CMGTPへの事後関連性検査と畳込み機構の組込み

館林, 剛史
九州大学大学院システム情報科学研究科知能システム学専攻 : 修士課程

越村, 三幸
九州大学大学院システム情報科学研究科知能システム学専攻

長谷川, 隆三
九州大学大学院システム情報科学研究科知能システム学専攻

https://doi.org/10.15017/1513731

出版情報：九州大学大学院システム情報科学紀要. 4 (2), pp.151-158, 1999-09-24. 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
システム情報科学研究科報告
第4巻　第2号　平成11年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.4, No.2, September 1999

CMGTPへの事後関連性検査と畳込み機構の組込み

舘林剛史* ・ 越村三幸** ・ 長谷川隆三**

Embedding Delayed Relevancy Testing and Folding-up into CMGTP

Tsuyoshi TATEBAYASHI, Miyuki KOSHIMURA and Ryuzo HASEGAWA

(Received June 21, 1999)

Abstract: We present a method embedding two mechanisms, which eliminate redundant inferences, into CMGTP. One is *delayed relevancy testing* which eliminates unnecessary subproofs by calculating relevancy to the proof. Another is *folding-up* which eliminates a duplicate subproof by using lemmas extracted from the proof. These two mechanisms are achieved by extracting literals that have contributed to the proof. We have implemented the method in Java and evaluated its effects for some typical problems taken from the TPTP problem library.

Keywords: Model generation theorem proving, Relevancy testing, Folding-up, Java implementation

1. はじめに

モデル生成法に基づく定理証明システムMGTP(Model Generation Theorem Prover)に負制約処理を加えたCMGTP(Constraint MGTP)が開発され，有限領域の組み合わせ問題を解くのに効果を上げている[1),2)]．CMGTPでは，モデル生成法に忠実に問題のモデルを構築しようとするため，証明には無関連な推論を行ったり，証明の分岐後に同じ証明を重複して行ってしまう，という効率上の問題点を原理的に抱えている．

本論文ではこの問題点を解決するために，証明に無関連な推論の除去と補題による重複証明を回避するための機構について述べる．前者は証明濃縮(proof condensation)[9)]，後者は畳込み(folding-up)法[7)]という名でタブロ法[5)]に基づく証明法の冗長性除去手法として知られる機構である．

証明濃縮では，証明が分岐しそのある分枝の証明が終了した時点で，その分岐自体が有効であったかどうかの判定を終了した分枝の証明を解析することで行う．分岐が有効でなかったと判定されると証明が未了の残りの分枝の証明を行わない．これにより探索空間の刈り込みが行える．証明濃縮は，関連性検査[8)]と呼ばれる探索空間の刈り込み手法とも関係[6)]しているので，本論文では，事後関連性検査と呼ぶことにする．

畳込み法は，終了した分枝の証明を解析することにより補題を抽出する．この補題を証明未了の部分に適用することにより，証明の重複を回避する．事後関連性検査及び畳込み法のいずれも証明の依存性の解析を行うことで実現できる．

以降の節では，まずCMGTPのアルゴリズムにふれた後，事後関連性検査と畳込み法について概説する．そして，両者を埋め込んだCMGTPのアルゴリズムを提示する．最後に本アルゴリズムのJavaによる実装の定量的評価を与える．

2. CMGTP

本論文を通じて，節は次のように含意形式 $A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m$ で表現される．ここで，$A_i(1 \leq i \leq n)$ 及び $B_j(1 \leq j \leq m)$ はリテラルである．$\rightarrow$の左側を前件部，右側を後件部という．またリテラルLの相補リテラルを$\overline{L}$と表す．$n = 0$のとき，前件部を特に$true$と書き，正節と呼ぶ．一方，$m = 0$のとき，後件部を特に$false$と書き，負節と呼ぶ．それ以外の節$(m \neq 0, n \neq 0)$は混合節と呼ばれる．

リテラルの連言 $A_1 \wedge \ldots \wedge A_n$ が基礎リテラルの集合Mに基礎代入σのもとで充足されるとは，$\forall i(1 \leq i \leq n) A_i\sigma \in M$であることをいい，リテラルの選言 $B_1 \vee \ldots \vee B_m$ が基礎リテラルの集合Mに基礎代入σのもとで充足されるとは，$\exists i(1 \leq i \leq m) B_i\sigma \in M$であることをいう．同じ基礎代入$\sigma$のもとで，前件部が充足され，後件部が充足されない節を違反(violated)節と呼ぶ．

Fig.1にCMGTPの証明手続きを示す．CMGTPは，与えられた節集合Sのモデル(真と解釈される基礎リテラルの集合)の構築を試みる．モデルの構築に成功すればSは充足可能(satisfiable)，失敗すれば充足不可能(unsatisfiable)である．

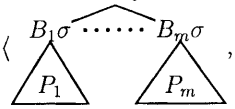
手続き$CMGTP$は，証明の対象となる節集合Sを受け取り，その注釈付き証明木を返す．注釈付き証明木は，通常の証明木に(どの節を用いて証明を行ったかを示す)証

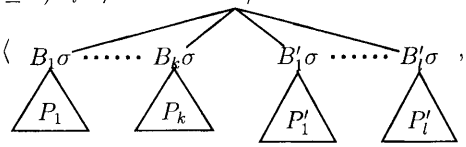
平成11年6月21日受付

* 知能システム学専攻修士課程
** 知能システム学専攻

procedure $CMGTP(S)$; /* Input(S): clause set */
(0) **return** $(MG(\emptyset, \emptyset, S))$;

procedure $MG(Mc, \mathcal{D}, S)$; /* Input(Mc): model candidate, ($\mathcal{D}$): model extending candidate */
(1) **if** there is a negative clause $(A_1 \wedge \ldots \wedge A_n \rightarrow false) \in S$ and a ground substitution σ such that $\forall i(1 \leq i \leq n) A_i\sigma \in Mc$ **then**
(2) **return** $\langle \bot, A_1\sigma \wedge \ldots \wedge A_n\sigma \rangle$; /* model candidate rejection by the negative clause */
(3) $\mathcal{D} = simplify(Mc, \mathcal{D})$; /* simplifying $\mathcal{D}$ with Mc */
(4) **if** $(\langle A_1\sigma \wedge \ldots \wedge A_n\sigma, \emptyset, \{B_1\sigma, \ldots, B_m\sigma\} \rangle \in \mathcal{D})$ **then**

return $\langle$  , $A_1\sigma \wedge \ldots \wedge A_n\sigma \rangle$ where each P_i $(1 \leq i \leq m)$ is in the form of $\langle \bot, \overline{B_i\sigma} \wedge B_i\sigma \rangle$

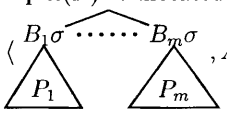
(5) $\mathcal{D} = \mathcal{D} \cup \{\langle A_1\sigma \wedge \ldots \wedge A_n\sigma, \{B_1\sigma, \ldots, B_m\sigma\}, \emptyset \rangle \mid (A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m) \in S$ and there is a ground substitution σ such that $\forall i(1 \leq i \leq n) A_i\sigma \in Mc\}$;
(6) **while** $(\mathcal{D} \neq \emptyset)$ {
(7) $\mathcal{D} = \mathcal{D} \setminus \{\langle A_1\sigma \wedge \ldots \wedge A_n\sigma, \{B_1\sigma, \ldots, B_k\sigma\}, \{B'_1\sigma, \ldots, B'_l\sigma\} \rangle\}$;
(8) **if** $\forall i(1 \leq i \leq k) B_i\sigma \notin Mc$ **then** /* model extension */
(9) **return** $\langle$  , $A_1\sigma \wedge \ldots \wedge A_n\sigma \rangle$

where $P_i = MG(Mc \cup \{B_i\sigma\}, \mathcal{D}, \mathcal{S})$ $(1 \leq i \leq k)$ and each P'_j $(1 \leq j \leq l)$ is in the form of $\langle \bot, \overline{B'_j\sigma} \wedge B'_j\sigma \rangle$

}
(10) **return** $\langle \top, \emptyset \rangle$; /* Mc is a model of S. */

procedure $simplify(M, \mathcal{D}) : \mathcal{D}_s$; /* Input($M$): a set of ground literals, ($\mathcal{D}$): model extending candidate */
Rewrite each element D $(= \langle Ante, \{B_1\sigma, \ldots, B_k\sigma\}, Closed \rangle) \in M$ according as the following rule (1) as much as possible, then **return** $\mathcal{D}$;
(1)if $\exists L \in M \exists i(1 \leq i \leq k)(\overline{L} = B_i\sigma)$, then rewrite D to $\langle Ante, \{B_1\sigma, \ldots, B_{i-1}\sigma, B_{i+1}\sigma, \ldots, B_k\sigma\}, \{B_i\sigma\} \cup Closed \rangle$

Fig.1 Proof procedure of CMGTP

procedure $Trans(P)$;
/* Input(P): annotated proof-tree */
if $P = \langle$  $, Ante \rangle$ **then**

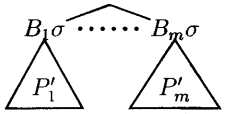
return 
where $P'_i = Trans(P_i)$ $(1 \leq i \leq m)$
if $P = \langle \bot, Ante \rangle$ **then return** $\bot$;
if $P = \langle \top, \emptyset \rangle$ **then return** $\top$;

Fig.2 Proof-tree transformation

明の履歴を付加したものである．この履歴は，次節で述べる関連リテラルの集合を定義する際に用いられる．注釈付き証明木から通常の証明木は**Fig.2**に示す変換手続きによって得られる．

手続きMG中，Mcは構成途中のモデルを表し，これをモデル候補と呼ぶ．Mcは，空集合$\emptyset$で初期化される((0)行目)．モデル生成手続きは，Mcに違反する節をSから見つけ出し，その節が違反節でなくなるようにMcを拡張していく手続きである．拡張の仕方が複数あれば，すべての拡張が試される．違反する節がなければ，McがSのモデルであることが分かり，Sは充足可能であると結論される((10)行目)．このときに手続きの値として返される注釈付き証明木の$\top$の部分がモデルがみつかったことを示している．

違反節を計算するために，まず，(ア)節の前件部がMcで充足しているかを検査し，基礎代入σで充足していれば，(イ)σで後件部が充足しているかを検査する．モデル拡張候補$\mathcal{D}$は，(ア)の検査を通過した節を蓄えておくためのものである．(ア)の処理は，(5)行目で行われ，(イ)の処理は(8)行目で行われている．$\mathcal{D}$の各元は三組で表現され，第一要素は違反節の前件部，第二要素は違反節の後件リテラルの集合，第三要素は空集合，で初期化される．

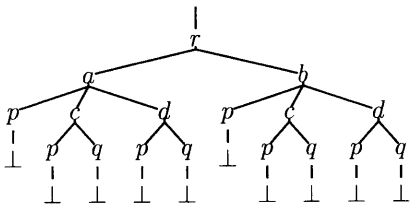


Fig.4 A normal proof-tree of $S1$

第一要素は手続き中で不変だが，第二要素と第三要素は，後に述べる$\mathcal{D}$の縮約手続き$simplify$により更新される．

違反節$A_1\sigma \wedge \ldots \wedge A_n\sigma \rightarrow B_1\sigma \vee \ldots \vee B_m\sigma$を違反節でなくすためには，$Mc$に$B_1\sigma \sim B_m\sigma$のいずれかを加えて拡張すればよい(モデル拡張)．それを行っているのが(9)行目のP_iの計算における$Mc \cup \{B_i\sigma\}$である．

負節の前件部がMcで充足している場合((1)行目)，この負節は違反節となるが，この負節が違反節ではないようにするMcの拡張は不可能なので，Mcは棄却される(モデル棄却(2)行目)．このときに手続きの値として返される注釈付き証明木の$\langle\frac{|}{\perp}$の部分がMcが棄却されたことを示している．節集合Sの注釈付き証明木の葉が全て$\langle\frac{|}{\perp}, Ante\rangle$とラベル付けされていれば$S$は充足不可能，そうでなければ充足可能である．

CMGTPとMGTPの違いは，$\mathcal{D}$の縮約手続き$simplify$があるかどうかである(3行目)．MGTPに縮約手続きを加えたものがCMGTPである．縮約手続きは，$\mathcal{D}$中の各元の第二要素のリテラルの内，モデル候補Mcの要素の相補リテラルと等しいものを，第二要素から取り除き，第三要素に加える．縮約の結果，第二要素が空集合になることもあるが，これは，Mcと$\mathcal{D}$が矛盾していることを表す．したがってこのとき((4)行目)，Mcは棄却される．

例 1 (注釈付き証明木I) 節集合$S1$

$C1: true \rightarrow r$　　$C2: r \rightarrow a \vee b$

$C3: r \rightarrow p \vee c \vee d$　$C4: r \rightarrow p \vee q$

$C5: p \rightarrow false$　　$C6: q \rightarrow false$

の注釈付き証明木の例を**Fig.3**に，通常の証明木の例を**Fig.4**に示す．

例 2 (注釈付き証明木II) 節集合$S2$

$C1: true \rightarrow t \vee p$　$C2: p \rightarrow q \vee s$

$C3: q \rightarrow r$　　$C4: s \rightarrow r$

$C5: t \rightarrow p$　　$C6: p \wedge r \rightarrow false$

の注釈付き証明木の例を**Fig.5**に，通常の証明木の例を**Fig.6**に示す．

3. 証明の依存性解析による冗長性の削除

もし，証明中に「この定理は，この定理とこの定理を用いて証明された」といった定理の因果関係(証明の依存性)を書き留めておけば，証明が終了した後，「実は，この場合分けは証明に不要であった」とか，「これだけの条件がそろえば，この定理が成り立つ」といったことを明らかにできる．

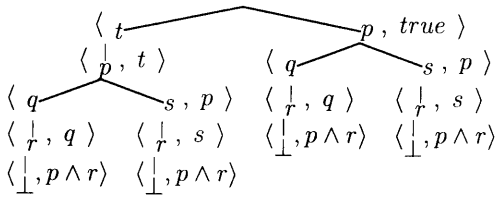


Fig.5 An annotated proof-tree of $S2$

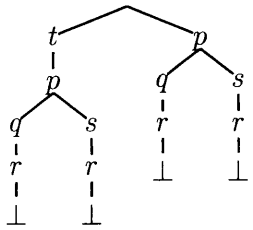


Fig.6 A normal proof-tree of $S2$

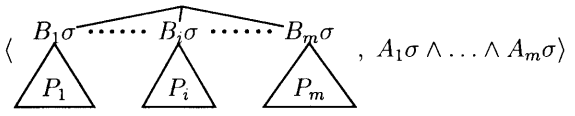


Fig.7 Model extension

前者の場合，不要と判明するのが，すべての場合を尽くした後であれば，後の祭りであるが，そうではなく，一部分の証明の後に判明すれば，残りの証明を削除できる．後者の場合は，補題を生成し，補題の適用によって重複した証明を取り除くことができる．

前者の機能のことを事後関連性検査，後者の機能のことを畳込み法，と言う．

3.1 事後関連性検査

証明に関連しないモデル拡張を削除する方法に 事後関連性検査 がある．モデル拡張が証明に関連しているかどうかの判定を行うには，関連リテラルを求める必要がある．ここで，関連リテラルは次のように定義される．

定義 1 (関連リテラル) 注釈付き証明木Pに対して，関連リテラルの集合$Rel(P)$は次のように再帰的に定義される．

1. $P = \langle\frac{|}{\perp}, A_1\sigma \wedge \ldots \wedge A_n\sigma\rangle$である場合，
   $Rel(P) = \{A_1\sigma, \ldots, A_n\sigma\}$.
2. $P = \langle\frac{|}{\top}, \emptyset\rangle$ である場合，$Rel(P) = \emptyset$.
3. Pが**Fig.7**で示す証明木であり，
   (a) $\forall i(1 \leq i \leq m) B_i\sigma \in Rel(P_i)$の場合，
   $Rel(P) = \cup_{i=1}^{m}(Rel(P_i) \backslash \{B_i\sigma\}) \cup \{A_1\sigma, \ldots, A_n\sigma\}$
   (b) $\exists i(1 \leq i \leq m) B_i\sigma \notin Rel(P_i)$の場合，
   $Rel(P) = Rel(P_{i_0})$
   (但しi_0は，$B_{i_0}\sigma \notin Rel(P_{i_0})$を満たす．)

部分証明Pが$\frac{|}{\top}$を含まなければ，$Rel(P)$はPを作るのに寄与したリテラルの内，証明木中Pより上方に現れるリ

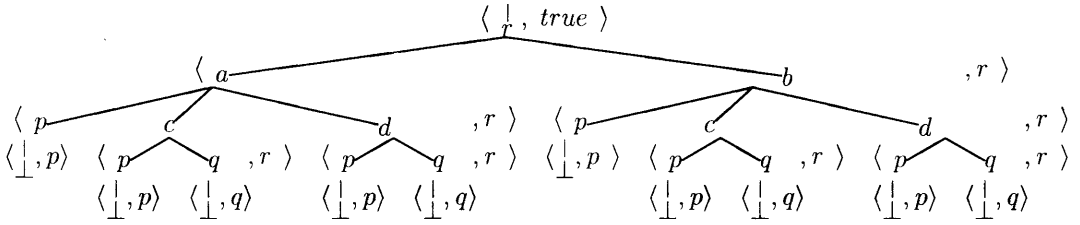


Fig.3 An annotated proof-tree of $S1$

テラルの集合となる．一方Pがモデルを見つけたことを示す$\frac{|}{\top}$を含む場合には，$Rel(P)$は空集合$\emptyset$になる．逆に$Rel(P)=\emptyset$であれば，Pは$\frac{|}{\top}$を含む．つまり，Pが$\frac{|}{\top}$を含むことと，$Rel(P)=\emptyset$であることは同値である．

証明に関連するモデル拡張は，関連リテラルを用いて次のように定義される．

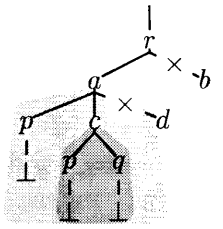
定義 2 (証明に関連したモデル拡張) 節 $A_1\sigma \wedge \ldots \wedge A_n\sigma \rightarrow B_1\sigma \vee \ldots \vee B_m\sigma$ によるモデル拡張が証明に関連しているとは，このモデル拡張を根とする**Fig.7**で示すような証明木において，$\forall i(1 \leq i \leq m) B_i\sigma \in Rel(P_i)$であることをいう．

証明に関連していないモデル拡張で，$\forall i(1 \leq i \leq m) Rel(P_i) \neq \emptyset$を満たすようなものは削除することができる．この削除が健全であるのは次の理由による．いま，節$A_1\sigma \wedge \ldots \wedge A_n\sigma \rightarrow B_1\sigma \vee \ldots \vee B_m\sigma$によるモデル拡張が証明に関連していないものとする．そうするとこのモデル拡張を根とする**Fig.7**で示す証明木において，$1 \leq i \leq m$を満たすあるiに対して$B_i\sigma \notin Rel(P_i)$となので，この証明木を$P_i$で置き換えても，証明が得られる．

さて，モデル拡張後の各分枝の部分証明$P_i(1 \leq i \leq m)$が左から順(昇順)に行われるとする．部分証明が終わるたびに$B_i\sigma \in Rel(P_i)$かどうかの検査を行い，もし，$B_i\sigma \notin Rel(P_i)$かつ$Rel(P_i) \neq \emptyset$ であれば，残りの部分証明$P_j(i < j \leq m)$を行う必要はなくなり，冗長な探索を削除できる．

例 3 (事後関連性検査による冗長な推論の削除) 例 *1*で示した証明木**Fig.4**は，証明に関連しないモデル拡張を含んでいる．$C1$でモデル拡張を行った時点で，$C2$, $C3$, $C4$の三つの節が違反節となる．違反節をこの順に用いてモデル拡張を行うと**Fig.4**のような証明木が選られる．この証明では，$C2$と$C3$ によるモデル拡張は証明に関連していない．

Fig.8に，**Fig.4**で示した証明から冗長な探索を削除した例を示す．内側の網掛け部分の証明の関連リテラル集合は$\{r\}$であり，これにcは含まれていない．これより，$C3$によるモデル拡張は証明に関連していないことがわかり，dの下の証明を削除できる．外側の網掛け部分でも同様のことがいえて，bの下の証明を削除できる．



SP	RL
inner	r
outer	r

SP: subproof
RL: relevant literals

Fig.8 Eliminating irrelevant model extensions

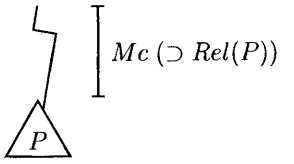


Fig.9 Grafted subproof

3.2 畳込み法 (Folding-up)

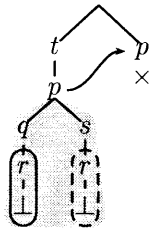
補題生成による重複証明の除去する方法として 畳込み法(folding-up) がある．いま，ある部分証明Pから関連リテラルの集合$Rel(P)(\neq \emptyset)$が計算されたものとする．そして別の部分の証明途中で，モデル候補Mcに対して$Rel(P) \subset Mc$であることが判明すれば，その部分の証明は，それ以上行なう必要はなく，完了してよい．なぜなら，Mcの下にPをつなげた図形が証明木となるからである(**Fig.9**)．

つまり，$Rel(P) \neq \emptyset$であれば$S \cup Rel(P)$は，(入力節集合Sの充足可能性に関わらず)充足不能である．このことを$S \cup Rel(P) \vdash \bot$もしくは，$Rel(P) \vdash_S \bot$と表記することにする．単位補題は，$Rel(P)$より次のようにして作られる．

定義 3 (単位補題) $Rel(P) = \{R_1, \ldots, R_n\}$ であるとき，$Rel(P) \setminus \{R_i\} \vdash_S \overline{R_i}\ (1 \leq i \leq n)$ を$Rel(P)$から得られる単位補題 *(unit lemma)* という．また，$Rel(P) \setminus \{R_i\}$をこの補題が適用できる文脈と呼ぶ．

単位補題は，証明手続きにおいて次のように利用される．まず，何らかの方法でモデル候補Mcが補題の文脈を満たす($Mc \supset Rel(P) \setminus \{R_i\}$)ことが分かっているものとする．そして，モデル拡張時のある後件リテラル$B_j\sigma$について，$B_j\sigma = R_i$であるとき，$B_j\sigma$による部分証明$MG(Mc \cup \{B_j\sigma\}, S)$を行なわない．

このように補題の適用にあたっては，モデル候補が補



SP	RL	UL
□	p, q	$\{p\} \vdash \neg q$
- -	p, s	$\{p\} \vdash \neg s$
outer	p	$\emptyset \vdash \neg p$

UL: unit lemma

Fig.10 Eliminating a duplicate proof

題の文脈を満たすかどうかの判定が必要となる．これには集合の包含関係の判定が必要となり，計算コストがかかる．そこで，本論文では，$Rel(P)$から得られるn個の単位補題の内，次のような高々一つの単位補題を利用する．

定義 4 (抽出される単位補題) **Fig.7**で示すような証明木において，部分証明木$P_i(1 \leq i \leq m)$から抽出される単位補題は，

$$\begin{cases} Rel(P_i) \setminus \{B_i\sigma\} \vdash_S \overline{B_i\sigma} & (B_i\sigma \in Rel(P_i)\text{のとき}) \\ \text{抽出しない} & (B_i\sigma \notin Rel(P_i)\text{のとき}) \end{cases}$$

である．

この補題の文脈は，P_iの兄弟証明$P_j(1 \leq j \leq m \wedge j \neq i)$のモデル候補に含まれる．つまり，この補題は$P_i$の兄弟証明で利用することが(文脈を検査しなくても)できる．さらにこの補題は，文脈が成り立たなくなる祖先まで持ち上げることができ，その祖先の子証明でも利用できる．

さて，容易に推察できるように，$Rel(P)$からは単位補題以外にも多くの補題を得ることができる．例えば，$Rel(P) \setminus \{R_i, R_j\} \vdash_S \overline{R_i} \vee \overline{R_j}\ (1 \leq i < j \leq n)$という補題も考えられる．このような補題は，$Rel(P) \vdash_S \bot$も補題と考えれば，$2^n$個ある．本論文では，単位補題以外の補題は扱わない．といのは，これらの補題によって逆に探索空間が広がる恐れもあるからである[7]．

例 4 (畳込み法による重複証明の回避) 例2で示した$S2$の証明木**Fig.6**は，$C1$でのモデル拡張により証明が分岐するが，左側のpの下と右側のpの下の部分証明は全く同じ証明である．

Fig.10に，この重複証明を削除した例を示す．図中の表は各部分証明の関連リテラルと抽出される補題を示す．外側の枠で囲んである部分証明(網掛け部分)からは単位補題$\emptyset \vdash \neg p$が抽出される．この補題の文脈は空集合なので，証明木の根まで持ち上げることができる．そして，右側のpに補題を適用しpから下の探索を刈り込むことができる．

4. 事後関連性検査と畳込み法のCMGTPへの組込み

事後関連性検査と畳込み法を組込んだCMGTPの手続きを**Fig.11**に示す．**Fig.1**のCMGTP手続きは，入力節集合Sの注釈付き証明木を返していたが，この手続きは，関連リテラルの集合Relと証明で抽出した単位補題の集合$OLem$の組を返す．手続きMGの引数として$ILem$が新たに加わっているが，これは，文脈がモデル候補Mcで満たされている単位補題の集合である．

事後関連性検査は，節点Nの下の証明にNのラベルLが寄与したかどうかを検査する．これには，Relが利用される．Relは補題を生成するのにも用いられる．また，畳込み法においては，生成した補題をどこまで持ち上げることができるかを判定する必要がある．各補題の文脈はこの判定を行う際に参照される．

以下では，**Fig.11**の手続きを順に説明していく．

- 1，2行目：負節によってモデル候補を棄却する場合は，負節の前件部リテラルの集合$\{A_1\sigma, \cdots, A_n\sigma\}$を$Rel$として返す．この時点では，補題はまだ生成されないので，$OLem$は$\emptyset$となる．
- 3行目：縮約はモデル候補Mcに加えて，補題集合$ILem$を用いても行われる．手続き**simplify**の(1)がMcによる縮約，(2)が$ILem$による縮約の処理を示している．$\mathcal{D}$の元D中のΛは，Dを縮約したMcの元と補題の文脈の和集合を表している．これは，関連リテラルの計算に用いられる．モデル候補要素による縮約では，そのモデル候補要素自身をΛに加える(手続き**simplify**の(1))．一方，補題$\Gamma \vdash L$による縮約では，文脈ΓをΛに加える(手続き**simplify**の(2))．
- 4行目：縮約によって，後件部が空になるような$\mathcal{D}$の元Dがあった場合は，そのモデル候補は棄却される．このときのRelは，そのD中の前件リテラル$A_1\sigma, \ldots, A_n\sigma$と$\Lambda$の和集合となる．
- 5行目：Mcに基礎代入σのもとで前件部が充足している節のσによる基礎代入例が$\mathcal{D}$に加えられる．
- 6行目から始まる**while**文は，後件部がMcで充足されないような節の基礎代入例を$\mathcal{D}$から見つけ出し，その節でモデル拡張を行う(9行目からの**for**文)．そのような節が見つからなければ，McはSのモデルであるので，Relは空集合となる(17行目)．
- 9行目から始まる**for**文では，モデル候補Mcを$B_i\sigma$で拡張して証明手続きが再帰的に繰り返される(10行目)．その結果は，Rel_i，$OLem_i$に格納され，事後関連性検査と補題生成に用いられる．
- 11行目：$OLem_i$の各補題が現在の節点より上に持ち上げることができるかを検査し，できるものは，Lem_iに格納される．Lem_iは続く兄弟節点の証明にも用いられる．
- 12，13行目：$B_i\sigma$が証明に関連していたかどうかの判定を行い，関連していなければ，続く兄弟節点の証明を行わない(12行目)．関連していれば，新しい補題$\Gamma_i \vdash \overline{B_i\sigma}$を$Lem_i$に加える(13行目)．

procedure $CMGTP(S)$;
  return $MG(\emptyset,\emptyset,\emptyset,S)$;
procedure $MG(Mc,\mathcal{D},ILem,S)$; /* Input($ILem$) : a set of unit lemmas available in Mc */
(1) **if** there is a negative clause $(A_1 \wedge \ldots \wedge A_n \rightarrow false) \in S$ and a ground substitution σ
    such that $\forall i(1 \leq i \leq n)A_i\sigma \in Mc$ **then**
(2) **return** $\langle\{A_1\sigma,\ldots,A_n\sigma\},\emptyset\rangle$; /* model candidate rejection by the negative clause */
(3) $\mathcal{D} = simplify(Mc,ILem,\mathcal{D})$; /* simplifying $\mathcal{D}$ with M and $ILem$ */
(4) **if** $(\langle A_1\sigma \wedge \ldots \wedge A_n\sigma,\emptyset,\Lambda\rangle \in \mathcal{D})$ **then return** $\langle\{A_1\sigma,\ldots,A_n\sigma\} \cup \Lambda,\emptyset\rangle$
(5) $\mathcal{D} = \mathcal{D} \cup \{\langle A_1\sigma \wedge \ldots \wedge A_n\sigma,\{B_1\sigma,\ldots,B_m\sigma\},\emptyset\rangle \mid (A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m) \in S$ and there is a
      ground substitution σ such that $\forall i(1 \leq i \leq n)A_i\sigma \in Mc\}$;
(6) **while** $(\mathcal{D} \neq \emptyset)$ {
(7) $\mathcal{D} = \mathcal{D} \setminus \{\langle A_1\sigma \wedge \ldots \wedge A_n\sigma,\{B_1\sigma,\ldots,B_k\sigma\},\Lambda\rangle\}$;
(8) **if** $\forall i(1 \leq i \leq k)B_i\sigma \notin Mc$ **then** {
(9) **for**$(i = 1; i \leq k; i{+}{+})$ { /* model extension */
(10) $\langle Rel_i, OLem_i\rangle = MG(Mc \cup \{B_i\sigma\},\mathcal{D},ILem,S)$;
(11) $Lem_i = \{(\Sigma \vdash L) \in OLem_i | B_i\sigma \notin \Sigma\}$; /* filtering out non-available lemmas */
(12) **if** $(Rel_i \neq \emptyset \wedge B_i\sigma \notin Rel_i)$ **then return**$\langle Rel_i, \bigcup_{j=1}^{i} Lem_j\rangle$; /* irrelevant */
(13) **if** $(B_i\sigma \in Rel_i)$ **then** { $\Gamma_i = Rel_i \setminus \{B_i\sigma\}$; $Lem_i = Lem_i \cup \{\Gamma_i \vdash \overline{B_i\sigma}\}$; } /* relevant */
(14) $ILem = ILem \cup Lem_i$;
  }
(15) **if** $(\exists i(1 \leq i \leq n)Rel_i = \emptyset)$ **then return** $\langle\emptyset, \bigcup_{i=1}^{m} Lem_i\rangle$;
(16) **else return** $\langle\bigcup_{i=1}^{m} \Gamma_i \cup \Lambda \cup \{A_1\sigma,\ldots,A_n\sigma\}, \bigcup_{i=1}^{m} Lem_i\rangle$;
  }
 }
(17) **return** $\langle\emptyset,\emptyset\rangle$;
procedure $simplify(M,Lem,\mathcal{D})$; /* Input($Lem$): a set of unit lemmas */
 Rewrite each element D $(= \langle Ante,\{B_1\sigma,\ldots,B_k\sigma\},\Lambda\rangle) \in \mathcal{D}$ according as the followin rules (1) and (2)
 as much as possible, then **return** $\mathcal{D}$;
 (1) if $\exists L \in M \exists i(1 \leq i \leq k)(\overline{L} = B_i\sigma)$, then rewrite D to $\langle Ante,\{B_1\sigma,\ldots,B_{i-1}\sigma,B_{i+1}\sigma,\ldots,B_k\sigma\},\Lambda \cup \{L\}\rangle$
 (2) if $\exists(\Gamma \vdash L) \in ILem \exists i(1 \leq i \leq m)(\overline{L} = B_i\sigma)$ then, rewrite D to
  $\langle Ante,\{B_1\sigma,\ldots,B_{i-1}\sigma,B_{i+1}\sigma,\ldots,B_k\sigma\},\Lambda \cup \Gamma\rangle$

Fig.11 CMGTP with delayed relevancy testing and folding-up

- 14行目：Lem_iは続く兄弟節点の証明に用いることができるので，$ILem$に加える．
- 15，16行目：全ての分枝の証明が終わったら，出力となるRelと$OLem$を計算して，証明を終わる．あるRel_iが空集合ならそこでモデルが見つかっているのでRelは空集合となる(15行目)．そうでなければ，関連リテラルの定義にしたがって，Relを求める(16行目)．

5. Javaによる実装と評価

前節で述べたアルゴリズムをJava言語を用いて実装した．実装にあたっては，一から作るのではなく，既にあるJava版CMGTP[2]に事後関連性検査と畳込み法を組み入れる方法をとった．

5.1 実装上の留意点

二つの手法を実現するために，新たに必要となる主な機能は，(1)前件が充足可能となった節の前件部基礎例の記憶，(2)関連性の判定，(3)補題を持ち上げる箇所の計算，(4)補題による選言縮約，(5)選言縮約に用いた補題の文脈の記憶，がある．**Fig.11**と関連付ければ，(1)は5行目，(2)は12と13行目，(3)は11行目，(4)と(5)は手続き$simplify$の(2)の手続き，に相当する．

(1)の前件部の記憶を単純な線形リストで行うと，(2)及び(3)の計算(**Fig.11**中，$B_i\sigma \in \Sigma$や$B_i\sigma \in Rel_i$の判定)に，ΣやRel_iの長さに比例した時間を要してしまう．前件リテラルを(証明木の葉から根に至る順に)ソーティングしておけば，ΣやRel_iの長さによらず，定数のコストでこれらの計算を行うことができる．例えば，$B_i\sigma \in Rel_i$の判定は，Rel_iを表しているリストの先頭リテラルと$B_i\sigma$の

一致性を判定すれば済む．Java版CMGTPにおいてリテラルの一致性の判定は，$B_i\sigma$の大きさによらず，Javaの==演算一回で行うことができる．

ソーティングは，モデル候補の要素にふられているリテラル番号を元に行うことができる．Java版CMGTPでは，リテラルをモデル候補に登録する際に登録順序に対して昇順に番号が割り当てられる．したがって，この番号を元に降順にソーティングを行えばよい．

ソーティングによって，(3)の計算は，さらに最適化できる．今，補題$\Gamma \vdash L$のΓが上記のようにソーティングされているとすると，この補題を持ち上げることのできる箇所は，証明木中Γの先頭リテラルが出現するところである．したがって，**Fig.11** のように証明木中を一段一段，補題を持ち上げることなく，一気に該当箇所まで持ち上げることが可能である．また，ソーティングによって和集合演算のコストも削減できる．

(4)の補題$\Gamma \vdash L$による選言縮約には，Java-CMGTPに元来ある相補リテラルの選言縮約の手法をそのまま流用できる．具体的には，pac, nacセル(論文[2]3.3.1を参照)に加えて$plac, nlac$をリテラルの属性に加える．$plac$はLが正リテラルのときに，$nlac$はLが負リテラルのときにonになる．

5.2 実験評価

定理証明のベンチマーク集TPTP[10](1.2.1版)からいくつか問題を選び，Java版CMGTPの性能評価を行った．評価にあたっては，KLIC言語によるMGTPの実装に事後関連性検査と畳込み法を組み入れたもの[3]との比較も行った．

Table–1に実験結果を示す．表中，JavaはCMGTPを，KLICはMGTPを指す．○欄は，事後関連性検査と畳込み法を組入れてあることを，×欄は，組入れていないことを表す．名前の終りにdomとついている問題は，値域限定[†1]を満たしていないことを表す．No. of branchesとNo. of nodesは証明木の枝数と節点数を表す．この量が問題の規模を表す物差しとなる．表中の例題は，PUZ018-2domを除いて充足不可能な問題である．

(1) 枝刈り効果

表中の全ての問題に対して，証明木の大きさが小さくなっており，探索空間の刈り込みに成功していることがわかる．実際，TPTP問題の内，証明に分岐を生ずるほとんどの問題で，枝刈り効果を確認している．PUZ018-2domのように，×の場合には時間切れで解けず，○で解けるようになった問題はPUZ問題(全57題)で4題，SYN問題(全479題)で36題あった．

(2) JavaとKLIC

概ね似たような枝刈り効果を示している．証明時間が100ミリ秒を超える問題では，Java版CMGTPの方がKLIC版MGTPより1.7～12.4倍程度高速である．一方，証明時間が100ミリ秒以下の問題では，CMGTPの方が低速になる．これは，Java版CMGTPで行われている前件部リテラルのソーティングと，実行時に行われるJavaバイトコードのJITコンパイルのオーバヘッドが原因であると考えられる．

6. おわりに

冗長な証明探索を刈り込む事後関連性検査と畳込み法を組込んだ定理証明システムCMGTPのアルゴリズムを示し，これをJava言語により実装した．前者は，証明に寄与したリテラル(関連リテラル)から証明に無関連な分岐(モデル拡張)を同定しこれを取り除くことにより，探索の刈り込みを行う．後者は，関連リテラルから補題を作り出し，この補題を利用して重複証明を回避する．

このように，二つの手法で刈り込める冗長な探索は別種のものであるが，いずれも関連リテラルの計算により実現される．この実現法の原理は簡潔であるが，その効果は大きい．また，本実現法はタブロ法に基づく定理証明システムにも適用可能である．

最近，仮説推論や故障診断といった応用の面から極小モデルの効率的な生成法が注目されている[4]．畳込み法による刈り込み手法は，モデル棄却の観点からの枝刈り手法なので，極小モデル生成にはそのままでは適用できない．今後，畳込み法と類似の手法が極小モデル生成に適用できないか考察していく予定である．

†1 節が値域限定を満たすとは，後件部に現れる変数が全て前件部に現れるこという．問題に含まれる全ての節が値域限定を満たしているとき，その問題は値域限定を満たすという．

参考文献

1) 白井 康之，長谷川 隆三，“モデル生成型定理証明システムによる制約充足問題の解決とその並列化”，電子情報通信学会論文誌，Vol. J80-D-II, No.1, pp.224–236 (1997)
2) 長谷川 隆三，藤田 博，“制約問題を解くためのモデル生成型定理証明系の新実装”，九州大学大学院システム情報科学研究科報告，Vol.4, No.1, pp.57–62 (1999)
3) 松本 誉史，越村 三幸，久保山 哲二，長谷川 隆三，“MGTPにおけるケース分割の重複削除手法とその評価”，九州大学大学院システム情報科学研究科報告，Vol.2, No.1, pp.75–80 (1997)
4) François Bry and Adnan Yahya, Minimal Model Generation with Positive Unit Hyper-Resolution Tableaux, In *Proc. Fifth Int. Workshop, TABLEAUX'96*, LNAI 1071, pp.143–159 (1996)
5) Melvin Fitting, *First-Order Logic and Automated Theorem Proving*, Springer (1996)
6) Miyuki Koshimura and Ryuzo Hasegawa, A Proof of Completeness for Non-Horn Magic Sets and Its Application to Proof Condensation, *Position Papers,*

Table 1 Performace results

PUZ010-1				PUZ018-2dom				SYN006-1dom			
Java		KLIC		Java		KLIC		Java		KLIC	
×	○	×	○	×	○	×	○	×	○	×	○
216150	5359	216150	4060	–	64	–	55	96	4	96	4
310772	12509	310772	9916	–	405	–	458	532	118	532	118
8427	1068	104180	5800	T.O.	2492	T.O.	6610	54	67	40	20
SYN009-1				SYN015-2dom				SYN036-3dom			
19683	3	19683	3	196	29	196	12	516	36	516	34
29526	14	29526	14	1800	175	2551	172	8323	257	8323	263
137	45	750	0	175	90	1110	90	228	72	1340	40
SYN036-4dom				SYN037-2dom							
1037	17	1146	20	2793	15	2868	15	No. of branches			
4087	61	4309	72	23788	129	23328	130	No. of nodes			
10315	191	31440	540	5281	94	9020	30	Time(msec)			

Time limit: 10 sec T.O.: time out

Environment Java: JDK1.2.1_03 Sun Ultra-10 333MHz(128MB)
KLIC : KLIC-3.002(gcc2.7.2.f.2) Sun Ultra-10 333MHz(128MB)

TABLEAUX'99, TR99-1, pp.101–115, Department of Computer Science, University at Albany (1999)

7) R. Letz, K. Mayr and C. Goller, Controlled Integration of the Cut Rule into Connection Tableau Calculi, *J. of Automated Reasoning*, Vol.13, pp.297–337 (1994)

8) D.W. Loveland, D.W. Reed and D.S. Wilson, SATCHMORE: SATCHMO with RElevancy, *J. of Automated Reasoning*, Vol.14, pp.325–351 (1995)

9) F. Oppacher and E. Suen, HARP: A Tableau-Based Theorem Prover, *J. of Automated Reasoning*, Vol.4, pp.69–100 (1988)

10) G. Sutcliffe, C. Suttner and T. Yemenis, The TPTP Problem Library, In *Proc. CADE-12*, pp.252–266, (1994)