畳込みと分岐補題を統合したモデル生成

松下, 慎
九州大学大学院システム情報科学府知能システム学専攻 : 修士課程

長谷川, 隆三
九州大学大学院システム情報科学研究院知能システム学部門

藤田, 博
九州大学大学院システム情報科学研究院知能システム学部門

越村, 三幸
九州大学大学院システム情報科学研究院知能システム学部門

https://doi.org/10.15017/1525440

出版情報：九州大学大学院システム情報科学紀要. 7 (1), pp.23-28, 2002-03-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第7巻　第1号　平成14年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.7, No.1, March 2002

畳込みと分岐補題を統合したモデル生成

松下　慎* ・ 長谷川隆三** ・ 藤田　博** ・ 越村三幸**

Integrating Folding-up and Splitting Lemmas into Model Generation

Makoto MATSUSHITA, Ryuzo HASEGAWA, Hiroshi FUJITA, and Miyuki KOSHIMURA

(Received December 14, 2001)

Abstract: MGTP(*Model Generation Theorem Prover*) is a first order theorem prover based on model generation method which tries to construct Herbrand models of a given problem. We have embeded the folding-up rule into MGTP. The folding-up rule is used to ignore duplicate subproofs. We also have embeded the splitting lemma rule into MGTP. The splitting lemma rule is used to prune redundant models. In this study, we integrate both folding-up and splitting lemma rules into MGTP in order to strengthen MGTP. We evaluate effects of the integration by proving some typical problems.

Keywords: Folding-up, Splitting lemmas, Contrapositive Horn clauses, Model generation, Automated theorem proving

1. はじめに

モデル生成法は，一階述語論理の定理証明法である．モデル生成法の証明は公理(正節)から始まり，推論規則(混合節)を適用して次々と定理(アトム)を生成していく過程を，目標とする定理(負節)が得られるまで続ける，ボトムアップ実行に基づいている．このモデル生成法に基づいた定理証明器として，MGTP(*Model Generation Theorem Prover*)が開発されている．

これまでにMGTPに対する拡張として，証明の依存性解析による冗長探索の削除[1]や分岐補題の抽出による極小モデル生成の効率化[2] の手法によって探索空間の刈込みを行ってきた．

前者は証明に寄与したリテラルを求めることで補題を生成し，部分証明の重複や証明に関連しないモデル拡張を削除する方法である．これをfolding-upと呼ぶ．

後者は相補分割(*complement splitting*)を強化した分岐補題[2]という概念を導入することによって，非極小モデルに至るモデル候補を早期に棄却する方法である．これをminimal model checkingと呼ぶ．

またこれらの手法とは別に，予め問題を変形することによって刈込みを行う手法も開発している[3]．これは，入力する節集合と共にそれらの対偶ホーン節を用いてモデル生成を行うものである．対偶ホーン節を用いることによって証明の分岐が減少し，またモデル候補の棄却を行い易くなることで探索空間を刈込むことが出来ると考えられる．

本論文では，folding-up とmininal model checkingの2つの手法を同時に取り入れたMGTPを提案する．これにより，1つの問題に対して2つの手法を同時に適用することが可能となる．したがって，より多くの問題について探索空間の枝刈りを行うことができる．実験の結果2つの手法を取り入れたMGTPは，各々の手法で得られる結果と同等以上の効果が得られることが確認された．また，それに加え対偶ホーン節を導入することによってより大きな刈込みができることも確認された．

本論文の構成は以下のとおりである．2章でMGTPの基本的な証明手続きを示す．3章では補題による重複証明の削除について説明し，4章で分岐補題と非極小モデルの棄却について紹介する．5章では対偶ホーン節について述べる．6章で関連リテラルを計算し，補題を生成する手続きを示す．7章では一方の手法のみを取り入れたシステムなどとの比較評価実験の結果を示す．最後に8章で結論を述べる．

2. モデル生成型定理証明系MGTP

MGTPは$A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m$ なる含意式表現の節の集合に対し，**Fig. 1** のような手続きでモデルの構成を試みる．ここで，A_i, B_jはリテラル，$\rightarrow$の左辺を前件，右辺を後件という．また，$m = 0$を負節(*negative clause*)，$m = 1$をホーン節，$m \geq 2$を非ホーン節という．

連言照合関数$CJM(u, M)$は，モデル候補Mとモデル拡張候補アトムu で充足される前件を持つ節の具体例を探し，ホーン節の後件の集合と非ホーン節の後件の集合の対を返す．$Simp\&Subsump(D, M)$では**Fig. 2** (b)に示す選言縮約と包摂テストを実行する．

モデル候補の棄却条件は，1) CJM関数において負節

平成 13 年 12 月 14 日受付

* 知能システム学専攻修士課程
** 知能システム学部門

```
M0 = ⟨φ, U0, D0⟩; M = {M0};
/* U0: (Consequents of Horn clauses) */
/* D0: (Consequents of non-Horn clauses) */
L1: while (M ≠ φ) do { M = ⟨M, U, D⟩ ∈ M;
  while (U ≠ φ) { U = U \ {u ∈ U};
    if (u ∉ M) /* Unit subsumption */ {
      M = M ∪ {u}; ⟨U', D'⟩ = CJM(u, M);
      U = U ∪ U'; D = D ∪ D';
      D =Simp&Subsump(D, M);
      when ( Model candidate rejected ) {
        M = M \ M; continue L1; } } }
  if (D = φ) { output MODEL(M);
    M = M \ M; continue L1; }
  else { d = (d1 ∨ ... ∨ dm) ∈ D;
    Mk = ⟨M, U ∪ {dk}, D \ {d}⟩;
    M = (M \ M) ∪ {Mk}(1 ≤ k ≤ m); } }
if (/* No model found */) output UNSAT;
```

Fig. 1　MGTP procedure.

$$\frac{a \in M \quad \neg a \in M}{\bot} \qquad \frac{a(\neg a) \in M \quad \neg a(a) \vee C \in D}{C}$$

(a) Unit refutation　　(b) Disj-simplification

Fig. 2　Inference rules in MGTP.

$S0 =$
$\{ \rightarrow a \vee b.$
$a \rightarrow .$
$b \rightarrow c. \}$

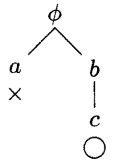

Fig. 3　Clause set S0 and its proof-tree.

に対して連言照合が成功する，2) **Fig. 2 (a)**の単位反駁が導かれる，3) **Fig. 2 (b)**の選言縮約によってD中のある選言が空となる，場合に成立する．

節集合が充足不能のときは，全てのモデル候補を棄却して有限終了する．一方，節集合が充足可能な場合，無限個の要素を持つモデルが存在しない限り，全てのモデルを生成することができる．

MGTP手続きに従い構成されるモデル候補集合$\mathcal{M}$は，木で表現することができる．これを“証明木”と呼ぶ．証明木の根から一つの葉に至る経路上の節点をラベル付けするアトムの集合が一つのモデル(候補)を表す．×印の葉は対応するモデル候補が棄却されたことを，○印の葉は対応するモデル候補がモデルであることを示す．

例えば，**Fig. 3**の節集合S0に対する手続き終了後の証明木は同図右のようになる．

$S1 =$
$\{ \rightarrow a \vee b.$
$a \rightarrow c.$
$b \rightarrow c.$
$c \rightarrow d \vee e.$
$c \wedge d \rightarrow .$
$c \wedge e \rightarrow . \}$

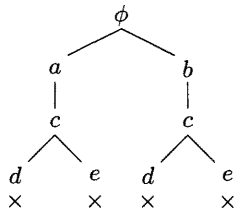

Fig. 4　Duplicate sub-proof trees.

$S1 =$
$\{ \rightarrow a \vee b.$
$a \rightarrow c.$
$b \rightarrow c.$
$c \rightarrow d \vee e.$
$c \wedge d \rightarrow .$
$c \wedge e \rightarrow . \}$

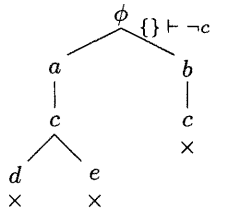

Fig. 5　Eliminating sub-proof tree by lemma.

3. 補題による重複証明の削除

Fig. 4の節集合$S1$に対してモデル生成法を用いると，同図右のような証明木が得られる．この証明木において，左右の部分証明木のリテラルc以下は全く同じ証明になっている．このように証明が分岐した場合それぞれの証明は全く独立に行われるため，各分枝で同じ証明を繰り返し行う可能性がある．このような重複証明があると証明木が余分に増えるとともに，モデル生成に時間がかかることになる．

ここで証明における関連リテラル集合(*relevant set*)を計算し，補題を生成することによって重複証明を削除することを考える．関連リテラルとは証明に寄与したリテラルである．関連リテラルの計算と補題生成の方法は5章において示す．**Fig. 4**の証明木の場合，左の部分証明において補題“$\{\} \vdash \neg c$”が生成され，右の部分木の証明で用いられる．右の証明木ではこの補題によって，リテラルcが現れたときに証明が棄却され，重複証明を回避することができる．この重複証明を削除した証明木を**Fig. 5**に示す．**Fig. 4**と比較すると，枝数が減少していることが分る．

4. 分岐補題と非極小モデルの棄却

選言$b \vee a$は$(b \wedge \neg a) \vee a$と等価である．したがって，**Fig. 6**のリテラルbの下に$\neg a$を加えてもよい．この加えられた$\neg a$を分岐仮定(*Splitting assumption*) と呼び，$[\neg a]$と表す．これは相補分割(*complement splitting*)規則の適用に相当している．この分岐仮定$[\neg a]$と節: $b \rightarrow a.$から導かれるaとに単位反駁を適用してこの枝が棄却される．

証明が終了すると，その状況に対応した関連リテラル

$S2 =$
$\{ \rightarrow b \vee a.$
$b \rightarrow a. \}$
$\rightarrow b \vee a.$
$\Rightarrow \quad \rightarrow (\neg a \wedge b) \vee a.$

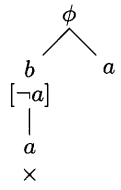

Fig. 6 Splitting assumption.

$S3 =$
$\{ \rightarrow a \vee b.$
$a \rightarrow c \vee d.$
$b \rightarrow a. \}$

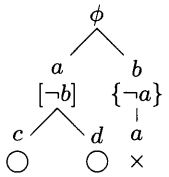

Fig. 7 Minimal model and splitting lemma.

$S4 =$
$\{ \rightarrow b \vee c.$
$b \rightarrow a.$
$b \wedge a \rightarrow .$
$\cdots \quad \}$

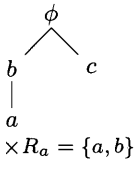

Fig. 8 Closed by negative clause and relevant set.

Lemma $\Gamma \vdash \neg a$ is already extracted at brother branches.

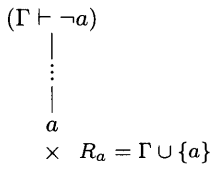

Fig. 9 Closed by lemma and relevant set.

の計算や補題の生成が行われる．証明が終了したとき枝が閉じているならば前章のように関連リテラルを求め補題を生成する．枝が開いているならば，分岐の弟枝にのみ有効な補題を生成する．この補題を特に分岐補題(*Splitting lemma*)と呼ぶ．この分岐補題によって兄枝に包摂されるモデル，すなわち非極小モデルを棄却することが可能になる．

例えば**Fig. 7**では，左枝の部分証明では枝が開いて証明が終了している．分岐補題"$\{\neg a\}$" が生成され右枝のリテラルbの親リテラルに加えられる．右枝ではaが現れたとき，分岐補題によって証明を棄却できる．これは右枝のbから導かれるモデルが非極小であることを表す．(aの下で得られたモデル$\{a, c\}, \{a, d\}$は，bの下で得られるべきモデル$\{b, a, c\}, \{b, a, d\}$を包摂する．)

また分岐補題により非極小モデルを早期に棄却することによって，探索空間の刈込みも可能であることは明らかである．

5. 対偶ホーン節の作成 : UR化

節のUR(*Unit Resulting*)化は与えられた節の各リテラルをそれぞれ後件とし，元の節と同意義なホーン節，すなわち対偶ホーン節を作成する．例えば次のような節cが与えられたとき，下の4つの節が新たに問題に加えられる．

入力節 c	$p \wedge q \rightarrow r \vee s.$
UR化された節	$p \wedge q \wedge \neg s \rightarrow r.$
	$p \wedge q \wedge \neg r \rightarrow s.$
	$q \wedge \neg r \wedge \neg s \rightarrow \neg p.$
	$p \wedge \neg r \wedge \neg s \rightarrow \neg q.$

入力節のUR化による利点は，MGTPにおけるモデル拡張はホーン節を優先して行うことから，枝分れが少ない効率的な証明が可能となることである．またモデル候補に負リテラルが導入されるので，モデル候補が矛盾する可能性が増す．これによって探索空間の刈込みを行う．

6. 関連リテラル集合と補題生成

3章と4章で述べたように，ある枝の証明が終了したとき関連リテラル集合の計算や補題の生成が以下のように行われる．

6.1 負節により閉じた場合の関連リテラル集合

証明枝が閉じたとき，その葉となるリテラルを棄却リテラルと言う．棄却リテラルをaとし，aに対する関連リテラル集合をR_aとする．またAは任意のリテラルの連言，またはリテラルの集合である．

負節により枝が閉じたとき，その前件リテラルが関連リテラル集合となる．すなわち，節 : $A \rightarrow .$によって枝が閉じたならば，その関連リテラル集合は$R_a = A$である．この様子を**Fig. 8**に示す．左枝が負節$b \wedge a \rightarrow .$により閉じたとき，リテラルaに対する関連リテラル集合$R_a = \{a, b\}$が求められる．

6.2 補題により閉じた場合の関連リテラル集合

補題によって枝が閉じたとき，補題と棄却リテラルが関連リテラル集合となる．すなわち，補題を$\Gamma \vdash \neg a$とすると，関連リテラル集合は$R_a = \Gamma \cup \{a\}$である．**Fig. 9**にその様子を示す．

6.3 分岐仮定により閉じた場合の関連リテラル集合

分岐仮定を導入した節$A \rightarrow (\neg a \wedge d) \vee a$ による分岐は$A \rightarrow d \vee a$による分岐の左枝に$\{d\} \vdash \neg a$なる補題を付加したものと考えられる．したがって，この下でaにより枝が閉じたときに求められる関連リテラル集合は$R_a = \{d\} \cup \{a\}$である．またこのときdに"used"がマークされる．この"used"のマークは補題生成のときに働く．

この様子を**Fig. 10**に示す．この場合，関連リテラル

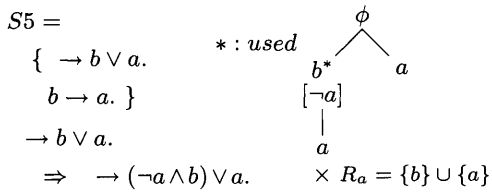

Fig. 10 Closed by Splitting assumption and relevant set.

集合は$R_a = \{b\} \cup \{a\}$となり，リテラルbに"used"がマークされる．

6.4 分岐補題により閉じた場合の関連リテラル集合

分岐補題で閉じた場合，その分岐補題が加えられた位置のリテラルと棄却リテラルが関連リテラル集合となる．例えば，**Fig. 7**の右枝の関連リテラル集合は$R_a = \phi \cup \{a\} = \{a\}$である．

6.5 枝が開いた場合の関連リテラル集合

開いた枝とは，1つ以上のモデルを持つ枝のことである．この枝に対する関連リテラル集合は$R = \phi$である．

6.6 証明枝の途中の関連リテラル集合

証明枝の途中のリテラルに対する関連リテラル集合は，子ノードに対応するリテラルをモデル候補に加えるために使用した節cとそのリテラルの関連リテラル集合から，節cに対する関連リテラル集合として求められる．関連リテラル集合の計算が証明木の下から上へあがるように行われるため，この手続きはliftingと呼ばれる．そこで，cに対する関連リテラル集合R_cは次のように計算される．

6.6.1 節cの後件リテラルが全て関連しているとき

リテラルaと関連リテラル集合Rについて$a \in R$ならばaは関連リテラルである．節cの後件のリテラルが全て関連リテラルであるとき，totally relevantという．

R_{d_1}，R_{d_2}は各々の枝の下で計算されたd_1，d_2に対する関連リテラル集合とする．また$d_i \in R_{d_i}(i = 1, 2)$とする．このとき節$c : A \rightarrow d_1 \vee d_2.$に対する関連リテラル集合は次のように求められる．

$$R_c = A \cup (R_{d_1} \setminus \{d_1\}) \cup (R_{d_2} \setminus \{d_2\})$$

後件がn個のリテラルの選言の場合も同様に求められる．**Fig. 11**にその例を示す．モデル拡張に節$c : a \rightarrow b \vee d.$が用いられたので，節$c$に対する関連リテラル集合は$R_c = R_a = \{a\} \cup (\{b\} \setminus \{b\}) \cup (\{d\} \setminus \{d\}) = \{a\}$ となる．

6.6.2 節cの後件リテラルが全て関連していないとき

節$c : A \rightarrow d_1 \vee \ldots \vee d_i \vee \ldots \vee d_n$において，$d_1 \sim d_{i-1}$は関連し，$d_i$が不関連($d_i \notin R_{d_i}$)のとき，この節$c$によるモデル拡張は冗長である．なぜならば，$d_i$が不関連であるならば，その枝が閉じた原因は$d_i$と全く無関係であり，し

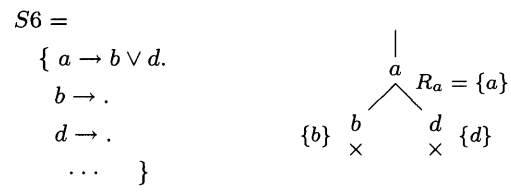

Fig. 11 Relevant set at lifting - totally relevant.

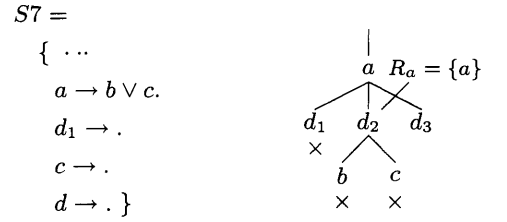

Fig. 12 Relevant set at lifting - irrelevant.

たがって節cによるモデル拡張がない場合においても同じ原因で枝は閉じるのである．そこで$R_c = R_{d_i}$とし，$d_{i+1} \sim d_n$の証明は行わない．これによって証明に関連していないモデル拡張を削除することができる．

Fig. 12に例を示す．このとき，d_2は関連リテラルではないため，d_3以下の証明は行われない．また，リテラルaの関連リテラル集合は$R_a = R_{d_2} = \{a\}$である．

6.7 補題の生成

リテラルaが関連リテラルのとき，補題を生成する．またこのとき，分岐仮定を使用して枝が閉じられたかを調べる．すなわち，

(1). aに"used"マークがないならば補題 $R \setminus \{a\} \vdash \neg a$ を生成する．
(2). aに"used"マークがあるならば補題は生成しない．

"used"マークがある場合，その枝は節の分岐に基づく仮定の下で閉じたのであり，このとき計算される関連リテラル集合にはその仮定が含まれる．これから補題を生成すると，この仮定が無効な枝に対してもその(本来無効な仮定を含む)補題が適用されることがある．したがって，"used"マークがある場合には補題は生成しない．

一方，aが関連リテラルでないならば，aの関連リテラル集合は$R_a = R$である．したがってRに対する操作は特に行わず，補題も生成しない．

6.8 分岐補題の生成

分岐の非最右枝が開いている場合，分岐補題を生成する．リテラルaにおける分岐補題の生成は次のように行われる．

(1). aに"used"マークがないならば分岐補題$\{\neg a\}$ を親リテラルの位置に生成する．この分岐補題は弟枝

$S8 =$
$\{ \cdots$
$t \to a \vee b \vee c.$
$b \to a.$
$\cdots\}$

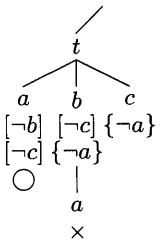

Fig. 13 Opened branch and splitting lemma.

$S9 =$
$\{ \to a \vee b.$
$a \to b \vee c.$
$b \to a. \}$

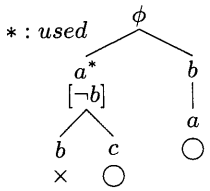

Fig. 14 example for "used".

のみに有効である.

(2). aに"used"マークがある場合，補題は生成しない

Fig. 13 に例を示す．最左枝が開いたときaは"used"ではないため分岐補題$\{\neg a\}$が生成される．この分岐補題はaの親であるtの位置に加えられる．図中ではb, cに有効であるため，その下に分岐補題を表わす.

また**Fig. 14** は"used"マークがある場合，すなわち分岐仮定によって閉じた枝がある場合である．図中の最左枝においてbと分岐仮定$[\neg b]$によって枝が閉じたとき，aに"used"がマークされる(図中では$*$で表わす)．よってリテラルaに対応する補題($\{\} \vdash \neg a$) や分岐補題($\{\neg a\}$)は生成されない.

7. 性能評価，実験

Minimal model checkingとfolding upの2つの手法を同時に持つMGTP(*mchk-fup*)を一方の手法のみを持つMGTP(minimal model checking: *mchk*, folding up: *fup*)や相補分割のみを持つMGTP(*CS*)と比較する．またmchk-fup版MGTPに対偶ホーン節を導入した場合(*m-f+UR*)の結果も示す.

結果を**Table–1** に示す．実験に使用した問題は次のとおり.

ex1

$$S_N = \{\to p_{i1} \vee p_{i2} \vee p_{i3} \vee p_{i4} \vee p_{i5} \vee p_{i6} \vee p_{i7} \vee p_{i8} \vee p_{i9} \vee p_{i10} \mid i = 1, \ldots, N\}$$

MM-SATCHMO[4]のベンチマーク例題からの引用である．証明木は各分岐が分岐数10の平衡木であり，生成されるモデルは全て極小である.

ex2

$$S_N = \{p_{i+1} \to p_i \vee q_i \vee r_i,\ r_i \to q_i,\ q_i \to p_i \mid i = 1, \ldots, N\} \cup \{\to p_N\}$$

証明木は右に行くほど深く葉数も増大する非平衡木である．最左枝の1本のみが極小モデルであり，その他のモデルは全てこれに包摂される．分岐補題を用いれば，これらは全て棄却できる.

ex3

$$S_a = \{\to a_i \vee b_i \vee c_i \vee d_i \vee e_i \mid i = 1, \ldots, 4\} \cup \{a_3 \to a_2,\ a_4 \to a_3,\ a_1 \to a_4\}$$

Niemeläの論文[5]からの引用である．非極小モデルが全て分岐補題によって棄却できる.

no50-100-1

全ての節が3つのリテラルからなる3-SAT問題である．充足不能であり分岐補題は生成されない.

yes50-100-1

同じく3-SAT問題であるが，こちらの問題は充足可能である.

syn009-1

TPTPライブラリ[6]から引用した一階述語論理の例題である．充足不能であり分岐補題は生成されない.

ex1は生成されるモデルが全て極小であるため分岐補題の効果が現れない．また全ての枝が成功枝であるため補題が生成されることもない．よって，どのMGTPでも証明木の刈込みは起こらない.

ex2，ex3は非極小モデルが全て分岐補題で棄却される問題であるため，mchk版MGTPが良い効果をあげることができる.

一方 no50-100-1，yes50-100-1，syn009-1 は fup 版MGTPが効果的であった．すなわち補題による刈込みが効果的な問題である.

mchk-fup版MGTPの結果に注目すると，fupが有効な問題に関して，同時にmchkが有効な問題に関して，それらの結果と同等な効果が得られている．また，問題ex3では2つの手法を同時に用いたことで，失敗枝(failed branches)がより減少するという結果が得られた.

さらに対偶ホーン節を導入することにより，no50-100-1やyes50-100-1の結果に現れているように，探索空間をさらに刈込むことが出来た.

ただしex1のように，どの手法によっても証明木の刈込みを行うことが出来ない問題の場合，関連リテラル集合の計算などのオーバーヘッドが生じ，証明時間が延びてしまうことがある.

8. ま と め

本稿では，証明の依存性解析による冗長探索の削除と分岐補題の抽出による極小モデル生成の効率化という既

Table-1 Performance comparison.

Problem	CS	mchk	fup	mchk-fup	m-f+UR
ex1	100000	100000	100000	100000	100000
(N=10)	0	0	0	0	0
	140	401	241	600	601
ex2	1594323	1	1594323	1	1
(N=14)	0	26	0	26	26
	3515	0	5508	10	10
ex3	501	341	501	341	341
	0	96	0	12	12
	10	10	20	20	20
no50-100-1	0	0	0	0	0
	197573	128246	41	41	26
	951	771	20	20	20
yes50-100-1	1	1	1	1	1
	383757	169786	750	736	683
	1893	912	120	110	120
syn009-1	0	0	0	0	0
	19683	19683	3	3	3
	100	120	20	20	20

top: No. of models, middle: No. of failed branches,
bottom: time (msec).

存の2つの手法を統合したMGTPを提案した．それぞれの手法における具体的な計算法を紹介し，実験評価によって2つの手法のどちらか一方が有効であれば，本MGTPもそれと同等以上の効果を得ることができることを確認した．

また，それらとは異なるアプローチから刈込みを狙う対偶ホーン節の概念を導入した．これによって，さらなる刈込みができることを確認した．

しかしながら，どの手法も効果のない問題に対しては証明木の刈込みが出来ないばかりか，補題等の計算によるオーバーヘッドによって証明時間が延びてしまう．このオーバーヘッドを如何に少なくするかが今後の課題である．

参 考 文 献

1) 越村三幸; 長谷川隆三: “証明の依存性解析による定理証明の冗長探索の削除,” 人工知能学会誌, Vol.15, No.6, (2000), 1064–1073.
2) 長谷川隆三; 藤田 博; 越村三幸: “分岐補題の抽出による極小モデル生成の効率化,” 人工知能学会論文誌, Vol.16, No.2, (2001), 234–245.
3) 松下 慎: “対偶ホーン節を用いたモデル生成法の探索空間の刈り込み,” 九州大学電気情報工学科卒業論文, (2001)
4) Bry, F.: http://www.pms.informatik.uni-muenchen.de/software/MM-SATCHMO, (1999).
5) Niemelä, I.: “A Tableaux Calculus for Minimal Model Reasoning,” *LNAI 1071*, (1996), 278–294.
6) Sutcliffe, G., Suttner, C. and Yemenis, T.: The TPTP Problem Library, *Proc. CADE-12*, (1994), 252–266.