4: loop
5:    各トランジション t に対し，N(t) を式 (15) から計算
6:    φ = φ ∪ (T_{k'}, ..., T_{k-1} を SAT 符号化した節集合 )
7:    if φ ∪ (D_k を SAT 符号化した節集合 ) が充足可能 then
8:       break /* デッドロック検出 */
9:    end if
10:   k' = k; k = 2k
11:   各プレイス p に対し，N(p) の値を更新
12: end loop
```

で表す．この時，$N(p)$ の初期値を以下のように定める．

$$N(p) = \max(N_0, W) \tag{14}$$

$N(p)$ の増加方法については，$k \leq 64$ （すなわち最初の 7 回）の間は上記の初期値を用い，その後，k が 2 倍されるごとに $N(p)$ も 2 倍する方法を採用した．

次に，各トランジション t の多重発火回数の上限を $N(t)$ とし，その決め方について説明する．$N(t)$ は，現在の $N(p)$ の値から可能な多重発火回数を計算して定めるのが妥当である．すなわち，以下のようにして求める方法を採用した．

$$N(t) = \min\left(\min_{p \in \bullet t}\left\lfloor \frac{N(p)}{F(p,t)} \right\rfloor, \min_{p \in t\bullet}\left\lfloor \frac{N(p)}{F(t,p)} \right\rfloor\right) \tag{15}$$

6.4 インクリメンタル SAT 解法の利用

本論文のシステムは，インクリメンタル SAT 解法が利用可能な SAT ソルバーである GlueMiniSat [10] と，それを利用する API ライブラリである iSAT Library [16] を利用して実装した．

Algorithm 1 に，インクリメンタル SAT 解法を用いたデッドロック検出のアルゴリズムの概要を示す．アルゴリズム中で I, T_i, D_i は式（13）中の制約を表している．また，k は有界モデル検査のステップ長を表し，k' は 1 回前の k の値を表す．まず，I のみを SAT 符号化し，ϕ に代入する（3 行目）．6 行目で増加した k の分だけ制約 T_i に対応した節を追加する．7 行目では，k ステップ以内にデッドロックが存在するという条件 D_k を仮定として SAT ソルバーで求解している．この仮定は，SAT ソルバーに節として永続的に追加されるわけではない．したがって，その結果が UNSAT であったとしても，解探索の途中で獲得された学習節は次のループでも保持され，再利用される．

7. 評価実験

評価実験には Model Checking Contest 2017 Reachability Deadlock 部門の問題から選んだ 297 問を使用した．これらの問題はデッドロックがないことが証明されていない問題（デッドロックがあることがわかっている問題もしくはデッドロックの有無が分かっていない問題）である．

最初の実験では，以下の 2 通りの手法について比較する．

(1) 連続発火モデル

(2) 多重発火モデル

次の実験では，Model Checking Contest 2017 の優勝および準優勝システムと比較を行い，提案手法の有効性について評価する．

なお，実行時間はすべて Ubuntu 14.04（Xeon 3.5GHz, メモリ 16GB）上で計測し，デッドロック検出までの制限時間は 3600 秒とした．なお，この制限時間は Model Checking Contest 2017 での設定と同一である．

7.1 2 通りの手法の比較

本論文での提案手法である連続発火モデルと多重発火モデルの比較を行う．連続発火モデルは既存の SAT 型手法である [13] を一般のペトリネットに拡張したものが対応し，多重発火モデルは連続発火モデルに多重発火を導入した手法である．なお，安全ペトリネットについては，多重発火モデルと既存の SAT 型手法 [13] は同じ手法となる．

まず**図 3** に，与えられた時間内に何問デッドロックを検出できているかを表すグラフを示す．例えば，500 秒以内にデッドロックを検出できた問題数は，$y = 500$ と交差した点での x の値となる．このグラフを見ると，どの制限時間に対しても多重発火モデルのほうが優れており，制限時間が延びるにつれ徐々にその差がひらいている．

表 1 により詳細の比較結果を示す．合計 297 問ある問題を，初期マーキング M_0 でのトークン数の最大値 $N_0 = \max_{p \in \mathbf{P}} M_0(p)$ により分類し，$N_0 = 1$ の場合，N_0 が 2 以上 10 以下の場合などに分けて示している．N_0 の欄の右側は，その分類における問題数である．また，連続発火モデルと多重発火モデルのそれぞれに対し，3600 秒の制限時間内にデッドロックを検出できた問題を「解けた問題数」に，解けた問題に対するステップ長（Algorithm 1 でデッドロック検出した時の k の値）の平均値を「平均ステップ長」に，解けた問題に対する CPU 時間（秒）の平均値を「平均 CPU 時間（秒）」に示している．

まず「解けた問題数」を比較すると，多重発火モデルのほうが合計で 15 問多く，N_0 のどの範囲においても連続発火モデル以上になっている．N_0 が大きくなるにつれて解けた問題数の差が大きくなっており，特に $N_0 \geq 11$ の場合に差が生じている．これは，平均 CPU 時間を見ても同様の傾向になっている．

すなわち，初期マーキングでのトークン数の最大値 N_0 が大きい問題について，多重発火モデルがより優れているといえる．ただし，N_0 が小さい問題についても連続発火モデルとほぼ同等である．

これは，4.2 節で示したように，多重発火モデルのほうがより短いステップ長でデッドロックに到達できるためだと

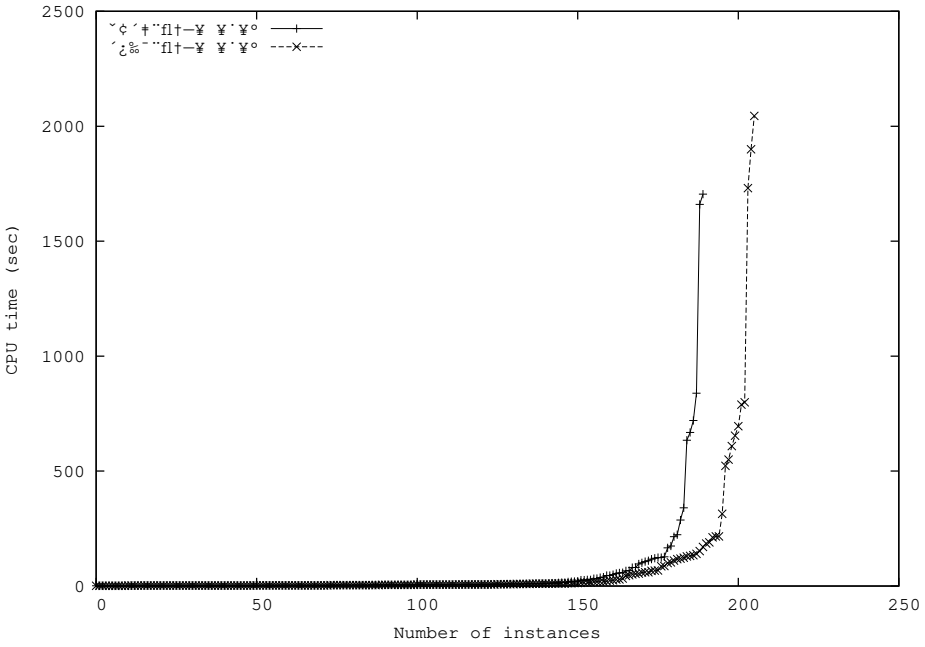


図 3 各手法で与えられた制限時間内にデッドロックを検出できた問題数
Fig. 3 Number of instances succeeded to detect deadlock within a given amount of time by each method

表 1 各手法でデッドロックを検出できた問題数，平均ステップ長，平均 CPU 時間 (N_0 は M_0 中のトークン数の最大値)
Table 1 Number of instances succeeded to detect deadlock, average number of steps, average CPU times by each method (N_0 is the maximum number of tokens in M_0)

		連続発火モデル			多重発火モデル		
N_0	問題数	解けた問題数	平均ステップ長	平均 CPU 時間（秒）	解けた問題数	平均ステップ長	平均 CPU 時間（秒）
1	161	142	6.9	228.1	142	6.7	231.0
2 ∼ 10	69	38	21.5	1415.0	38	20.0	1422.0
11 ∼ 20	16	5	21.0	2318.0	6	15.0	2059.0
21 ∼ 50	15	5	29.0	1611.0	8	4.0	416.0
51 ∼	36	0	—	—	11	7.0	316.1
合計	297	190			205		

考えられる．N_0 が大きいほど多重発火できる回数が増え，より効果が大きくなる．このことは表 1 の「平均ステップ長」の値からも確認できる．例えば，N_0 が 21 以上 50 以下の場合，連続発火モデルでデッドロックを検出するには平均 29 ステップを要したが，多重発火モデルではわずか 4 ステップとなっている．このように，デッドロックに到達するまでのステップ長が短く済むことが性能向上に寄与したと考えられる．

7.2 既存手法との比較

次に提案手法である多重発火モデルと SAT 型でない既存手法との比較結果を**表 2** に示す．比較には，Model Checking Contest 2017 Reachability 部門の優勝ソルバーである LoLA と，2 番目に結果が良かった Tapaal を用いた．

表 2 各手法でデッドロックを検出できた問題数
Table 2 Number of instances succeeded to detect deadlock by each method

N_0	問題数	多重発火モデル	LoLA	Tapaal
1	161	142	143	129
2 ∼ 10	69	38	63	59
11 ∼ 20	16	6	15	10
21 ∼ 50	15	8	14	12
51 ∼	36	11	30	31
合計	297	205	265	241

LoLA，Tapaal は SAT 型手法ではなく，ともにマーキンググラフを用いた手法である．マーキンググラフはペトリネットの状態を展開していく手法である．

制限時間内にデッドロックを検出できた数を比較した

表 3 Angiogenesis-PT での結果
Table 3 Results of Angiogenesis-PT

		多重発火モデル		LoLA	
問題名	N_0	CPU 時間（秒）	メモリ使用量（バイト）	CPU 時間（秒）	メモリ使用量（バイト）
Angiogenesis-PT-10	10	1.19	4,979,764	3.35	402,292
Angiogenesis-PT-15	15	1.36	4,986,748	36.97	2,171,764
Angiogenesis-PT-20	20	1.59	5,094,012	132.65	6,628,212
Angiogenesis-PT-25	25	1.35	5,092,920	Timeout	—
Angiogenesis-PT-50	50	1.62	5,115,492	—	Memory over

所，$N_0 \geq 2$ では提案手法が他手法よりも劣っていた．これは，節数がトークン数上限 N_0 の 2 乗に比例して増加することが一因だと考えられる．このような問題では，対数符号化法 [6], [8] を使用することで解ける可能性があり，今後の検討課題である．また，$N_0 = 1$ の場合では LoLA とほぼ同等で，Tapaal よりも優れていた．

しかし，LoLA，Tapaal が解けず，提案手法のみが解けた問題が 9 問あった．うち 7 問が Model Checking Contest 2017 に参加したいずれのソルバーでもデッドロックを検出できなかった問題である．そのうちの Angiogenesis-PT の問題について，LoLA との比較結果を**表 3** に示す．

Angiogenesis-PT はプレイス数，トランジション数，アーク数は一定で，初期マーキングにおけるトークン数がパラメータとなっている問題である．表 3 での N_0 は初期マーキングでの各プレイスのトークン数の最大値を表している．表を見ると，提案手法は N_0 の値に関わらず，CPU 時間とメモリ使用量はほぼ一定である．一方，LoLA では，N_0 が増加するにつれ CPU 時間とメモリ使用量が増大し，$N_0 = 25$ で 3600 秒の制限時間内で終了せず，$N_0 = 50$ では 16GB のメモリでメモリーオーバーとなっている．Angiogenesis-PT は多重発火が有効な問題であり，N_0 が増加するにつれ節数は増大するが，ステップ長は増加せずほぼ一定の時間で解けている．

LoLA はペトリネットの状態を展開していく手法を用いているが，この手法は状態空間爆発が起こりやすい．一方，提案手法の場合，多重発火が有効な問題では，CPU 時間およびメモリ使用量がそれほど増加せずに，より大きなトークン数に対応できている．この点は他手法にない大きな特徴といえる．しかし，一般にどのような問題に対して優れているのかについては，現時点では明確でなく今後の研究課題である．

8. おわりに

本論文では，SAT ソルバーを用いて一般のペトリネットでのデッドロック検出を行う方法を述べた．提案手法の特徴は以下の通りである．

(1) SAT ソルバーによる一般のペトリネットでのデッドロック検出の実現
非負整数値であるトークン数を表現するために**順序符号化** [19], [20] を用いた．これにより，既存の SAT 型手法である [13], [15] では対応できなかった一般のペトリネットでのデッドロック検出が可能となった．

(2) SAT ソルバーによる効率良いデッドロック検出を可能とするための多重発火モデルの導入
既存の SAT 型手法である [13] で採用されていたモデル（連続発火モデルと呼ぶ）よりも短いステップ長でデッドロック検出が可能となるモデルを提案した．これにより全般に性能が向上したが，特にトークン数が比較的多い問題に対する効果が大きいことがわかった．

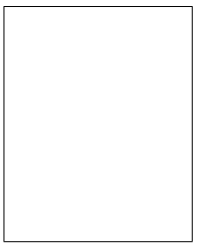
SAT 型でない既存手法との比較では，Model Checking Contest 2017 デッドロック検出部門で優勝したツール LoLA と，2 番目に性能がよかった Tapaal と比較した．既存手法がいずれもデッドロックの検出に失敗した問題での検出に成功し，提案手法の有効性が確認できた．

謝辞 本研究は JSPS 科研費 16H0280 の助成を受けたものです．

参考文献

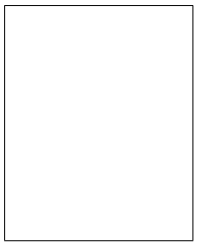
[1] 番原睦則，鍋島英知：SAT 技術の進化，情報処理，Vol. 57, No. 8, pp. 704–709 (2016).
[2] 番原睦則，田村直之：SAT によるシステム検証，人工知能学会誌， Vol. 25, No. 1, pp. 122–129 (2010).
[3] Biere, A.: Bounded Model Checking, *Handbook of Satisfiability*, IOS Press, pp. 457–481 (2009).
[4] Biere, A., Cimatti, A., Clarke, E. M. and Zhu, Y.: Symbolic Model Checking without BDDs, *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 1999), LNCS 1579*, pp. 193–207 (1999).
[5] Eén, N. and Sörensson, N.: Temporal Induction by Incremental SAT Solving, *Electronic Notes in Theoretical Computer Science*, Vol. 89, No. 4 (2003).
[6] Gelder, A. V.: Another Look at Graph Coloring via Propositional Satisfiability, *Discrete Applied Mathematics*, Vol. 156, No. 2, pp. 230–243 (2008).
[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, No. 1, pp. 57–67 (2010).
[8] Iwama, K. and Miyazaki, S.: SAT-Variable Complexity of Hard Combinatorial Problems, *Proceedings of the IFIP 13th World Computer Congress*, pp. 253–258 (1994).
[9] Knuth, D. E.: *Satisfiability*, The Art of Computer Programming, Vol. 4, Fascicle 6, Addison-Wesley Profes-

sional (2015).
[10] 鍋島英知，岩沼宏治，井上克巳：GlueMiniSat 2.2.5: 単位伝搬を促す学習節の積極的獲得戦略に基づく高速 SAT ソルバー，コンピュータソフトウェア，Vol. 29, No. 4, pp. 146–160 (2012).
[11] 鍋島英知，岩沼宏治，井上克巳：SAT ソルバーの最近の進展，情報処理， Vol. 57, No. 8, pp. 724–729 (2016).
[12] 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, pp. 68–76 (2010).
[13] Ogata, S., Tsuchiya, T. and Kikuno, T.: SAT-Based Verification of Safe Petri Nets, *Proceedings of the Second International Conference on Automated Technology for Verification and Analysis (ATVA 2004)*, pp. 79–92 (2004).
[14] 奥川峻史：ペトリネットの基礎，共立出版 (1995).
[15] Rodríguez, C. and Schwoon, S.: Cunf: A Tool for Unfolding and Verifying Petri Nets with Read Arcs, *Proceedings of the 11th International Symposium on Automated Technology for Verification and Analysis (ATVA 2013)*, pp. 492–495 (2013).
[16] 迫 龍哉，宋 剛秀，番原睦則，田村直之，鍋島英知，井上克巳：インクリメンタル SAT 解法ライブラリとその応用，コンピュータソフトウェア，Vol. 33, No. 4, pp. 16–29 (2016).
[17] 寸田智也，宋 剛秀，番原睦則，田村直之：SAT 技術を用いたペトリネットのデッドロック検出手法の提案，2017 年度人工知能学会全国大会 (第 31 回) 論文集 (2016).
[18] 寸田智也，宋 剛秀，番原睦則，田村直之：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，日本ソフトウェア科学会第 33 回大会，pp. 513–521 (2016).
[19] Tamura, N., Taga, A., Kitagawa, S. and Banbara, M.: Compiling Finite Linear CSP into SAT, *Constraints*, Vol. 14, No. 2, pp. 254–272 (2009).
[20] 田村直之，丹生智也，番原睦則：SAT 変換に基づく制約ソルバーとその性能評価，コンピュータソフトウェア，Vol. 27, No. 4, pp. 183–196 (2010).
[21] 田村直之，丹生智也，番原睦則：制約最適化問題と SAT 符号化，人工知能学会誌，Vol. 25, No. 1, pp. 77–85 (2010).



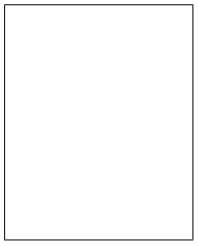
寸田 智也

2016 年神戸大学工学部情報知能工学科卒業．同年神戸大学大学院システム情報学研究科情報科学専攻博士前期課程入学．2018 年同大学大学院修了予定．制約プログラミング，SAT 技術の応用に興味を持つ．



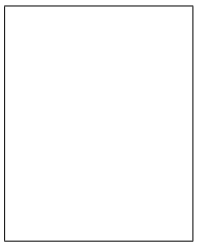
宋 剛秀 （正会員）

2004 年神戸大学工学部電気電子工学科卒業．2006 年同大学大学院自然科学研究科電気電子工学専攻修了．サントリー（株）を経て，2011 年総合研究大学院大学複合科学研究科情報学専攻博士後期課程修了．2012 年より神戸大学情報基盤センター助教．制約プログラミング，SAT 技術およびそれらの応用に興味をもつ．日本ソフトウェア科学会，人工知能学会会員．



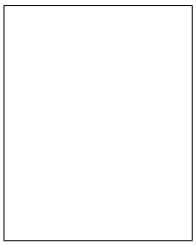
番原 睦則 （正会員）

1994 年神戸大学理学部数学科卒業．1996 年同大学大学院自然科学研究科博士課程前期数学専攻修了．1996 年国立奈良工業高等専門学校助手．1998 年同校講師．2003 年より神戸大学情報基盤センター講師．2007 年同校准教授．博士 (工学)．論理プログラミング，制約プログラミング，SAT 技術等に興味をもつ．日本ソフトウェア科学会，人工知能学会会員．



田村 直之 （正会員）

1980 年神戸大学理学部物理学科卒業．1985 年同大学大学院自然科学研究科博士課程システム科学専攻修了．学術博士．1985 年日本 IBM 東京基礎研究所入社．1988 年神戸大学工学部勤務．2003 年より神戸大学情報基盤センター教授．論理プログラミング，制約プログラミング，SAT 技術等に興味をもつ．日本ソフトウェア科学会，人工知能学会会員．



井上 克巳 （正会員）

1982 年京都大学工学部数理工学科卒業．1984 年同大学大学院工学研究科数理工学専攻修了．京都大学博士 (工学)．松下電器産業 (株)，(財) 新世代コンピュータ技術開発機構，豊橋技術科学大学，神戸大学を経て，2004 年より国立情報学研究所．現在，同情報学プリンシプル研究系教授および総合研究大学院大学複合科学研究科情報学専攻教授．2015 年東京工業大学大学院情報理工学研究科客員教授，2016 年より同情報理工学院情報工学系特任教授．人工知能，論理プログラミングに興味を持つ．日本ソフトウェア科学会，人工知能学会，AAAI 会員．