節集合の対称性を利用したSATCHMOによる充足可能性判定

坂井朱美[†] 井上克巳[††]

ボトムアップ型の定理証明器としてこれまでにDavis-Putnam手続きやSATCHMOなどが考案されている．しかし，命題論理の充足可能性判定問題はNP完全であるため，いかに高性能な定理証明器を用いても問題のサイズが大きくなれば証明時間が爆発的に増大してしまう．大きな定理証明問題をこれ以上速く解くには，定理証明問題の持つ特性を利用しなくてはならない．そこで本研究では，ボトムアップ定理証明器SATCHMOへシンメトリ（対称性）の概念を導入する．また，単位リテラル規則・純リテラル規則を導入して命題節集合の縮小化を行う．さらに，実験により充足可能性の判定を効率良くできることを確認する．

Testing the Satisfiability of a Clause Set by SATCHMO with Symmetry

AKEMI SAKAI[†] and KATSUMI INOUE[††]

The Davis-Putnam procedure and SATCHMO have worked out as efficient bottom-up propositional provers until now. However, it takes much time to solve large problems, because solving the satisfiability problem of propositional calculus is NP-complete. Then, solving these problems quickly, we must utilize the characteristics of a given set of clauses. Therefore, we implement the symmetry finding method on the bottom-up prover SATCHMO and cut many branches of proof trees by symmetries. To implement this, we add One-literal rule and Pure-literal rule to SATCHMO (called SATCHMOST). Experimental results show that SATCHMOST has a good performance and can be used practically.

1. はじめに

これまでにボトムアップ型の定理証明方法としては，古くから最もよく使われてきたDavis-Putnam手続き[1]や，最近開発されたSATCHMO[2]およびMGTP[3]などが考案されている．

Davis-Putnam手続きは，命題論理で記述された定理証明問題を単位リテラル規則，純リテラル規則，分割規則を繰り返し実行して充足可能性を判定する定理証明器である．**SATCHMO**は，一階述語論理で記述された問題を解く定理証明器であり，ホーン節においてはPrologを利用して後向き推論を行い，非ホーン節においては前向き推論を行う．また**MGTP**はSATCHMO同様ボトムアップにモデル生成を行うが，前向き推論のみで解く．

このような定理証明器における問題点は以下のとおりである．命題論理の充足可能性判定問題はNP完全である．したがって，現状でいかに高性能な定理証明器を用いても，問題のサイズが大きくなれば証明時間は爆発的に増大してしまう．大きな定理証明問題をこれ以上速く解くには，問題の持つ特性を利用しなくてはならない．このような特性の1つとして，本研究においてはシンメトリ（symmetry）の概念を採用する．

シンメトリとは証明木における対称性である．証明木においてある部分木が解けたときに，それと対称的な部分木を削除することで定理証明の効率化を実現することを考える．たとえば，我々が理論研究等において定理を証明する際に，よく「同様にして」と述べて証明を省略することがあるが，このプロセスは一種のシンメトリ操作により説明できる場合がある．よって，シンメトリを自動的に発見・利用することは，定理自動証明にとって，演繹以外のより知的な操作を導入することにもなる．実際多くの場合，命題論理の問題（た

† 東京大学大型計算機センター
Computer Center, University of Tokyo
†† 神戸大学工学部電気電子工学科
Department of Electrical and Electronics Engineering, Kobe University

とえば典型的なベンチマーク問題である**pigeon-hole問題**[4),5)]など）には，節集合中にいくつかのシンメトリが存在する．Benhamou と Sais[5)]はそのような問題の証明をシンメトリを発見することにより短縮できることを示している．また，彼らはSLRI導出[6)]，Davis-Putnam手続き，意味導出[7)]などで，シンメトリ発見による証明木の縮小化を実現している．

本研究では，ボトムアップ定理証明器SATCHMOの利用において，シンメトリ発見による証明木の縮小化を試みる．ここでの問題点は，SATCHMOがモデル生成型の定理証明器であるために，シンメトリの適用の仕方がこれまでに提案された方法とはまったく異なることである．SATCHMOでは推論を行う際に，各推論ステップで節集合に導き出された新しい事実を付け加え，更新された節集合を次の推論ステップに引き渡して推論を行う．このため，推論の各ステップでシンメトリ発見の計算を行わなければならない．推論ステップが進むに従って節集合が増大しシンメトリ発見の計算が複雑になるため，計算時間が増大する．そこで本研究においては，新しく導出された事実を付け加えるのではなく，単位リテラル規則に沿って節集合を削除するようにSATCHMOを変形したバージョンを考案した．さらに節集合を縮小するために純リテラル規則を導入した．このSATCHMOにシンメトリ発見と単位リテラル規則，純リテラル規則を導入したバージョンを**SATCHMOST**（SATCHMO with Symmetry Testing）と呼ぶ．

このSATCHMOSTの比較対象として，Davis-Putnam手続き，それにシンメトリの概念を導入したもの，SATCHMOの各ボトムアップ定理証明器をProlog上で実現し，TPTP[4)]（1000の定理証明問題集）を用いて効率に関する比較実験を行った．

本論文の構成は以下のとおりである．2章でSATCHMOとDavis-Putnam手続きについて，また3章でシンメトリについて紹介する．4章で今回提案するSATCHMOSTについて説明し，5章で実験および評価を行う．6章でその考察，7章でまとめと今後の課題について述べる．

2. ボトムアップ型命題定理証明器

2.1 SATCHMO

SATCHMO（SATisfiability CHecking by MOdel）はManthey と Bry[2)]により提案されたProlog上の非常にシンプルな充足可能性判定器である．SATCHMOは非ホーン節でのみ分岐をともなう前向き推論を行い，ホーン節ではPrologを利用して後向き推論を行う．SATCHMOはモデル生成に基礎をおく定理証明器であり，Prolog上では事実を簡単にアサートできるという利点を活かして，効率的に充足可能性の検査を行うことができる．なおSATCHMOでは値域限定された一階述語論理の節集合を扱うことができるが，本論文においては命題節集合しか扱わないものとする．

2.1.1 表記法

まず，SATCHMOが扱う節に関する定義を述べる．式

$$B_1,\ldots,B_n \rightarrow A_1;\ldots;A_m. \quad (0 \leq m, 0 \leq n)$$

を**節**といい，$A_1 \vee \ldots \vee A_m \vee \neg B_1 \vee \ldots \vee \neg B_n$ を意味し，以下のようにも表される．

$$A_1;\ldots;A_m \leftarrow B_1,\ldots,B_n. \quad (0 \leq m, 0 \leq n)$$

ここで $A_1,\ldots,A_m$ と $B_1,\ldots,B_n$ はいずれもアトムであり，$A_1;\ldots;A_m$ を節の**頭部**（ヘッド），$B_1,\ldots,B_n$ を節の**本体**（ボディ）という．上の形式の節において $m=0$ のとき，**負節**といい，空の頭部を *bottom* で表す．また $m=1$ のときは**確定節**である．$m \leq 1$ のときは**ホーン節**であり，確定節または負節である．**非ホーン節**とは $m \geq 2$ の節であり，頭部が複数のリテラルの選言である節である．また $n=0$ の節を**正節**といい，空の本体を *true* で表す．

2.1.2 基本手続き

SATCHMOの基本動作を，文献8)に示された次のPrologプログラムを用いて説明する．ここで，プログラム中の `--->` は $\rightarrow$ を意味する．$\leftarrow$ については `:-` で表され確定節にのみ許されるものとする．

```
saturate :- expandable_with(Head) ->
                    expand(Head) ; true.
expandable_with(Head) :- (Body--->Head),
                    Body, \+Head.
expand(bottom) :- !, fail.
expand(X;Y)  :- !, (expand(X);expand(Y)).
expand(Atom) :- update(Atom).
update(Atom) :- assume(Atom), saturate.
assume(Atom) :- asserta(Atom).
assume(Atom) :- retract(Atom), !, fail.
```

SATCHMOの推論では，アトムの選択とモデル生成の様子をたとえば**図1**のような**木**（証明木）として記録する．木において，枝の端末（葉）に *bottom* を生成しなければ，その枝に存在する根から端末までのアトムの集合は**モデル**である．

`saturate` の開始により，`expandable_with` を呼び出し，現在までに得られてデータベースに格納されているアトムの集合によって充足されない節を1つ見つける．ここで，**カレントブランチ**を木の根から現在

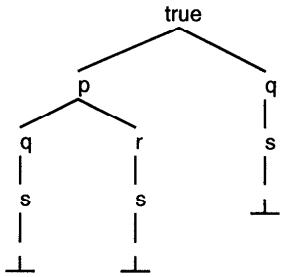

図1 節集合 S のモデル生成木
Fig. 1 Proof tree of set of clauses S.

のノードまでの道筋とすると，見つけるべき節は，そのボディのアトムがすべてカレントブランチの中に存在し，ヘッドのアトムがすべてカレントブランチの中に存在しないものである．もしこのような節が見つからなければ，節集合中にモデルがあることを意味し，**saturate** は成功する．もし節が見つかれば，それを充足させるために，現在のノードから枝を増やすように **expand** を呼ぶ．節のヘッドが *bottom* ならば，その枝を端末としてバックトラックする．また節のヘッドが複数のアトムからなるならば，そのノードからヘッドのアトムの数だけ枝を生成する．新しい葉ノードの生成は，データベースにアトムを付け加えることによって記録される．さらに深く木を築くために **saturate** を呼ぶ．データベースに格納されたアトムはバックトラックする際にそこから削除される．そして再びモデル生成を試みる．もし **saturate** が失敗に終われば，節集合は充足不可能である．

例1 節集合 S:

$true \rightarrow p; q.$
$q \rightarrow s.$
$r \rightarrow s.$
$p \rightarrow q; r.$
$q, s \rightarrow bottom.$
$p, s \rightarrow bottom.$

実行によって作られたモデル生成の様子は図1の木として記録される．このモデル生成木は最初の深さの節を左から右に探索する．S のすべての可能な道筋において *bottom* を生成するので S は充足不可能である．

2.2 Davis-Putnam 手続き

Davis-Putnam 手続きでは基礎節の集合 S を入力とする．Davis-Putnam 手続きは，以下の3つの規則を繰り返し適用することにより，S の充足可能性を調べる．

(1) **単位リテラル規則**：S 内に，単位リテラル L だけからなる節があるとすると，S から L を含む節を除き S' とする．S' が空ならば，S は充足可能である．

以上の操作の後に，S' から今除いたリテラル L の補リテラル $\overline{L}$ を含む節があれば $\overline{L}$ を除き S'' とする．S'' 内に空節 □ を含んでいれば S は充足不可能である．

(2) **純リテラル規則**：S がリテラル L を含むが $\overline{L}$ を含まないならば L は**純である**（pure）という．S が純リテラル L を含めば，L を含む節すべてを除き S' とする．S' が充足不可能であることと S が充足不可能であることとは同値である．

(3) **分割規則**：A_i, B_i, R が L, $\overline{L}$ を含まないで，S を論理積標準形にしたときに

$$S = (A_1 \vee L) \wedge \cdots \wedge (A_m \vee L) \wedge (B_1 \vee \overline{L}) \wedge \cdots \wedge (B_n \vee \overline{L}) \wedge R$$

の形式で書けるとき，$S_1 = A_1 \wedge \cdots \wedge A_m \wedge R$ と $S_2 = B_1 \wedge \cdots \wedge B_n \wedge R$ に分割する．S が充足不可能であることと $(S_1 \vee S_2)$ が充足不可能，すなわち S_1 と S_2 がともに充足不可能であることとは同値である．

なお本来の Davis-Putnam 手続きには，S からトートロジ節を除く**トートロジ規則**がある．しかしこの規則の適用によって充足可能性が変わることはないため，効率の点から本研究では用いないこととする．

例2 Davis-Putnam 手続きで節集合

$$S = \{ p \vee q \vee \neg r,\ p \vee \neg q,\ \neg p,\ r,\ u \}$$

の充足可能性を導く例を示す．

(1) $\{ p \vee q \vee \neg r,\ p \vee \neg q,\ \neg p,\ r,\ u \}$
（u に純リテラル規則を適用）

(2) $\{ p \vee q \vee \neg r,\ p \vee \neg q,\ \neg p,\ r \}$
（r に単位リテラル規則を適用）

(3) $\{ p \vee q,\ p \vee \neg q,\ \neg p \}$
（$\neg p$ に単位リテラル規則を適用）

(4) $\{ q,\ \neg q \}$
（q に単位リテラル規則を適用）

(5) { □ }

したがって，S は充足不可能である．

3. シンメトリ

3.1 シンメトリの定義

シンメトリとは証明木における対称性である．多くの場合，節集合中にいくつかのシンメトリが存在する．以下の定義は文献5)に従う．

V を命題論理変数の集合とするとき，全単射 σ :

$V \to V$ を命題変数の**置換**（permutation）という．S を節集合，c を S 中の節，σ を S 中に現れる命題変数の置換とするとき，$\sigma(c)$ は c 中の各命題変数に σ を適用して作られた節であり，$\sigma(S) = \{\sigma(c) \mid c \in S\}$ とする．もしリテラルの集合 P が，任意のリテラル l に対して $l \in P$ ならば $\bar{l} \in P$ を満たすとき**完全である**という．

定義1 P を完全なリテラル集合とし，P 中のすべてのリテラルが節集合 S に現れるとする．$\sigma : P \to P$ で定義された置換が以下を満たすならば**シンメトリ**である：

1. $\forall l \in P$ に対して $\sigma(\bar{l}) = \overline{\sigma(l)}$
2. $\sigma(S) = S$

リテラル l, l' に対して $\sigma(l) = l'$ であるシンメトリ σ が存在するとき，$(l \sim l')$ と表示する．

定義2 リテラル集合 $\{l, l_1, l_2, \ldots, l_n\}$ が節集合 S 中に存在し，$\sigma(l) = l_1, \sigma(l_1) = l_2, \ldots, \sigma(l_{n-1}) = l_n, \sigma(l_n) = l$ であるシンメトリ σ が存在するとき，$\{l, l_1, l_2, \ldots, l_n\}$ を**シンメトリサイクル**という．

例3 節集合 S を $S = \{a \vee \neg b,\ c\}$ とし，$P = \{a, \neg a, b, \neg b, c, \neg c\}$ を完全なリテラル集合とする．$\sigma : P \to P$ を以下で定義する．

$\sigma(a) = \neg b$
$\sigma(\neg a) = b$
$\sigma(b) = \neg a$
$\sigma(\neg b) = a$
$\sigma(c) = c$
$\sigma(\neg c) = \neg c$

σ はシンメトリであり，$\sigma(S) = \{\neg b \vee a,\ c\} = S$ である．また $\sigma(S)$ は S 中の節の順序を入れ換えるか，もしくはリテラルの位置を変換するかして作られた節集合である．

性質1[5] 節集合 S 中の2つのリテラル l, l' が $(l \sim l')$ であるならば，l が真となるモデルが存在することと l' が真となるモデルが存在することとは同値である．

性質2（シンメトリの必要条件）[5] 節集合 S 中で2つのリテラル l, l' がシンメトリであるならば，S 中での l の出現回数は l' のそれと同じであり，l の現れる節の長さと l' のそれは一致する．

3.2 シンメトリの効果

シンメトリを見つけ出すことにより，証明空間を縮小できることを次の例で示す．

例4 pigeon-hole 問題（n 羽の鳩を $n-1$ 個の穴に入れる．ただし1羽の鳩しか1つの穴に入ることができない）．$n = 3$ の場合の節集合を以下に示す．

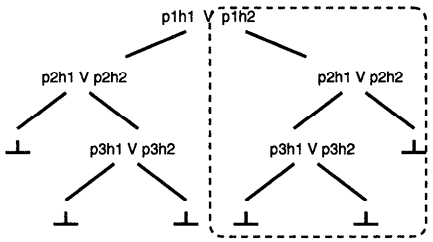

図2 pigeon-hole 問題の証明木
Fig. 2 Proof tree of pigeon-hole problem.

ここで $pihj$ は鳩 i を穴 j に入れることを意味する．

$true \to p1h1 \vee p1h2.$
$true \to p2h1 \vee p2h2.$
$true \to p3h1 \vee p3h2.$
$bottom \leftarrow p1h1 \wedge p2h1.$
$bottom \leftarrow p1h1 \wedge p3h1.$
$bottom \leftarrow p2h1 \wedge p3h1.$
$bottom \leftarrow p1h2 \wedge p2h2.$
$bottom \leftarrow p1h2 \wedge p3h2.$
$bottom \leftarrow p2h2 \wedge p3h2.$

この節集合の SATCHMO による証明木を図2に示す．この木は左右対称であるので，節集合中のシンメトリ $(p1h1 \sim p1h2)$ を見つけ出すことにより，証明木の右半分の枝をカットすることができる．点線で囲った部分がカットできる部分木であることを示す．

3.3 発見方法の概略

節集合 S 中でシンメトリを発見することは，S を変化させない置換 σ を発見することになる．以下に文献5)によるシンメトリ発見方法の概略について説明する．

置換はその最小単位である**素置換**（transposition）の積で表現することができる．つまり，すべての置換 β は素置換 (x_i, y_i) を用いて，

$$\beta = (x_1, y_1)(x_2, y_2)\cdots(x_n, y_n)$$

と書ける．たとえば，S 中で性質2を満たす2つのリテラル x_0 と y_0 がシンメトリであるかどうかを調べるとする．そのために，x_0 を y_0 で置き換えても S を変化させないような置換 σ を見つけ出す．ここで，σ を $\sigma = (x_0, y_0)\sigma'$ の形式で表し，σ' は素置換の積とする．x_0 を y_0 で置き換えることができるかどうか調べるために，x_0 が現れる各々の節中で x_0 を y_0 に置き換える．この操作を，x_0 の現れる節と y_0 の現れる節を**関連づける**という．この処理の後，x_0 を y_0 で置き換えるためにはほかにどのような素置換が必要であるかを，性質2を用いて計算して σ' を調べ，各素置換についても同じ処理を適用する．素置換が σ の

他に見つからないならば，それが S のシンメトリである．

素置換 (x_i, y_i) を試みたとき，x_i が現れる節を関連づけることができない可能性がある．この場合 σ はシンメトリではないため，バックトラックして他の可能性を探す．

3.4 Davis-Putnam 手続きへのシンメトリ適用

Davis-Putnam 手続きにおけるシンメトリの導入がすでに提案されており，証明木の縮小化を実現している[5]．本研究ではこれを比較対象として用いる．

l を節集合 S 中に現れるリテラルとする．Davis-Putnam 手続きにおける分割規則によって，S 中で l を含むモデルが存在しないとき，S の任意のモデル M に対して l は偽でなければならない．性質1より l と l' がシンメトリであれば，l' を含むモデルも存在せず，S の任意のモデル M において l' も偽でなければならない．ここで l' の割当てに相当する木の枝をカットすることができ，その結果として節集合は縮小化される．このように各々のシンメトリリテラルは推論木からカットされる．n 個からなるシンメトリサイクルが存在すれば，リテラルを $n-1$ 回カットすることができる．

4. SATCHMOST

本章では，SATCHMO へのシンメトリ概念の導入と証明木の縮小化について述べる．節集合のサイズを減少するために単位・純リテラル規則を適用し，さらに負の単位リテラル規則を導入した．このシステムを **SATCHMOST**（SATCHMO with Symmetry Testing）と呼ぶ．

4.1 SATCHMO へのシンメトリ適用

SATCHMO は非ホーン節のヘッドで分岐を起こすため，ある非ホーン節のヘッドに現れるリテラルどうしの中でシンメトリを発見することで，推論ステップ数を減らすことができる．これにより SATCHMO の証明木が縮小される．

また，SATCHMO は推論を実行する際に各推論ステップで節集合に導き出した事実を付け加え，更新した節集合を次の推論ステップに引き渡し推論を行う．各ステップで証明木の削除を行うとき，節集合を更新するため，分岐前のステップで計算したシンメトリを用いて証明木を削除することはできない．そのため推論の各ステップでシンメトリ発見の計算を行う．

以下に SATCHMOST におけるシンメトリ発見アルゴリズムを示す．ただし，$l_1, \ldots, l_n$ は分岐対象である非ホーン節のヘッドに現れるリテラル，$|C|$ は節 C の長さを表す．

(1) 初期設定：$Sym = Sym1 = nil$；

(2) $l_1, \ldots, l_n$ の中から性質2のシンメトリの必要条件を用いてシンメトリの候補となるリテラル X，Y を選ぶ；
候補がなければ (7) へ；
$ClauseSet(X,Y) \leftarrow X$ または Y を含む節集合；

(3) $ClauseSet(X,Y)$ から X の現れる節 C_X と Y の現れる節 C_Y をそれぞれ1つずつ見つける；

(4) **if** $|C_X| = |C_Y|$ かつ C_X と C_Y の各リテラルを関連づけることにより素置換の積 σ' を求めることができる
then $Sym1 \leftarrow Sym1 \cdot \sigma'$；
$ClauseSet(X,Y) \leftarrow$
$ClauseSet(X,Y) - \{C_X, C_Y\}$；

(5) $ClauseSet(X,Y)$ の中に (3)，(4) の処理を行っていない節があれば (3) へ；

(6) **if** X と Y を関連づけることができている
then $Sym \leftarrow Sym \cdot (X,Y) \cdot Sym1$；
(2) へ；

(7) Sym における最大のシンメトリサイクルを見つけ，その中の1つのリテラルについてのみ推論を続ける；
他のリテラルは証明木よりカットする．

4.2 SATCHMO への単位リテラル規則導入

SATCHMOST では，シンメトリにより証明木を縮小するとき，推論の各ステップでシンメトリ発見の計算を毎回行わなくてはならない．しかし推論ステップが進むに従って分岐したモデル候補の数が増大し，シンメトリ発見の計算が増えて処理の効率が悪くなる．そこで本研究では，新しく分岐で得られたアトムを単位節として節集合に付け加えるのではなく，そのアトムに沿って単位リテラル規則を適用し節集合を削除する方法を導入する．

たとえば，SATCHMO では，$(true \rightarrow a \vee b.)$ を処理する場合，次のように a を付け加えて節集合 S を S' に更新する：

S：	$\Rightarrow$	S'：
$true \rightarrow a \vee b.$		$a.$
$true \rightarrow c \vee d.$		$true \rightarrow a \vee b.$
⋮		$true \rightarrow c \vee d.$
$bottom \leftarrow a \wedge d.$		⋮
⋮		$bottom \leftarrow a \wedge d.$
		⋮

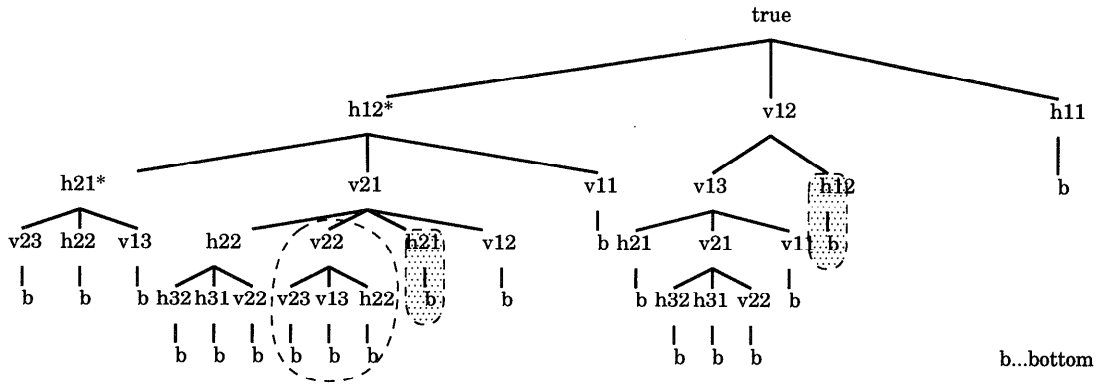

図3 checkerboard and dominos 問題の証明木
Fig. 3 Proof tree of checkerboard and dominos problem.

これに対し，単位リテラル規則を導入したSATCHMOで $(true \rightarrow a \vee b.)$ を処理する場合，a を付け加えた後，単位リテラル規則により，節 $(true \rightarrow a \vee b.)$ と負に現れている a を他の節より削除する：

S：	$\Rightarrow$	S'：
$true \rightarrow a \vee b.$		$a.$
$true \rightarrow c \vee d.$		$true \rightarrow c \vee d.$
⋮		⋮
$bottom \leftarrow a \wedge d.$		$bottom \leftarrow d.$
⋮		⋮

4.3 純リテラル消去戦略

さらに節集合の縮小のためにSATCHMOに純リテラル消去戦略を導入する．

純リテラル規則を使用する対象は，単位リテラル規則によって縮小した節中のアトム A である．縮小したことによって A が他に正に現れないならば，節集合から負に A が現れる節を削除する．

SATCHMOに単位・純リテラル規則を導入したもので，$(true \rightarrow a \vee b.)$ を処理する場合，次のように S を S' に更新する：

S：	$\Rightarrow$	S'：
$true \rightarrow a \vee b.$		$a.$
$true \rightarrow c \vee d.$		$true \rightarrow c \vee d.$
$bottom \leftarrow a \wedge c.$		$bottom \leftarrow c.$
$bottom \leftarrow c.$		$bottom \leftarrow d.$
$bottom \leftarrow a \wedge d.$		
$bottom \leftarrow b \wedge c.$		
$bottom \leftarrow b \wedge d.$		

上では，まず a を付け加え単位リテラル規則を適用する．この結果 $(true \rightarrow a \vee b.)$ が消去され，b が負にしか出現しなくなる．よって純リテラル規則により b が負に現れる節も削除する．

このように，単位・純リテラル規則の適用により節集合のサイズを縮小することができる．

4.4 負の単位リテラル規則の導入

4.2節では，単位リテラル規則を，推論により加えた正リテラル（アトム）に対してのみ用いていた．ここでは，負リテラルにも単位リテラル規則を適用し，節集合のさらなる縮小化を図る．

証明木のあるノード N において，N の下の部分木への推論がすべて $bottom$ へと導かれたならば，N の兄弟ノードの下の部分木における推論で不要な節やリテラルを節集合から削除することができる．ここでノード N の**兄弟ノード**とは，N の親ノードの子ノードで N 以外のものをいう．この操作を**負の単位リテラル規則**を適用するといい，4.2節の単位リテラル規則と区別する．

以下でcheckerboard and dominos問題（PUZ015-2.003）[4]を例に説明する．この問題をSATCHMOによって解くと**図3**のようなモデル生成木となる．

今，レベル（深さ）2で $h21*$ からのすべての推論が充足不可能であると分かるとする．節集合 $S \cup \{h12\}$ 中で $h21$ は真とはなり得ないので，このノードの兄弟ノード以降で S にモデルがあるとすれば $h21$ は偽でなければならない．したがって，このとき $h21$ の兄弟ノードである $v21$，$v11$ より下の部分木における節集合に節

$$h21 \rightarrow bottom.$$

を加えたと見なすことができ，負の単位リテラル規則により以下の処理を行う．

(1) 節集合 $S \cup \{h12\}$ 中でボディに $h21$ を含む節があれば，その節を削除する．

(2) 節集合 $S \cup \{h12\}$ 中でヘッドに $h21$ を含む節

があれば，$h21$ のみを取り除く．

(3) リテラル $h21$ とシンメトリ関係となるリテラルがあれば，そのリテラルについても同様に(1)，(2)の処理を行う．

これにより，このノード $h21$ の兄弟ノード以降での推論に $h21$ が出現することはなく，$h21$ の枝はカットできる（網掛け部分）．

同様にレベル1で $h12*$ からの推論がすべて *bottom* へと導かれたことにより，その兄弟ノード $v12$ より下に現れる $h12$ の枝はカットできる（網掛け部分）．

つまりこの方法は，各ノードで *bottom* が導かれたリテラルの兄弟とその子孫に対して効果がある．このように節集合のサイズを減少させることで，網掛けで囲った部分がカットされ証明木が縮小する．また，点線で囲った部分の枝 $v22$ は，左部の $h22$ と対称的であることからカットできる部分を示す．したがって，シンメトリを発見し負の単位リテラル規則を導入することによって，証明木の葉の数を19から14に減少することができる．

5. 実験結果

TPTP（1000の定理証明問題集）[4]を用いて，各証明木による効率に関する比較検討を行う．本実験はSPARCstation 20/71上で行った．

5.1 命題変数の組合せ問題（SYN001）

命題変数の組合せ問題は，長さ n で n 種類のリテラルからなる節で，すべての正リテラルと負リテラルの組合せを考え，その節集合の充足可能性を判定する問題である[4]．**表1** に証明時間を示す．また以下の表すべてにおいて，次の略記法を用いる．

SAT：SATCHMO
SAT+S：(SAT)+symmetry
SAT+S+up：(SAT+S)+単位・純リテラル規則
SATST：SATCHMOST
=(SAT+S+up)+負の単位リテラル規則
()：証明木の葉の数

表1において，証明木の葉の数に注目する．SATと比較すると，$n=6$ のときSAT+S，SAT+S+up，SATSTはそれぞれ葉の数を $\frac{1}{180}$，$\frac{1}{180}$，$\frac{1}{720}$ にそれぞれ大幅に減少している．しかし $n=1\sim6$ のときSAT+SやSAT+S+upは葉の数をおさえたにもかかわらずSATよりも処理速度が遅い．これはシンメトリ発見にかかる負荷が大き過ぎるために，全体処理時間に影響を及ぼしていると考えられる．シンメトリの導入は失敗かと思わせるが，$n=7$ 以降でこれらはSATよりも高速であり，シンメトリ発見効果が現れている．特にSATSTは葉の数をすべての実験で1におさえて最も効果的に推論を行っている．

表1 命題変数の組合せ問題の実行時間
Table 1 Results on all signed combinations of some propositions.

n	SAT	SAT+S	SAT+S+up	SATST
4	0.16 (24)	0.22 (2)	0.21 (2)	0.12 (1)
5	1.96 (120)	2.0 (3)	1.99 (3)	0.95 (1)
6	25.12 (720)	42.64 (4)	39.79 (4)	15.54 (1)
7	1901.28 (5040)	1322.19 (5)	1300.93 (5)	387.75 (1)

5.2 シンメトリがない節集合

前節では節集合が大きければ大きい問題であるほど，シンメトリ発見計算にかかる負荷があってもシンメトリサイクルを利用して推論した方が効率が良かった．ではシンメトリサイクルのまったくない節集合における証明時間についてはどうだろうか？ **表2** にその結果を示す．比較として，単位・純リテラル規則導入だけの効果を見るためにSAT+upを加えた．

やはり，SAT+SとSAT+S+upはシンメトリ発見に時間を費やすためSATより全体的に性能が落ちる．しかし負の単位リテラル規則を導入しているSATSTは節集合を縮小しているのでSAT+SとSAT+S+upよりもやや回復しており，シンメトリ発見にかかる負荷をおさえていると考えられる．むしろ，SATより良い結果となっているものもある．またSAT+upにおいては，すべての問題に関して最も高速に解くことができた．このことから，シンメトリがない問題に対しても，**単位・純リテラル規則をSATCHMOに導入することは効果的である**と考えることができる．

5.3 pigeon-hole問題（MSC007）

この問題においては，他の定理証明器との比較を行う．**表3** の測定結果が示すように，SATCHMOSTの優位性を検証することができた．

1時間以内に解くことができる n の値は，DPは8，DP+Sは16，SATは8，SATSTは27までであった．SATSTが最も高速に証明していることが分かる．

6. 考察

命題変数の組合せ問題では，シンメトリ発見は単位・純リテラル規則の併用で効果を最大限に発揮し，さらに負の単位リテラル規則の導入によって証明木を縮小して効率良く証明していることが分かる．

シンメトリサイクルのまったくない問題では，シンメトリ発見の計算にかかる負荷は大きいが，負の単位リテラル規則の導入によって節集合のサイズを縮小し，その負荷をおさえることができた．これによりシ

表2 シンメトリサイクルがない節集合の実行時間
Table 2 Results on problems with no symmetry cycle.

問題	*dep*	SAT	SAT+S	SAT+up	SAT+S+up	SATST
PUZ9-1	4	0.05 (6)	0.08 (6)	0.03 (6)	0.07 (6)	0.05 (6)
PUZ16-2	4	0.08 (10)	0.18 (10)	0.07 (10)	0.16 (10)	0.13 (10)
PUZ30-2	7	0.71 (65)	3.3 (65)	0.40 (65)	1.96 (65)	0.48 (65)
SYN93-1	9	0.29 (16)	0.52 (16)	0.32 (16)	0.56 (16)	0.44 (16)
SYN97-1	9	0.31 (16)	1.06 (16)	0.69 (16)	1.16 (16)	0.91 (16)
SYN98-1	15	2.78 (128)	5.75 (128)	3.29 (128)	5.88 (128)	4.52 (128)

SAT+up：SATCHMO+単位・純リテラル規則
dep：証明木の最大深さ

表3 pigeon-hole問題の実行時間
Table 3 Results on pigeon-hole problem.

n	DP	DP+Sym	SAT	SATST
3	0.02	0.07	0.02 (6)	0.1 (3)
4	0.16	0.22	0.11 (33)	0.19 (6)
5	1.25	0.68	0.88 (196)	0.51 (10)
6	8.47	2.04	6.62 (1305)	1.31 (15)
7	62.00	5.67	61.66 (9786)	1.97 (21)
8	470.95	14.27	648.03 (82201)	4.18 (28)

DP：Davis-Putnan手続き
DP+Sym：Davis-Putnan手続き+symmetry
n：pigeon数

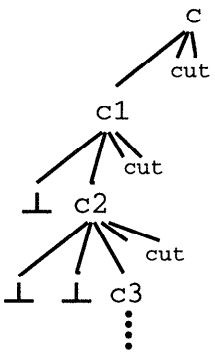

図4 pigeon-hole問題の証明木
Fig. 4 Proof tree of pigeon-hole problem.

ンメトリの有無にかかわらず，単位・純リテラル規則のSATCHMOへの適用が有効であるといえる．

pigeon-hole問題では，SATCHMOSTが，Davis-Putnan手続き，それにシンメトリを導入したもの，SATCHMOのいずれよりも群を抜いて高速であった．なぜこの問題に対して高速であるのかを以下で詳しく説明する．

この問題では，各推論ステップで分岐する非ホーン節のヘッド中にシンメトリサイクルを発見することができ，次の枝で *bottom* を導出しないすべてのリテラルがシンメトリサイクルをなす．n pigeonの場合，1ステップ目では，非ホーン節のヘッド中のリテラルは $n-1$ 個存在する．その中には次の枝で *bottom* を導くリテラルが存在しないため，$n-1$ 個のシンメトリが存在する．よって証明木の $n-2$ 個の枝をカットすることができる（残りの1個の枝は次の枝で *bottom* を導出する）．2ステップ目では $n-3$ 個の枝をカットすることができ，n ステップ目ではすべての枝から *bottom* を導き出し終了する（図4）．

シンメトリを利用しない場合，この問題を解く計算量は指数オーダーとなる．しかしシンメトリを利用すると，図4のようにシンメトリを持つ枝は残りをすべてカットし，$O(n^2)$ で解くことができる．さらに単位・純リテラル規則，負の単位リテラル規則を用いたSATCHMOSTについては，推論空間を縮小しながら推論を行うため，各ステップのシンメトリ発見時間を短縮することができる．

以上により，SATCHMOSTにおいては，各ステップでつねにシンメトリを含むような問題に関して劇的な効果をあげることが分かる．

7. まとめと今後の課題

本研究ではボトムアップ定理証明器SATCHMOにシンメトリの概念，単位・純リテラル規則，負の単位リテラル規則を導入した．シンメトリの発見は証明木を縮小する効果，単位・純リテラル規則の適用は節集合のサイズを減少する効果，負の単位リテラル規則の適用は節集合のサイズを減少して証明木も縮小する効果がそれぞれある．

このようにSATCHMOを変形したバージョンであるSATCHMOSTでは，数多くのシンメトリサイクルを発見できる問題に関して，節集合を削減し証明木を縮小して推論するため，非常に良い結果を示すことができた．また，シンメトリサイクルがまったく存在しない問題に関しては，シンメトリ発見の計算にかかる負荷が大き過ぎるために処理速度が遅くなるという弱点があった．しかしこの問題は，単位・純リテラル規則，負の単位リテラル規則の適用によって，節集合

のサイズを縮小しシンメトリ発見の計算にかかる負荷をおさえることで解決することができた．

今後の課題としては，以下の3点があげられる．

- **シンメトリ発見のための計算をするかしないかを判定する方法の検討．**
  シンメトリ発見のための計算量は一般に大きい．よって何らかの方法により，効果的なシンメトリの発見が期待できないと分かるならば，発見に関わる計算をしないで済ます方がよい．しかしながら，SATCHMOSTはシンメトリを持つ問題においてはそれを利用してきちんと高速に解くことができる．よって，シンメトリの有無が分からない場合には，SATCHMOとSATCHMOSTを同時に別の計算機上で走らせればよいという考え方もできる．この考えに立てば本課題は重要ではない．
- **述語論理の問題への拡張．**
  SATCHMOSTの単位・純リテラル規則は基礎節にしか適用できない．これを述語論理式に拡張するためには単位導出を工夫する必要がある．
- **モデル生成への適用．**
  本論文では充足可能性の検査を行うことを目的としていた．充足可能である場合に，すべての極小モデルを求める問題に対してシンメトリを利用できるように本手法を拡張する予定である．このためには，シンメトリ情報を利用して $(l \sim l')$ のとき l を含むモデルがあるならば，l を l' に置換することで別のモデルを得る方法が考えられる．

謝辞 SATCHMOSTの最初の実現に携わった（株）トキメックの遠藤晃司氏に感謝します．

参考文献

1) Davis, M., Logemann, G. and Loveland, D.: A Machine Program for Theorem Proving, *Comm. ACM*, Vol.5, No.7, pp.394–397 (1962).
2) Manthey, R. and Brys, F.: SATCHMO: A theorem prover implemented in Prolog, *Proc. 8th International Conference on Automated Deduction*, Lecture Notes in Computer Sience, Vol.310, pp.415–434 (1988).
3) Fujita, H. and Hasegawa, R.: A Model Generation Theorem Provers in KLI using Ramified-stack Algorithm, *Proc. ICLP'91*, pp.535–548 (1991).
4) Suttner, C.B. and Sutcliff, G.: The TPTP Problem Library (TPTP v1.2.1-TR Date 11.8.96), Technical Report 96/6, Department of Computer Science (1996).
5) Benhamou, B. and Sais, L.: Theoretical Study of Symmetries in Propositional Calculus and Applications, *Proc. 11th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, Vol.607, pp.281–294 (1992).
6) Cubbada, C. and Mouseigne, M.D.: Variantes de l'Algorithme de SL-resolution avec Retenue d'Information, These de 3ème cycle, GIA, Marseille, Lumniy, France (1990).
7) Oxusoff, L. and Rauzy, A.: L'Evaluation Semantique en Calcul Propositionnel, These de 3ème cycle, GIA, Marseille, Luminy, France (1989).
8) Schütz, U. and Geisler, T.: Efficient Model Generatoion through Compilation, *Proc. 13th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, Vol.1104, pp.433–447 (1996).

(平成 9 年 3 月 3 日受付)
(平成 10 年 4 月 3 日採録)

坂井 朱美

1975年生．1997年豊橋技術科学大学工学部情報工学課程卒業．同年4月より東京大学大型計算機センター技官．

井上 克巳（正会員）

1959年生．1982年京都大学工学部数理工学科卒業．1984年同大学院工学研究科数理工学専攻修士課程修了．松下電器産業（株），（財）新世代コンピュータ技術開発機構，豊橋技術科学大学工学部情報工学系を経て，1997年より神戸大学工学部電気電子工学科助教授．京都大学博士（工学）．人工知能，論理プログラミング，計算機科学に関する教育研究に従事．人工知能学会，AAAI各会員．