整数型変数を持つ並行システムに対する有界モデル検査手法

Bounded Model Checking for Concurrent Systems with Integer Type Variables

井上 裕之 † 土屋 達弘 † 菊野 亨 †
Hiroyuki Inoue† Tatsuhiro Tsuchiya† Tohru Kikuno†

1. まえがき

モデル検査（model checking）とは，状態機械でモデル化されたシステムの状態空間を探索することにより，時相論理式等で記述された性質が，そのモデルにおいて満たされるか否かを形式的に判定する手法である．しかしながら，現実的なシステムでは探索すべき状態が莫大な数になるため，適用が困難になることがある．そのため，様々な効率化の手法が研究されてきている．このような手法の一つに，充足可能性判定を利用した有界モデル検査がある．有界という名が示すとおり，有界モデル検査では，初期状態から一定の回数の遷移を考慮してモデル検査を行う．具体的には，一定回数の遷移数までのシステム動作をブール値の数式 (述語) として記号的に表現し，その充足可能性を判定することで検証を行う．元来有界モデル検査ではシステム動作の記号表現に命題論理式を用いていた．しかし，プログラムは 2 値ではなく，一般の整数値を変数値として扱うことが普通であるので，そのような変数を考慮した検証が行われなければならない．

この有界モデル検査は，SMT(Satisfiability Modulo Theories) ソルバを利用することで実現することができる．近年において，SMT の関連技術は急速な進歩を遂げ，著しい高速化が実現されている．しかし，並行プログラムのようなソフトウェアシステムを対象とした場合，このような期待される性能は得られないことが多い．この原因は，並行プログラムのような非同期システムでは，遷移関係を表す論理式の簡潔な表現が得られないため，充足可能性判定の対象となる論理式が大きくなり検証時間が増大するためと考えられる．

そこで本研究では，この問題を解決するため，整数型の変数を扱うことが可能である SMT ソルバを利用した並行システムに対する新しい有界モデル検査手法を提案する．具体的には，状態遷移の記号表現において，従来法における 1 回の遷移を表現する論理式と同程度の大きさで，複数個の遷移を表現する方法を提案する．この結果，従来の有界モデル検査手法での論理式と比較して，より広い状態空間について考慮できる．

本報告の構成は以下の通りである．2 章ではモデル検査について説明する．3 章では検査対象である並行システムと提案手法の説明で用いる用語の諸定義を行う．4 章では提案手法の説明を行い，5 章では検証実験の結果を示す．最後に，6 章では本研究のまとめと今後の課題について述べる．

2. モデル検査

2.1 モデル検査

モデル検査は検査対象のシステムが形式仕様を満たしているかを検証する手法である[7]．検査対象を状態遷移システムとしてモデル化し，モデルの状態を網羅的に探索することで，モデルが時相論理式で表現された性質を満たすか否かを検査する．時相論理は時間経過によって真偽が変化する命題を表現するための論理体系であり，モデル検査では，状態変化を時間経過とみなす．時相論理にはいくつかの種類があるが，モデル検査においては，線形時相論理と計算木論理のいずれかが利用される場合がほとんどである．

形式的には，有限状態遷移システムは 3 項組 (S, I, T) となる．S は状態の集合，I は初期状態の集合，$T(\subseteq S \times S)$ は遷移関係である．モデル検査では，与えられた時相論理式は全ての初期状態について評価され，どの初期状態でも式が真か，そうでないかを判定する．

並行システムのモデル検査で問題となるのが状態爆発である．状態爆発とは，並行システムの様に，システムを構成する要素が複数存在し独立に動作する場合，システム全体での状態数が莫大となり，検証が困難になるという問題である．

2.2 有界モデル検査

有界モデル検査では，遷移関係を数式で記号的に表し，充足可能性判定によって検証を行う[1]．充足可能性判定とは，与えられた式が true になる変数への付値があるかどうかを判定する手続きのことである．検査手法について詳しくは，4.1 節で説明する．有界という名が示すとおり，有界モデル検査では初期状態から一定の回数の遷移を考慮してモデル検査を行う．従来の有界モデル検査では命題論理式を扱う．このような命題論理式の充足可能性問題 (SAT) を解くツールを，SAT ソルバと呼ぶ．

一方，SMT(Satisfiability Modulo Theories) とは線形算術等の論理を加えることで SAT を一般化したものである．そして，SMT ソルバとは，これらの論理上の式の充足可能性を決定するツールであり，無限状態のシステムに対して有界モデル検査を行うことができる．

2.3 二分決定グラフに基づく記号モデル検査

記号表現を用いるモデル検査は一般に記号モデル検査と呼ばれる[6]．最も一般的な記号モデル検査手法では，状態集合や遷移関係を二分決定グラフ（BDD）を用いてブール式によって表現，操作することで検証を行う．この手法を実装したツールとしては，SMV ファミリー (SMV，NuSMV，Cadence SMV) が著名である．

また，ブール式以外の記号表現の方法としては，モデル検査ツール ALV[3] で用いられている Presburger 式がある。Presburger 式の特性により，ALV では上下限のない整数変数を状態変数とする状態数無限のプログラムについてもモデル検査を行うことができる．

3. 諸 定 義

3.1 検証対象のシステムの記述

検証の対象となるシステムは，3 項組 $C = (V, I, E)$ によって記述されるものとする[2]．ここで，V, I, E はそれぞれ

† 大阪大学 Osaka University

- V : システムの変数の有限集合
- I : 初期状態
- E : イベントの有限集合

を表す．V に含まれる変数は整数型もしくは列挙型とする．I はシステムの動作開始時の状況を表す条件であり，V 上の述語で与える．E の各イベント e はアトミックな動作であり，イベントの動作可能条件とアクションの組として，

$$e = (e \text{ が動作可能となる条件}, e \text{ のアクション})$$

という形式で記述される．動作可能条件は，V 上の述語とする．また，アクションは，そのアクション後（すなわち次状態）での変数集合を V' として，$V \cup V'$ 上の述語で与える．

例として，図 1 に 2 つのバッファの動作について記述したものを示す．2 つのバッファの容量に制限はなく，これらは，無限にアイテムを格納することのできるバッファである．

システムの動作は遷移システムとして解釈される[2]．遷移システムは，3 項組 $M = (S, I, T)$ であり，有向グラフを用いて図示できる．各項は，

- S : 状態の集合
- I : 初期状態の集合（$I \subseteqq S$）
- T : 遷移関係（$T \subseteqq S \times S$）

を表す．各状態は上記の記述 C における変数の値の組として表されるので，状態集合 S は，各変数の値域のデカルト積として得ることができる．変数 v の値域を $\mathrm{domain}(v)$ と表すことにすると，例えば，$V = \{v_1, v_2, \cdots, v_n\}$ であるようなシステムの各状態 $s \in S$ は以下のように表される．

$$s \equiv \bigwedge_{i=1}^{n} (v_i = x_i) \text{ ただし } x_i \in domain(v_i)$$

I は初期状態の集合を表す．T は，イベント e に関する遷移関係 $T_e \subseteq S \times S$ を用いて，

$$T \equiv \bigcup_{e \in E} T_e \cup \{(s,s) | s \in S \wedge \forall e \in E \ : \ s \notin enabled(e)\}$$

と定義される．ここで，T_e は次のように定義される．イベント e の動作可能な状態の集合を $\mathrm{enabled}(e)$，イベント e のアクションによっておこる状態遷移の集合を $\mathrm{action}(e)$ とする．より正確には，$\mathrm{enabled}(e)$ は，e の動作可能条件である述語を満たす状態集合と定義する．また，$\mathrm{action}(e)$ は，e のアクションに対する述語を満たす状態と次状態の対 (s, s') の集合と定義する．これを用いて，T_e は

$$T_e = \{(s,s') | s \in enabled(e) \wedge (s,s') \in action(e)\}$$

と表される．例えば，図 1 における e_1 の場合であれば，

$$T_{e_1} = \{((produced, comsumed, buffer1, buffer2, producer), (produced', consumed', buffer1', buffer2', producer')) |$$
$$producer = Idle \wedge producer' = Ready$$
$$\wedge produced' = produced \wedge consumed' = consumed$$
$$\wedge buffer1' = buffer1 \wedge buffer2' = buffer2\}$$

となる．

V 上の述語 f が与えられた場合，f を満たす変数値を有する状態の集合は一意に定まる．従って，状態集合と対応する述語は同じ記号 f によって表すものとする．また，与えられた状態 s について，f が満たされるなら真，それ以外の場合偽となる関数を $f(s)$ と表記する．

$$\begin{aligned}
V \equiv \{ & \\
& produced, consumed, buffer1, buffer2 : \text{positive integer} \\
& producer : \{Idle, Ready\} \\
I \equiv & (produced = consumed = buffer1 = buffer2 = 0) \\
E \equiv \{ & \\
& e_1 = (producer = Idle, \\
& \quad propucer' = Ready), \\
& e_2 = (producer = Ready, \\
& \quad produced' = produced + 1 \\
& \quad \wedge (buffer1' = buffer1 + 1 \wedge buffer2' = buffer2 \\
& \quad \vee buffer1' = buffer1 \wedge buffer2' = buffer2 + 1)), \\
& e_3 = (producer = Ready, \\
& \quad producer' = Idel)' \\
& e_4 = (buffer1 > 0 \vee buffer2 > 0, \\
& \quad consumed' = consumed + 1 \\
& \quad \wedge (buffer1 > 0 \wedge buffer1' = buffer1 - 1 \\
& \quad \wedge buffer2' = buffer2 \\
& \quad \vee buffer1' = buffer1 \\
& \quad \wedge buffer2 > 0 \wedge buffer2' = buffer2 - 1)) \\
\}
\end{aligned}$$

図 1 システムの記述例 — 2 つの無限バッファの挙動

$V \cup V'$ 上の述語が与えられた場合，その述語を満たす状態対の集合も一意に定まるので，状態対の集合についても状態集合の場合と同様な表記法を用いる．

3.2 検証可能な性質

今回提案する手法においては，可達性についてのみ検証可能である．可達性とはシステムの初期状態から到達可能ないかなる状態においても，ある性質 p が満たされているという性質のことで，これは，時相論理式を用いて，AGp(論理式 p は全ての状態で成り立つ) と表すことができる．p は V 上の式として与えられるものとする．

4. 提 案 手 法

4.1 従 来 法

遷移関係 T を $V \cup V'$ 上の述語で表すと，次のように定義することができる．

$$T(s,s') \equiv \bigvee_{e \in E} T_e(s,s') \vee (s = s') \wedge \bigwedge_{e \in E} s \notin enabeled(e)$$

これを図 1 の例を用いて表すと以下のようになる，

$$
\begin{aligned}
T(s,s') \equiv\ & (nw = 0 \wedge nr' = nr + 1 \wedge nw' = nw) \\
& \vee (nr > 0 \wedge nr' = nr - 1 \wedge nw' = nw) \\
& \vee (nr = 0 \wedge nw' = nw + 1 \wedge nr' = nr) \\
& \vee (nw > 0 \wedge nw' = nw - 1 \wedge nr' = nr) \\
& \vee (\neg(nw = 0) \wedge \neg(nr > 0) \wedge \neg(nr = 0) \\
& \wedge \neg(nw > 0) \\
& \wedge nr' = nr \wedge nw' = nw)
\end{aligned}
$$

$$
\begin{aligned}
V \equiv\ & (nr, nw : \text{positive integer}) \\
I \equiv\ & (nr = nw = 0) \\
E \equiv\ & \{ \\
& e_1 = (nw = 0, nr' = nr + 1), \\
& e_2 = (nr > 0, nr' = nr - 1), \\
& e_3 = (nr = 0, nw = 0, nw' = nw + 1), \\
& e_4 = (nw > 0, nw' = nw - 1) \\
& \}
\end{aligned}
$$

図 2 Readers-Writers アルゴリズム

このような T を用いると，あるシステムにおいて，AGp という性質が成り立つかどうかという検証は，k 回以下の遷移のみを考慮した場合，以下の式の充足可能性問題に帰着できる．

$$
\begin{aligned}
& I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \cdots \wedge T(s_{k-1}, s_k) \\
& \wedge (\neg p(s_0) \vee \neg p(s_1) \vee \cdots \vee \neg p(s_k))
\end{aligned}
$$

この式の 1 行目は，$s_0, s_1, \cdots, s_k$ が初期状態から実行可能な系列であるときかつそのときのみ真となる．2 行目は，$s_0, s_1, \cdots, s_k$ のいずれかの状態において $\neg p$ が満たされるときかつそのときのみ真となる．よって，この式全体としては，初期状態から k 回以内のイベントによって遷移できるいずれかの状態において，$\neg p$ が満たされるときかつそのときのみ充足可能である．

すなわち，この式が充足可能となるならば，与えられたシステムにおいて $\neg p$ が満たされる，言い換えると，p が満たされないような状態が存在するということがいえ，AGp という性質は成り立たないということがいえる．

図 2 の例の場合，この式の 2 行目がどのようになるかを示す．i 回遷移後の状態 s_i での変数をそれぞれ nr_i, nw_i として表す．検証したい性質 p を $((nr = 0 \vee nw = 0) \wedge nw \leqq 1)$ とすると, 以下のようになる．

$$
\begin{aligned}
& (\neg((nr_0 = 0 \vee nw_0 = 0) \wedge nw_0 \leqq 1) \\
& \vee \neg((nr_1 = 0 \vee nw_1 = 0) \wedge nw_1 \leqq 1) \\
& \vee \cdots \\
& \vee \neg((nr_k = 0 \vee nw_k = 0) \wedge nw_k \leqq 1))
\end{aligned}
$$

この手法の問題点は，充足可能性を判定すべき式が大きくなってしまうため，その判定に多くの時間がかかってしまうという点である．

4.2 提 案 法

以上のような問題点を解決するために，論理式の大きさを変えずに，より多くの遷移について考慮できるような改良手法を考える．

$D_e(s, s') \equiv T_e(s, s') \vee (s = s')$ という式を定義する．これは，イベント e によって，状態 s から s' への遷移がおこるか，あるいは状態 s と状態 s' が等しい状態であるときかつそのときにみ真となる．

これを 4.1 節の図 2 を用いて表すと，イベント e_1 の場合，以下のようになる．

$$
\begin{aligned}
D_{e_1}(s, s') \equiv\ & \vee (nw = 0 \wedge nr' = nr + 1 \wedge nw' = nw) \\
& \vee (nr' = nr \wedge nw' = nw)
\end{aligned}
$$

提案手法では，以下のように式 φ を生成し，その充足可能性を判定することで検証を行う．

$$
\begin{aligned}
\varphi = & I(s_0) \\
& \wedge D_{e_1}(s_0, s_1) \wedge D_{e_2}(s_1, s_2) \wedge \cdots \wedge D_{e_n}(s_{n-1}, s_n) \\
& \wedge D_{e_1}(s_n, s_{n+1}) \wedge D_{e_2}(s_{n+1}, s_{n+2}) \\
& \qquad \wedge \cdots \wedge D_{e_n}(s_{2n-1}, s_{2n}) \\
& \cdots \\
& \wedge D_{e_1}(s_{(k-1)*n}, s_{(k-1)*n+1}) \wedge \cdots \wedge D_{e_n}(s_{k*n-1}, s_{k*n}) \\
& \wedge \neg p(s_{k*n})
\end{aligned}
$$

ここで 4.1 節の図 2 の例を用いて，$\neg p(s_{k \times n})$ を表す．検証したい性質 p を $((nr = 0 \vee nw = 0) \wedge nw \leqq 1)$ とすると，以下のようになる．

$$
\neg((nr_{k \times n} = 0 \vee nw_{k \times n} = 0) \wedge nw_{k \times n} \leqq 1)
$$

状態 s_{k*n} は初期状態から k 回以下の遷移で到達可能なすべての状態と，$k \times n$ 回までの遷移で到達可能な状態のうち，一部の状態をとり得る．すなわち，φ が充足不能ならば，初期状態から k 回以下の遷移で到達可能な状態および $k \times n$ 回の遷移で到達可能な状態のうちの一部の状態のなかには，$\neg p$ が満たされるような状態は存在しないということ，充足可能である場合は，最大で $k \times n$ 回の遷移で到達可能な状態のなかに，$\neg p$ を満たす，すなわち p が満たされないような状態が存在するので，p は常には満たされないといえる．

以上より，φ が充足可能であった場合は，与えられたシステムにおいて AGp は満たされないと結論づけることができる．また，φ が充足不能であった場合は，少なくとも k 回までの遷移と，$k \times n$ 回までの遷移のうちの一部を考慮した場合であれば満たされるといえる．

5. 実　　験

5.1 実験の条件，検証対象

提案法を用いて実際に検証を行った．実験は，Intel Core 2 Duo 2GHz および 2GByte のメモリからなる計算機上で行った．

検証の対象としたシステムは，図 3，4 に示すようなアルゴリズムのシステムである．従来法と提案法を比較するために，SMT ソルバである Yices[5] を用いて検証を行う．

検証する性質は，図 3 の Barber アルゴリズムでは，席に座っている客を cinchair，帰った客を cleave，客が来た状態を bavail，美容師が髪を切っている状態を bbusy，美容師が待機

$$
\begin{aligned}
V \equiv \{ & \\
& cinchair, cleave, bavail, bbusy, bdone : \text{positive integer} \\
& pc_1, pc_2, pc_3 : \{1, 2\} \\
\} & \\
I \equiv \{ & \\
& einchair = cleave = bavail = bbusy = bdone = 0 \\
& pc_1 = pc_2 = pc_3 = 1 \\
\} & \\
E \equiv \{ & \\
& e_1 = (pc_1 = 1 \wedge pc_1' = 2 \\
& \qquad \wedge cinchair < bavail \wedge cinchair' = cinchair + 1) \\
& e_2 = (pc_1 = 2 \wedge pc_1' = 1 \\
& \qquad \wedge cleave < bdone \wedge cleave' = cleave + 1) \\
& e_3 = (pc_2 = 1 \wedge pc_2' = 2 \\
& \qquad \wedge bavail' = bavail + 1) \\
& e_4 = (pc_2 = 2 \wedge pc_2' = 1 \\
& \qquad \wedge bbusy < cinchair \wedge bbusy' = bbusy + 1) \\
& e_5 = (pc_3 = 1 \wedge pc_3 = 2 \\
& \qquad \wedge bdone < bbusy \wedge bdone' = bdone + 1) \\
& e_6 = (pc_3 = 2 \wedge pc_3 = 1 \\
& \qquad \wedge bdone = cleave) \\
\} &
\end{aligned}
$$

図 3 Barber アルゴリズム（プロセス数 3）

している状態を bdone としたとき，時相論理式によって

$$
\begin{aligned}
\mathbf{AG}(& cinchair \geq cleave \wedge bavail \geq bbusy \geq bdone \\
& \wedge cinchair \leq bavail \wedge bbusy \leq cinchair \\
& \wedge cleave \leq bdone)
\end{aligned}
$$

と与えられる．これは初期状態から到達可能ないずれの状態においても不整合（空きの席を確認して座ろうとしたら，すでに客が座っているなど）が生じないことを表している．また検証する性質は，図 4 の Bakery アルゴリズムでは，critical section を C としたとき，時相論理式によって

$$
\begin{aligned}
\mathbf{AG}(\neg((& pc1 = C \wedge pc2 = C) \vee (pc1 = C \wedge pc3 = C) \\
& \vee (pc1 = C \wedge pc4 = C) \vee (pc2 = C \wedge pc3 = C) \\
& \vee (pc2 = C \wedge pc4 = C) \vee (pc3 = C \wedge pc4 = C)))
\end{aligned}
$$

と表される性質を検証する．これは初期状態から到達可能ないずれの状態においてもプロセスの 2 つ以上が同時に critical section に入らないということを表している．

また，有界モデル検査以外のモデル検査手法と比較するために，同じ例に対して二分決定グラフ（BDD）を用いた記号モデル検査ツールである NuSMV[4] と，Presburger 式を用いた記号モデル検査ツールの ALV[3] を用いた検証も同様に行う．

5.2 結果，考察

5.2.1 Barber アルゴリズム

図 3 の Barber アルゴリズム（居眠り床屋問題）に対して，SMT ソルバ Yices を用いて，従来法と提案法での検証を行った結果を表 1 に示す．なお，提案法では以下のいずれの場合においても論理式は充足不能となり，k 回以下の遷移を考慮した場合では，検証対象とした性質は満たされるという結果を得た．また，検証時間とは論理式の充足可能性判定にかかった時間のことである．なお，k が 25 までの場合しか示していないが，25 より大きな場合でも検証を行うことができた．次に，従来法と提案法を用いた検証結果を比較すると，提案法の方が高速である． ここで，従来法と提案法で書かれたシステムを表す式と検証したい性質の式の変数と演算子の数を数えた結果

$$
\begin{aligned}
V \equiv \{ & \\
& a, b, c, d : \text{positive integer} \\
& pc_1, pc_2, pc_3, pc_4 : T, W, C \\
\} & \\
I \equiv \{ & \\
& a = b = c = d = 0 \\
& pc_1 = pc_2 = pc_3 = pc_4 = T \\
\} & \\
E \equiv \{ & \\
& e_1 = (pc_1 = T \wedge pc_1' = W \\
& \qquad \wedge (a' = b + 1 \vee a' = c + 1 \vee a' = d + 1) \\
& \qquad \wedge a' > b \wedge a' > c \wedge a' > d) \\
& e_2 = (pc_1 = W \wedge pc_1' = C \\
& \qquad \wedge (b = 0 \vee a < b) \wedge (c = 0 \vee a < c) \wedge (d = 0 \vee a < d)) \\
& e_3 = (pc_1 = C \wedge pc_1' = T \\
& \qquad \wedge a' = 0) \\
& e_4 = (pc_2 = T \wedge pc_2' = W \\
& \qquad \wedge (b' = a + 1 \vee b' = c + 1 \vee b' = d + 1) \\
& \qquad \wedge b' > a \wedge b' > c \wedge b' > d) \\
& e_5 = (pc_2 = W \wedge pc_2' = C \\
& \qquad \wedge (a = 0 \vee b < a) \wedge (c = 0 \vee b < c) \wedge (d = 0 \vee b < d)) \\
& e_6 = (pc_2 = C \wedge pc_2' = T \\
& \qquad \wedge b' = 0) \\
& e_7 = (pc_3 = T \wedge pc_3' = W \\
& \qquad \wedge (c' = a + 1 \vee c' = b + 1 \vee c' = d + 1) \\
& \qquad \wedge c' > a \wedge c' > b \wedge c' > d) \\
& e_8 = (pc_3 = W \wedge pc_3' = C \\
& \qquad \wedge (a = 0 \vee c < a) \wedge (b = 0 \vee c < b) \wedge (d = 0 \vee c < d)) \\
& e_9 = (pc_3 = C \wedge pc_3' = T \\
& \qquad \wedge c' = 0) \\
& e_{10} = (pc_4 = T \wedge pc_4' = W \\
& \qquad \wedge (d' = a + 1 \vee d' = b + 1 \vee d' = c + 1) \\
& \qquad \wedge d' > a \wedge d' > b \wedge d' > c) \\
& e_{11} = (pc_4 = W \wedge pc_4' = C \\
& \qquad \wedge (a = 0 \vee d < a) \wedge (b = 0 \vee d < b) \wedge c = 0 \vee d < c)) \\
& e_{12} = (pc_4 = C \wedge pc_4' = T \\
& \qquad \wedge d' = 0) \\
\} &
\end{aligned}
$$

図 4 Bakery アルゴリズム（プロセス数 4）

表 1 従来法と提案法を用いた検証結果 (Barber)

	検証結果 (秒)			検証結果 (秒)	
k	従来法	提案法	k	従来法	提案法
1	0.005	0.006	14	0.479	0.035
2	0.007	0.009	15	0.614	0.050
3	0.009	0.009	16	0.645	0.049
4	0.013	0.011	17	0.625	0.060
5	0.021	0.013	18	1.749	0.060
6	0.027	0.016	19	1.241	0.060
7	0.055	0.020	20	1.074	0.051
8	0.075	0.021	21	1.825	0.137
9	0.136	0.023	22	1.322	0.076
10	0.178	0.026	23	1.501	0.065
11	0.231	0.028	24	1.668	0.094
12	0.323	0.041	25	2.695	0.115
13	0.393	0.043			

表 2 従来法と提案法での変数と演算子の数 (プロセス数 4 の Barber)

従来法				
k	変数の数 (個)	and の数	or の数	その他
1	16	14	3	116
2	24	21	5	206
3	32	28	7	298
4	40	35	9	389
5	48	42	11	480
6	56	49	13	571
7	64	56	15	662
8	72	63	17	753
9	80	70	19	844
10	88	77	21	935
11	96	84	23	1026
12	104	91	25	1117
13	112	98	27	1200
14	120	105	29	1299
15	128	112	31	1390
16	136	119	33	1481
17	144	126	35	1572
18	152	133	37	1663
19	160	140	39	1754
20	168	147	41	1845
21	176	154	43	1936
22	184	161	45	2027
23	192	168	47	2118
24	200	175	49	2209
25	208	182	51	2300

提案法				
k	変数の数 (個)	and の数	or の数	その他
1	26	18	7	64
2	49	29	13	103
3	60	40	19	142
4	77	51	25	181
5	94	62	31	220
6	111	73	37	259
7	128	84	43	298
8	145	95	49	336
9	162	106	55	376
10	179	117	61	415
11	196	128	67	454
12	213	139	73	493
13	230	150	79	532
14	247	161	85	571
15	264	172	91	610
16	281	183	97	649
17	298	194	103	688
18	315	205	109	727
19	332	216	115	766
20	349	227	121	805
21	366	238	127	844
22	383	249	133	883
23	400	260	139	922
24	417	271	145	961
25	434	282	151	1000

表 3 従来法と提案法それぞれの到達可能な状態数 (プロセス数 4 の Barber)

	状態数 (個)			状態数 (個)	
k	従来法	提案法	k	従来法	提案法
1	2	32	14	26	1732
2	3	148	15	28	1874
3	4	287	16	30	1992
4	6	417	17	32	2144
5	8	549	18	34	2266
6	10	659	19	36	2404
7	12	796	20	38	2536
8	14	928	21	40	2678
9	16	1070	22	42	2796
10	18	1188	23	44	2948
11	20	1340	24	46	3070
12	22	1462	25	48	3208
13	24	1600			

を表 2 に示す．2 つを比較すると同じステップ数 k において提案法の方が，従来法よりも and，or の数が多い．しかし，演算子全体で見ると従来法の方が数が多い．これより，充足可能性判定にかかる時間が同程度となり，結果として検証時間に大きな差がなくなったと考えられる．また，別のプログラムによって計測した従来法と提案法での検証できる状態数を表 3 に示す．従来法と提案法で比較すると，到達可能な状態数は提案法の方が従来法より多く，$k = 25$ の時にはおよそ 67 倍の差がある．

ここで，従来法と提案法の次に，有界モデル検査以外のモデル検査手法を用いたツール，NuSMV，ALV によって検証を行った結果を表 4 に示す． どちらのツールを用いた場合も検証の対象とした性質は満たされるという結果を得た．左側が NuSMV を用いた場合の結果，右側が ALV を用いて検証を行った場合であり，x は変数の上限である．ただし，ALV は

表 4 有界モデル検査以外のモデル検査手法を用いた検証結果 (Barber)

NuSMV	
x	検証時間（秒）
1	0.007
2	0.008
3	0.010
4	0.012
5	0.014
6	0.017
7	0.018
8	0.026
9	0.029
10	0.036
11	0.038
12	0.054
15	0.070
16	0.096
17	0.103
18	0.122
19	0.123
20	0.149
21	0.157
22	0.184
23	0.170
24	0.216
25	0.224

ALV
検証時間（秒）
0.06

表 6 従来法と提案法での変数と演算子の数（プロセス数 4 の Bakery）

従来法				
k	変数の数 (個)	and の個数	or の個数	その他
1	16	46	35	262
2	24	85	68	507
3	32	124	101	752
4	40	163	134	997
5	48	202	167	1242
6	56	241	200	1487
7	64	280	233	1732
8	72	319	266	1977
9	80	358	299	2222
10	88	397	332	2467
11	96	436	365	2712
12	104	475	398	2957
13	112	514	431	3202
14	120	553	464	3447
15	128	592	497	3692
16	136	631	530	3937
17	144	670	563	4182
18	152	709	596	4427
19	160	748	629	4672
20	168	787	662	4917
21	176	826	695	5162
22	184	865	728	5407
23	192	904	761	5652
24	200	943	794	5897
25	208	982	827	6142

提案法				
k	変数の数 (個)	and の個数	or の個数	その他
1	41	27	30	125
2	73	47	58	233
3	105	67	86	341
4	137	87	114	449
5	169	107	142	557
6	201	127	170	665
7	233	147	233	773
8	265	167	226	881
9	297	187	254	989
10	329	207	282	1097
11	361	227	310	1205
12	393	247	338	1313
13	425	267	366	1421
14	457	287	394	1529
15	489	307	422	1637
16	521	327	450	1745
17	553	478	347	1853
18	585	367	506	1961
19	617	387	534	2069
20	649	407	562	2177
21	681	427	590	2285
22	713	447	618	2393
23	745	467	646	2501
24	777	487	674	2609
25	809	507	702	2717

変数の上限，実行ステップ数の上限の指定を行う必要がないため，実行結果と示す時間は一つとなる．

従来法を用いた検証結果と NuSMV，ALV を用いた検証結果との比較を行うと，検証時間だけを見ると従来法を用いた検証が最も時間がかかっている．また，提案法を用いた検証結果と NuSMV，ALV を用いた検証結果と比較すると，NuSMV については検証時間に大きな差がないが，ALV については $k \leqq 19$ のとき，提案法を用いた検証の方が短時間である．

5.2.2 Bakery アルゴリズム

次に，図 4 の Bakery のアルゴリズムに対して検証を行い，従来法と提案法の結果を表 5 に示す．

従来法と提案法を比較してみると，提案法の方が高速である．しかし，明らかに表 1 での結果よりも検証時間の差が大きい．

表 5 従来法と提案法を用いた検証結果（Bakery）

	検証時間 (秒)			検証時間 (秒)	
k	従来法	提案法	k	従来法	提案法
1	0.006	0.008	14	427.382	1.541
2	0.012	0.020	15	1060.64	4.508
3	0.022	0.041	16	2643.615	3.693
4	0.049	0.080	17	6714.140	6.005
5	0.160	0.160	18	9267.436	4.016
6	0.476	0.178	19	25713.673	5.505
7	0.901	0.428	20	$\geq$ 30000	8.571
8	2.167	0.589	21	$\geq$ 30000	31.581
9	5.554	0.692	22	$\geq$ 30000	15.505
10	13.867	0.957	23	$\geq$ 30000	50.173
11	20.734	0.981	24	$\geq$ 30000	12.697
12	48.532	2.954	25	$\geq$ 30000	68.364
13	210.032	2.131			

表 7 従来法と提案法それぞれの到達可能な状態数（プロセス数 4 の Bakery）

	状態数 (個)			状態数 (個)	
k	従来法	提案法	k	従来法	提案法
1	5	169	14	533	1833
2	21	297	15	569	1961
3	57	425	16	621	2089
4	109	553	17	661	2217
5	149	681	18	697	2345
6	185	809	19	749	2473
7	237	937	20	789	2601
8	277	1065	21	825	2729
9	313	1193	22	877	2857
10	365	1321	23	917	2985
11	405	1449	24	953	3113
12	441	1577	25	1005	3241
13	493	1705			

そこで，従来法と提案法で書かれたシステムを表す式と検証したい性質の式の変数と演算子の数を数えた結果を表 6 に示す．2 つを比較すると同じステップ数 k において従来法の方が，提案法よりも and の数がおよそ 2 倍多い．また，他においても従来法の方が数が多い．これより，充足可能性判定に時間がかかり，結果として検証時間に大きな差が出たと考えられる．また，別のプログラムによって計測した従来法と提案法での検証できる状態数を表 7 に示す．従来法と提案法で比較すると，到達可能な状態数は提案法の法が従来法より多く，$k = 25$ の時およそ 3 倍ほどの差がある．

次に，NuSMV と ALV によって検証を行った結果を表 8 に示す．NuSMV を用いた検証結果と，提案法を用いた検証結果

表 8 有界モデル検査以外の検査手法を用いた検証結果（Bakery）

NuSMV	
x	検証時間（秒）
1	0.011
2	0.013
3	0.015
4	0.019
5	0.024
6	0.031
7	0.037
8	0.046
9	0.057
10	0.075
11	0.085
12	0.102
13	0.116
14	0.130
15	0.144
16	0.168
17	0.199
18	0.227
19	0.252
20	0.281
21	0.313
22	0.354
23	0.377
24	0.412
25	0.463

ALV
検証時間（秒）
24.46

と比べると，提案法を用いた検証の方が時間がかかっている．また，ALV においては，$k \leqq 20$ のとき，提案法を用いた検証の方が時間が短時間である．ALV を用いた検証との比較では，今回検証したシステムにおいては k の値が小さいときに，効果的であるが，k の値が増えると ALV を用いた検証，つまり，Presburger 式を用いたモデル検査の方が効果的である．

以上の結果から，SMT ソルバを利用した有界モデル検査では，従来法に比べ，提案法の方が高速で，より多くの状態を検証できるといえる．また，提案法を用いることで，検査対象としたシステムによっては，BDD を用いたモデル検査手法と比べた場合よりも効果的な検証が実現できた．

6. まとめと今後の課題

本稿では，有界モデル検査の新しい手法を提案した．提案した手法では，近年急速な進歩を遂げ，著しい高速化が実現されている SMT ソルバを用いる．この手法では，状態遷移の記号表現において，従来法における 1 回の遷移を表現する論理式と同程度の大きさで，複数個の遷移を表現した．これにより，従来の有界モデル検査と比較して，より広い状態空間について考慮でき，並行システムに対して，従来法よりも効果的な検証が可能になった．また，有界モデル検査以外のモデル検査手法である NuSMV と比較しても十分に効果的であることを示した．しかし，ALV との比較については，効果的であるということは示すことができなかった．

提案手法ではシステムの可達性，つまり，システムの初期状態から到達可能ないかなる状態においてもある性質が満たされるということについてのみ検証した．実際のシステムの検証においては，この性質だけの検証では十分ではない．そこで検証可能な性質を増やしていくことが今後の課題として挙げられる．また，バグが混入した並行システムの検証のように，有界モデル検査がそれ以外の手法と比べてより効果的と考えられる状況における性能評価や，提案法を実装したツールの作成などが今後の課題である．

参 考 文 献

1) A.Biere, A.Cimatti, E.M. Clarke, and Y.Zhu. Symbolic model checking without BDDs. In *Proceedings of Tools and Algorithms for the Analysis and Construction of Systems (TACAS'99), LNCS 1579*, pp. 193–207. Springer-Verlag, 1999.
2) T. Bultan, R. Gerber, and W. Pugh. Model-checking concurrent systems with unbounded integer variables: Symbolic representations, approximations, and experimental results. *ACM Trans. on Progr. Languages and Syst.*, 21(4), pp. 747–789, 1999.
3) T.Bultan and T.Yavuz-Kahveci. Action language verifier. In *Proceedings of the 16th IEEE international conference on Automated software engineering (ASE '01)*, pp. 382, Washington, DC, USA, 2001. IEEE Computer Society.
4) A.Cimatti, E.M. Clarke, E.Giunchiglia, F.Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In E.Brinksma and K.G. Larsen eds., *CAV*, Vol. 2404 of *LNCS*, pp. 359–364. Springer, 2002.
5) B. Dutertre and L. D. Moura. The yices smt solver. Technical report, 2006.
6) K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic, 1993.
7) Edmund M. Clarke, Jr., and Orna Grumberg, and Doron A.Peled. Model Checking. MIT Press, 1999.