社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
DC2011-20 (2011-10)

上界のない整数型変数を有する並行システムに対するk帰納法を用いたモデル検査

井上　裕之† 土屋　達弘† 菊野　亨†

† 大阪大学　〒565-0871 大阪府吹田市山田丘 1-5
E-mail: †{h-inoue,t-tutiya,kikuno}@ist.osaka-u.ac.jp

あらまし　モデル検査において上界のない整数型変数を有するシステムを対象とした場合，状態数は無限となるため，状態全てを個々に探索することは不可能である．そこで，k帰納法を用いたモデル検査によって，この無限状態の検証を行う方法を議論する．SMTソルバを用いて，上界のない整数型変数を含む式の充足可能性判定を行うことで，k帰納法を実現することができる．しかし，並行プログラムを対象とした場合，検証に時間がかかってしまうことが多い．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である．そこで本研究では，その問題を解決するため，よりコンパクトな遷移関係の式表現を用いた手法を提案する．
キーワード　モデル検査，k帰納法，SMT，並行システム

K-induction-based model checking of concurrent systems with unbounded integer variables

Hiroyuki INOUE†, Tatsuhiro TSUCHIYA†, and Tohru KIKUNO†

† Osaka University　1-5 Yamadaoka, Suita, Osaka 562-0871, Japan
E-mail: †{h-inoue,t-tutiya,kikuno}@ist.osaka-u.ac.jp

Abstract　We discuss k-induction-based model checking that uses a Satisfiability Modulo Theories (SMT) solver. The state space of a system with unbounded integer variables is infinite; thus it is impossible to visit all of its states. K-induction is one of the model checking approaches that can be used to reason about such an infinite state space. In this approach, the problem of model checking is reduced to the satisfiability problem that can be solved by an SMT solver. However, this approach does not work effectively when applied to concurrent programs, because the formula representing the behaviors of systems with high concurrency tends to become very large. To overcome this problem, we propose an alternative approach that uses more compact formulas.
Key words　model checking, k-induction, SMT, concurrent systems

1. はじめに

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モデル検査がある．

具体的には，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．また，上界のない整数型変数を有するシステムの場合，状態は無限となる．そのため，全ての個々の状態を探索することは不可能である．そこで，k帰納法を用いることで，全ての状態についてある性質が成り立つことを証明する．このk帰納法を用いたモデル検査は，SMT (Satifiability 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，上界のない整数型変数を有する並行システムに対する k 帰納法を用いたモデル検査手法を提案する

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とともに，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．3 章では k 帰納法の説明を行い，4 章では提案手法の説明を行う．5 章では実験結果を示し，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. 諸 定 義

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 [4]．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，モデルが与えられた性質を満たすか否かを検査する．

充足可能性判定を用いてモデル検査を行う事ができる．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．

従来の充足可能性判定を用いた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) [1] 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判定するツールである．

2.2 検証対象のシステム記述

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 [3]．ここで，V, I, E はそれぞれ

- V : システムの変数の有限集合
- I : 初期条件
- 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 が動作可能となる条件, e 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．図 1 にシステムの記述例を示す．システムの動作は遷移システムとして解釈される [3]．遷移システムは，3 項組 $M = (S, \hat{I}, T)$ である．

- S : 状態の集合
- $\hat{I}$: 初期状態の集合 ($\hat{I} \subseteq S$)
- T : 遷移関係 ($T \subseteq S \times S$)

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 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 $s =< x_1, x_2, \cdots, x_n > \in S$ は以下のように表される．

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

$\hat{I}$ 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 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 action(e) とする．より正確には，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．

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

図 1 システムの記述例 — 2つの無限バッファの挙動

Fig. 1 An example of a concurrent system: consumer-producer with double infinite buffers.

また，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．

$V \cup V'$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．

2.3 検証可能な性質

今回提案する手法においては，インバリアントについてのみ検証可能である．インバリアントp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，pが満たされているという性質のことである．pはV上の述語として与えられるものとする．

3. k帰納法

ここではk帰納法 (k-induction) という手法について説明する [9]．まず，$k=1$の場合，すなわち，通常の帰納法について説明する．以下の2条件が成り立てば，帰納法を用いて任意の状態sにおいてpが成り立たつことがいえる．

- 任意の状態sについて，sが初期状態ならばpが成り立つこと
- 任意の状態sについて，pが成り立つならば，sのどの次状態においてもpが成り立つこと

ただし，pが動作において常に成り立つ場合でも，2番目の条件は実際には成り立たないことが多い．そこで，以下のように一般化して，条件の前提を強めたものがk帰納法である．

- 任意の動作における最初のk状態すべてで，pが成り立たつこと
- $s_0 \rightarrow s_1, s_1 \rightarrow s_2, \ldots, s_{k-1} \rightarrow s_k$である任意の$k+1$状態$s_0, \ldots, s_k$について，$k$状態$s_0, \ldots, s_{k-1}$のすべてで$p$が成り立たつなら，$s_k$でも$p$が成り立たつこと

k帰納法が成功しない場合は，1番目の条件が成り立たない，もしくは，2番目の条件のみが成り立たない，の二つの場合に分けられる．前者の場合はもちろんpが成り立たない状態への到達性がいえる．一方，後者の場合は，kを増やして検証を繰り返す必要がある．システムの状態空間が有限のときにpが成り立たない状態に到達可能であるならば，いずれ1番目の条件が成り立たなくなる．しかしpが偽となる状態へ到達できないとき，状態空間が有限であっても2番目の条件を成り立たせるkが存在するとは限らない．たとえば，到達不可能な状態s_1, s_2が存在し，pはs_1で成り立たつがs_2では成り立たないものとする．また，$s_1 \rightarrow s_1$, $s_1 \rightarrow s_2$という遷移が存在するものとする．このとき，$s_1 s_1, \ldots s_1 s_2$という状態系列を考えると，kをいくら大きくしても2番目の条件は成り立たないことが分かる．この問題は到達不能な状態を通ってループする経路がある場合発生する．そこで，2番目の条件における対象とする$k+1$個の状態において，重複する状態がないという制約を付加することで，この問題を避けることができる．

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．

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

このようなTを用いると，あるシステムにおいて，インバリアントかどうか成り立つかどうかというk帰納法による検証は，以下の2式の充足可能性問題に帰着できる [5]．

- $I(s_0) \wedge \bigwedge_{i=0}^{k-2} T(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k-1} \neg p(s_i)$
- $\bigwedge_{i=0}^{k-1} p(s_i) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge \neg p(s_k)$

この1つ目の式は，$s_0, s_1, \cdots, s_{k-1}$が初期状態から実行可能な系列であり，その全ての状態でpが成り立つときかつそのときのみ充足不能となる．2つ目の式は，$s_0, s_1, \cdots, s_{k-1}$の全ての状態において$p$が成り立つ場合に，常に$s_k$においても$p$が成り立つときかつそのときのみ充足不能となる．この1つ目の式が充足可能となり，k帰納法が失敗するか2つの式どちらもが充足不能になるまで，kの値を増やして検証を行う．

さらに2番目の式について，前述したように，s_0からs_kにおいて状態の重複の回避のため，以下の式を付加する．

$$\bigwedge_{0 \leq j < j' \leq k} \bigvee_{1 \leq i \leq |V|} v_i^j \neq v_i^{j'}$$

ここでv_i^jは，状態s_jにおける変数v_iを表す．

提案手法における問題点は，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

式が大きくなるという従来法の問題点を解決するために，論理式を小さくする方法を考える．

まず，$D_e(s,s') \equiv T_e(s,s') \vee (s = s')$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．このとき，以下の式を充足させる付値を考える．

$$D_{e1}(s_0, s_1) \wedge D_{e2}(s_1, s_2) \cdots \wedge D_{en}(s_{n-1}, s_n) \wedge D_{e1}(s_n, s_{n+1}) \cdots \wedge D_{en}(s_{(k-1)*n-1}, s_{(k-1)*n})$$

すると，この付値によって表される状態系列$s_0, \ldots, s_{(k-1)*n}$は，スタッタリング遷移（同じ状態を繰り返す遷移）を許して，0

回以上，最大で $k*n$ 回のイベントによる遷移による動作を表すことになる．さらに，遷移よって変化しない変数に関する項が省略できるため，T_e の論理和で遷移関係を表す従来法より，式がコンパクトになることが知られている [8].

提案手法では，この式を利用して以下のように 2 式を生成し，その充足可能性を判定することで検証を行う．

- $I(s_0) \wedge D_{e1}(s_0, s_1) \wedge D_{e2}(s_1, s_2) \cdots \wedge D_{en}(s_{n-1}, s_n) \wedge D_{e1}(s_n, s_{n+1}) \cdots \wedge D_{en}(s_{(k-1)*n-1}, s_{(k-1)*n}) \wedge \bigvee_{i=0}^{(k-1)*n} \neg p(s_i)$
- $\bigwedge_{i=0}^{k*n-1} p(s_i) \wedge D_{e1}(s_0, s_1) \wedge D_{e2}(s_1, s_2) \cdots \wedge D_{en}(s_{n-1}, s_n) \wedge F_0 \wedge D_{e1}(s_n, s_{n+1}) \cdots \wedge D_{en}(s_{k*n-1}, s_{k*n}) \wedge F_{k-1} \wedge \neg p(s_{k*n})$

1 番目の式については，$k-1$ 回以上の遷移を考慮しているため，p が成り立たない状態に到達可能な場合，従来法で必要な値より小さい k で充足する．一方，2 番目の式については，正確に k 回のイベントによる遷移を考慮する必要がある．そのため，上式では F_i という式を用いている．これは，e_1 から e_n のイベントの実行を一通り考慮した場合，正確に一回アクションが実行されることを表す制約である．

また，k 帰納法の所で説明したように同じ状態をたどってしまうというループによって，k をいくら大きくしても 2 番目の式が充足不能になることがある．また，$D_e(s, s')$ では $\vee(s = s')$ の部分が表すように，提案手法では次状態と現状態が同じということを表している．このため，同じ状態から遷移しないということも起こりうる．そこで，2 番目の式に以下の制約を付加する．

$$\bigwedge_{0 \leq j < j' \leq k*n} \bigvee_{1 \leq i \leq n} v_i^j \neq v_i^{j'}$$

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 3.06GHz および 4GByte のメモリからなる計算機上で行った．SMT ソルバは Yices [6] を用いた．検証の対象としたのは Bakery 相互排除アルゴリズムである．検証する性質は，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である．Bakery アルゴリズムのモデルとしては，文献 [2] のものを用いた．

5.2 結果，考察

まず，critical section に入らないという性質について，検証を行った．その結果，プロセスが 2 つ以上でプログラムカウンタ $pc = W$，それぞれの所持するタイムスタンプ変数が 0 の場合，どんなに k の値を増やしても k 帰納法の 2 番目の式が充足可能になり，帰納法は失敗することが分かった．この理由を調べた結果，到達不能である $pc = W$，タイムスタンプ変数が 0 である状態から始まる状態系列を，考慮しているためであることが判明した．

そこで，このような状態が到達不可能であることを，まず k 帰納法により証明した．その後に，この到達不可能であるという条件を付加してインバリアントを強化し，k 帰納法により検証を行った．表 1 に，この段階で検証にかかった時間を示す．

従来法と提案法を用いた検証時間を比較すると，プロセス数が 2，3 の時を除いて，提案法の方が高速である．また，従来法ではプロセス数が多くなっていくと検証時間が爆発的に増えていき，提案法との差がより大きくなっている．この結果からプロセス数が多い場合，つまり，並行性が高い場合，提案法の方が有効であると言える．プロセス数が小さい時は，提案手法での充足可能性判定のための論理式の長さと従来手法の論理式の長さに大きな差がないため，検証時間に差がほとんどなくなったと考えられる．

表 1　従来法と提案法の検証時間

Table 1　Verification time of the conventional method and the proposing method

		検証結果 (秒)	
プロセス数	k	従来法	提案法
2	3	0.010	0.016
3	4	0.041	0.061
4	5	0.360	0.237
5	6	1.826	0.972
6	7	17.625	4.461
7	8	131.586	43.354
8	9	1446.467	257.503
9	10	9340.109	1885.448
10	11	87280.626	44697.797

6. まとめと今後の課題

本稿では，整数型変数を持つ並行システムに対する k 帰納法を用いたモデル検査手法を提案した．提案した手法における状態遷移の記号表現は，従来法において 1 回の遷移を表現する論理式よりもより小さくなる．これにより，充足可能性判定にかかる時間が短くなり，結果としてより高速な検証を行うことが可能となった．また，並行動作が多くなる場合に、より従来手法と提案手法での検証速度に差が生まれた．つまり，並行性が大きい場合，提案法によって k 帰納法を用いたモデル検査がより効果的に行える．

提案手法では従来手法よりも状態空間を広く探索できるため，それを生かし，k の値をより小さくすることが今後の課題として考えられる．また，他の並行システムを対象に検証を行い，性能評価を行うことや，提案手法を実装したツールの作成を目指すことも今後の課題である．

文　　献

[1] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability Modulo Theories. In Armin Biere, Hans van Maaren, and Toby Walsh, editors, Handbook of Satisfiability, IOS Press, 2009.

[2] T. Bultan, "BDD vs. constraint-based model checking: An experimental evaluation for asynchronous concurrent systems," in *Proc. of Tools and Algorithms for the Analysis and Construction of Systems (TACAS'00), LNCS 1785.* London, UK: Springer-Verlag, 2000, pp. 441–455.

[3] 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.

[4] Edmund M. Clarke, Jr., and Orna Grumberg, and Doron A. Peled. Model Checking. MIT Press, 1999.

[5] De Moura, L. and Rue, H. and Sorea, M. Bounded model checking and induction: From refutation to verification. Computer Aided Verification, 2003, 14–26, Springer

[6] B. Dutertre and L. D. Moura. The yices smt solver. Technical report, 2006.

[7] K. L. McMillan. *Symbolic Model Checking.* Kluwer Academic, 1993.

[8] S. Ogata, T. Tsuchiya, and T. Kikuno, "Sat-based verification of safe petri nets," in *Proc2. of 2nd International Symposium on Automated Technology for Verification and Analysis (ATVA 2004)*, ser. LNCS, vol. 3299, 11 2004, pp. 72–92.

[9] Sheeran, M. and Singh, S. and Stålmarck, G. Checking safety properties using induction and a SAT-solver. Formal Methods in Computer-Aided Design,127–144, 2000, Springer.