一般化補題を利用したモデル生成法

清水, 純一
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

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

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

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

https://doi.org/10.15017/1515818

出版情報：九州大学大学院システム情報科学紀要. 8 (1), pp.55-59, 2003-03-26. 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8巻 第1号 平成15年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.8, No.1, March 2003

一般化補題を利用したモデル生成法

清水純一*・越村三幸**・藤田博**・長谷川隆三**

Model Generation Using Generalized Lemma

Junichi SHIMIZU, Miyuki KOSHIMURA, Hiroshi FUJITA and Ryuzo HASEGAWA

(Received December 13, 2002)

Abstract: We can prune unnecessary search spaces using lemmas. In order to enhance the pruning ability of lemmas, we introduce a method to generalize them. There may be many cases to which generalized lemma can be applied but non-generalized one can not. Then we propose a method that can reduce the generation of lemmas using lemma subsumption. We evaluated effects of the method by proving some typical problem.

Keywords: Model generation, Theorem prover, Lemma, Unit generalized lemma, Lemma subsumption

1. はじめに

モデル生成法は一階述語論理の定理証明法の一つであり，その名の示す通り節集合のモデルを生成するとともに，節集合の充足可能性の判定を行う．その証明は公理(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おり，人間にとって分かりやすい形で論理展開できるという特徴を持っている．モデル生成法は定理証明以外にもプログラム検証や論理プログラム，演繹データベース，非単調推論などに利用されており，その応用範囲は広い．

一方でモデル生成法は，証明に無関係な部分証明を行ったり，重複した部分証明を繰り返し行うなど，証明の効率を下げてしまう問題点も持ち合わせている．このうち重複した部分証明を回避する手法の一つに，補題(lemma)を用いた方法がある．補題とは，得られた部分証明木から抽出される負節のことであり，冗長性を省くために用いられる．しかし，従来の補題(固定的補題)では，全く同じ部分証明の回避しか行えず，補題数の増加によるメモリの負担増と，補題適用に時間がかかるという問題があった．そこで提案されているのが，より多くの部分証明を回避できる補題(一般化補題)を生成する方法[2) 5)]である．

本論文では，補題適用の時間短縮を目的として，一般化補題の自動生成法および補題の包摂検査について述べる．補題の包摂検査により，不要な補題を削除し補題数の増加を抑えることができる．

本論文では以下のように構成される．まず2章では，モデル生成法の概要を述べる．3章では，補題についての説明を行う．4章では，一般化補題の作成法について述べる．5章では，補題の包摂検査について説明する．6章でこれらの手法の効果を確認する．7章で結論とこれからの課題を述べてまとめる．

平成14年12月13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

2. モデル生成法

本論文では，モデル生成法における問題を表す入力節をすべて次のような形式で表す．

$$A_1 \wedge ... \wedge A_n \rightarrow C_1 \vee ... \vee C_m.$$

ここで，$A_1, ..., A_n$および$C_1, ..., C_m$は原始論理式(アトム) である．アトムには，定数や変数，関数などの種類がある．$\rightarrow$は含意(imply)を表す記号である．$\rightarrow$の左側を前件部，右側を後件部といい，前件部は$A_1, ..., A_n$の連言($\wedge$)，後件部は$C_1, ..., C_m$ の選言($\vee$)となっている．また$n = 0$のとき正節，$m = 0$のとき負節と呼ぶ．それ以外の節，すなわち$m \neq 0$かつ$n \neq 0$のときは混合節と呼ぶ．

モデル生成法は，入力節の集合に対するモデルを構築していく手法である．モデルとは，与えられた入力節をすべて満たすために真となるべきアトムの集合で，入力節を一つずつ加えてできる部分集合を満たすモデル候補を段階的に作成していくことで，モデルを構築する．モデルおよびモデル候補は記号Mで表される．

例として，**Fig. 1** のような入力節の集合と，モデル候補の構築を表す証明木を示す．まず(1)を満たすには$p(a)$が真である必要があるので，$p(a)$をモデル候補に加える．次に(2)も同時に満たすためには，$q(a)$か$r(a)$が真であるので，まず$q(a)$をモデル候補に加え，$M = \{p(a), q(a)\}$とする．しかし，これは(3)を満たさないので，このモデル候補はモデルとなり得ない(棄却される)．そこで(2)の分岐先に戻り，$r(a)$をモデル候補に加え，$M = \{p(a), r(a)\}$とすれば(3)も満たすので，これをモデルとする．一方モ

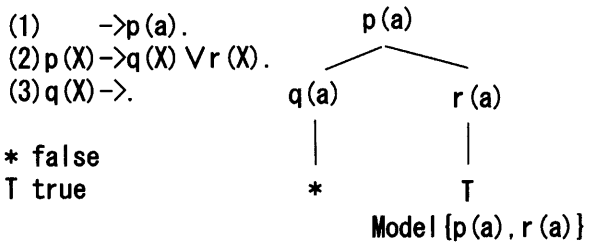

Fig.1　Proof tree.

デルが存在しなかった場合は，入力節の集合を満たす解が存在しないことを表し，その旨を返す．

3. 補　　題

Fig. 2 のような節集合とその証明木について考える．

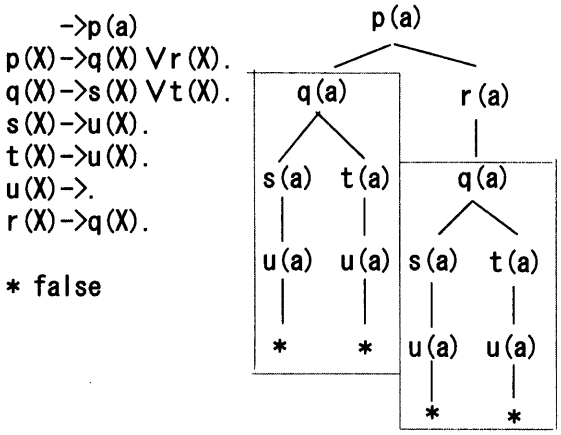

Fig.2　Proof tree.

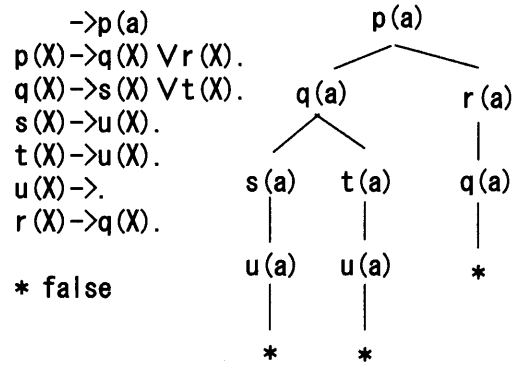

Fig.3　Proof tree used lemma.

この証明木を見ると，$q(X)$以下の部分証明木が全く同じになっている．これは，各部分証明が独立して行われているために，同じ部分証明を繰り返し行っていることを示している．この冗長性を回避するために補題を用いる．**Fig. 2** の場合，左側での部分証明木で補題$q(a) \rightarrow .$または$q(X) \rightarrow .$を作成することができれば，二回目以降の$q(a)$以下の証明は行われず証明木が直ちに閉じる．簡素化された証明木を**Fig. 3** に示す．

以下，$q(a) \rightarrow .$のような変数を含まない補題を固定的補題，$q(X) \rightarrow .$のような変数を含む補題を一般化補題と呼ぶ．なお本論文では補題の増加を抑えるため，前件部が唯一のアトムからなり，かつ後件部がない単位負節補題のみを扱うものとする．

これまで補題生成法として，証明に関連したアトムを計算し固定的補題を作り出す方法が考えられている[3]．しかし固定的補題は変数を含まないため，特定の場合のみにしか適用できず，補題数が増加し，メモリに負担がかかる．加えて，補題適用に時間がかかるという問題を持つ．そこで，変数を含む一般化補題を導入する．

4. 一般化補題の作成法

固定的補題$p(a) \rightarrow .$と一般化補題$p(X) \rightarrow .$を比較すると，変数Xに定数aを代入すれば同じなので，固定的補題は，定数の部分を変数に置き換えた一般化補題に含まれる．したがって補題数増加を抑えるためには，固定的補題よりも一般化補題の方が有効である．

一般化補題は，導出規則(**Fig. 4**)を用いて，二つ以上の節を一つにまとめることで作成される．

$$\frac{\Gamma_1 \rightarrow \Delta_1 \vee A. \quad A \wedge \Gamma_2 \rightarrow \Delta_2.}{\Gamma_1 \wedge \Gamma_2 \rightarrow \Delta_1 \vee \Delta_2.}$$

Fig.4　Binary Resolution.

今Δ_1，Δ_2，Γ_2が空の場合を考えてみよう．このとき，

$$\Gamma_1 \rightarrow A. \quad \text{と} \quad A \rightarrow . \quad \text{から} \quad \Gamma_1 \rightarrow .$$

を得る．この規則の連鎖により複数の負節ができる場合がある．例えば，四つの節

$$A \rightarrow B. \tag{1}$$
$$B \rightarrow C. \tag{2}$$
$$C \rightarrow D. \tag{3}$$
$$D \rightarrow \ . \tag{4}$$

があった場合，

- (3)と(4)から，$C \rightarrow .$
- これと(2)から，$B \rightarrow .$
- これと(1)から，$A \rightarrow .$

というように，続々と補題を作成できる．そこで本論文

では，この規則を用いて補題を作成することにする．この例からも分かるように，補題生成は単位負節から始まるので，単位負節(補題)で証明木が閉じたときから始めればよい．

次に，補題の連鎖的生成を中断する場合を考える．例えば，

$$A \to B \vee D. \quad (5)$$
$$B \to C. \quad (6)$$
$$C \to . \quad (7)$$
$$D \to . \quad (8)$$

があった場合，(5)，(6)，(7)の順で証明木を作り，単位負節(7)から補題生成を始める．

(6)と(7)から

$$B \to . \quad (9)$$

が得られる．次に，(9)と(5)から

$$A \to D. \quad (10)$$

が得られる．これは単位負節ではないので，ここで補題生成を中断すればよい．つぎに(5)の分岐先(8)の証明木を作る．(8)は単位負節なので，ここから補題を生成するが，(5)のBより先は手続きが終わっている．すなわち(5)の代わりに(10)を用い，(10)と(8)から補題

$$A \to . \quad (11)$$

が得られる．

今までの例より，補題生成は次のような手続きを踏めばよいと考えられる．

1. モデル生成法で証明木を作成していく．
2. 証明木が単位負節または補題で閉じたら，証明木の分岐部分が来るまで下から上にたどる形で導出規則を用いて補題を作成していく．
3. 証明木の分岐先まで戻ったら，新たに証明木を伸ばしていく．
4. 以下，証明木がすべてできるまで2. と3. を繰り返す．

これをふまえて，先程の例(**Fig. 3**)で一般化補題を生成する．

$$\to p(a). \quad (12)$$
$$p(X) \to q(X) \vee r(X). \quad (13)$$
$$q(X) \to s(X) \vee t(X). \quad (14)$$
$$s(X) \to u(X). \quad (15)$$
$$t(X) \to u(X). \quad (16)$$
$$u(X) \to . \quad (17)$$
$$r(X) \to q(X). \quad (18)$$

まず(12)，(13)，(14)，(15)，(17)の順に証明木を作成する．証明木が閉じたら，補題を作成していく．ここから

$$s(X) \to . \quad (19)$$
$$q(X) \to t(X). \quad (20)$$

が得られる．次に，(14)の分岐先である(16)，(17)の順に証明木を作成していき，証明木が閉じる．ここから

$$t(X) \to . \quad (21)$$
$$q(X) \to . \quad (22)$$
$$p(X) \to r(X). \quad (23)$$

が得られる．最後に(13)の分岐先である(18)の証明木を作ったあと，補題(22)を用いて証明木が閉じる．ここから，

$$r(X) \to . \quad (24)$$
$$p(X) \to . \quad (25)$$

が得られ，この節集合では合計5つの補題が作成される．

5. 補題の包摂検査

```
int k = 1
while(k ≠ n + 1) {
  if(X_k ⊃ Y) return; /* A ⊃ B... A implys B */
  else if(X_k ⊂ Y) {
    delete Lemma (X_k → .) ;
    add Lemma (Y → .) ;
    return;
    }
  else k = k + 1;
  }
add Lemma (Y → .) ;
return;
```

Fig.5 Lemma subsumption check.

補題を適用するときには，実際に適用できるかどうか検査を行うため，計算コストがかかる．したがって一般化補題であっても，補題を多く保存しておくと，補題の適用に時間がかかることには変わりない．

そこで，ある補題に含意される補題を間引くことを検討する．例えば，補題$p(f(f(X))) \to .$と補題$p(f(Y)) \to .$があったとすると，Yを$f(X)$と置き換えると同じ補題であり．$p(f(f(X))) \to .$は$p(f(Y)) \to .$に含まれていると言える．したがって，狭い範囲の補題$p(f(f(X))) \to .$は不要であり，これを削除することができる．このような検査を，補題の包摂検査[1]と呼ぶ．

補題の包摂検査は，できた補題を登録する際に行われる．補題ができてから包摂検査，補題の登録および不要

補題の削除を行うまでの流れは，現在ある補題を$X_1 \to .$から$X_n \to .$，いま登録しようとする補題を$Y \to .$として，アルゴリズム(**Fig. 5**)で表される．

6. 実　　験

6.1 実験1:一般化補題の効果

実験1として，固定的補題のみを生成するものと一般化補題を生成するものの二つをプログラムを用意し，補題が生成されるテスト問題(**Fig. 6**)を解かせる．

$\to r(a) \vee s(b) \vee t(c).$
$r(X) \to q(X) \vee r(f(X)).$
$q(X) \to q(f(X)).$
$s(X) \to r(f(X)) \vee s(f(X)).$
$t(X) \to s(f(X)) \vee t(f(X)).$
$q(f(f(\ldots f(X)\ldots))) \to .$
$r(f(f(\ldots f(X)\ldots))) \to .$
$s(f(f(\ldots f(X)\ldots))) \to .$
$t(f(f(\ldots f(X)\ldots))) \to .$

Fig.6　Test problem.

この問題は，四つの負節の前件部が$f(f(f(X)))$のような関数fの入れ子になっており，入れ子の中のfの数を変化させることで，問題の大きさを変えることができる．fが二個のとき，すなわち$q(f(f(X))) \to .$(r,s,tも同様)のときの証明木を，固定的補題のみの場合を**Fig. 7**，一般化補題の場合を**Fig. 8** で表す．

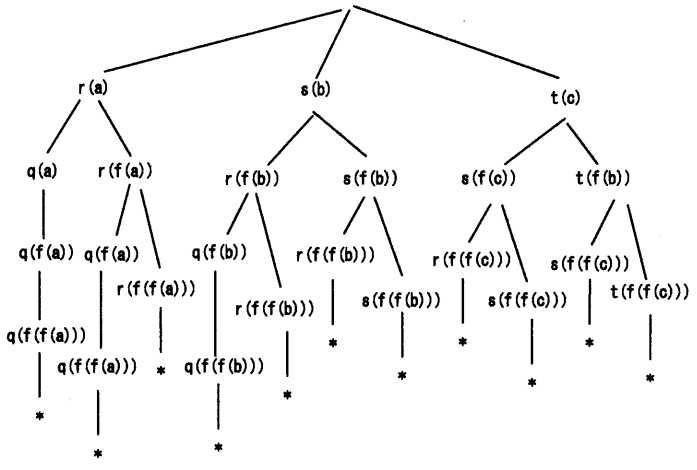

Fig.7　Proof tree(with normal lemma).

実験は，入れ子の大きさを10から100まで，10刻みで変えた問題を解かせ，証明時間，証明木のアトム数，証明木が棄却された数をそれぞれ計測した．なお実験環境は，OSがWindows 2000，CPUがPentium III 500MHz，メモリが320MBである．

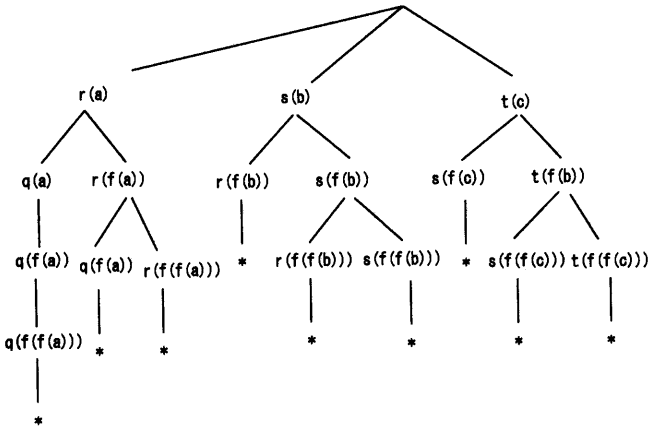

Fig.8　Proof tree(with generalized lemma).

実験結果を**Table 1** に示す．この問題の場合，固定的補題を用いた従来手法では，補題はできるものの適用する機会が無く，fの入れ子が増えてくると，証明木の増大と補題数の増大でメモリが足りなくなる．一方，一般化補題を用いた手法では，補題を効果的に使用することができ，その結果証明木が縮小し，証明時間も短くなっている．

6.2 実験2:一般化補題の包摂検査の効果

実験1と同じ問題(**Fig. 6**)を用いて，一般化補題の包摂検査の有無で計測時間を比較する．実験1と同様，fの入れ子の個数を10から100まで，10刻みで計測した．

結果を**Table 2** に示す．この問題の場合，qが先頭につく一般化補題は，$q(f^{100}(X)) \to .$，$q(f^{99}(X)) \to .$,…，$q(f(X)) \to .$の順番で，合計百個できる．包摂検査をしない場合は最終的にこれら百個の補題がすべて登録された状態になるが，包摂検査を行うと，$q(f^{100}(X)) \to .$と$q(f^{99}(X)) \to .$の包摂検査で$q(f^{100}(X)) \to .$が捨てられ，以下繰り返すと，最後は$q(f(X)) \to .$のみが残る．よって補題の数が大幅に減ることになり，補題適用が容易になる．

この結果から，包摂検査にかかった時間を差し引いても，補題適用時間が短縮されたことで，総合的に問題を解く時間が短縮されていることが分かる．

6.3 実験3:TPTPライブラリの問題を解かせる

実験1と実験2より，一般化補題および補題包摂検査の効果は確認できた．そこで，定理証明器のベンチマークであるTPTPライブラリ [4]の問題を解かせた．その結果単位負節補題は作成できず，補題を作成しようとする時間だけ証明時間が増加した．これは，問題中に単位負節作成の条件となる$\Gamma_1 \to A.$と$A \to .$の形式の節が両方含

Table 1 Experimental Result 1.

	normal lemma			generalized lemma		
nest	total	failed	time	total	failed	time
10	1033	243	110	73	33	50
20	10688	1583	501	143	63	91
30	46468	5023	1912	213	93	110
40	135873	11563	5328	283	123	141
50	316403	22203	13409	353	153	180
60	635558	37943	29693	423	183	211
70	1150838	59783	59606	493	213	261
80	Memory Overflow			564	243	301
90	Memory Overflow			633	273	360
100	Memory Overflow			689	297	440

nest:nest of f　　total:total atoms
failed:failed branches　　time:proof times (ms)

Table 2 Experimental Result 2.

nest	10	20	30	40	50	60	70	80	90	100
-(ms)	50	91	110	141	180	211	261	301	360	440
+(ms)	50	70	100	120	130	150	190	200	240	270

nest:nest of f　　-:without lemma subsumption test
+:with lemma subsuption test

まれている問題がなく，単位負節が作成できなかったことが考えられる．

7. おわりに

本論文では，一般化補題の自動生成法と補題の包摂検査について述べた．

本論文で述べた単位負節補題の生成法は，導出規則の適用によって行われた．しかし，導出規則を適用する前の節に制限があるため，問題の内容によっては単位節補題が全くできないことも考えられ，また前件部のアトムが1個しかないため，適用できる問題も限られてくる．したがって非単位負節補題を今後とり入れる必要があろう．

非単位節負節補題は，前件部のアトムがANDで結合されている負節補題を指す．作り方は単位負節補題と同様，負節からはじめて導出規則で作ることが可能である．例えば，$A \wedge B \rightarrow C.$と$C \wedge D \rightarrow .$を統合すると補題$A \wedge B \wedge D \rightarrow .$ができる．しかし，単位負節補題の作成の際に用いた導出規則適用の制限が無いため，補題数はかなり増えてしまうであろう．今後の課題として，非単位負節補題を生成する場合に，補題生成の制限をどのようにかけていくか，またいかに適用しやすい補題を残すかという点が挙げられる．

参考文献

1) Alexander Leitsch:The Resolution Calculus (Texts in Theoretical Computer Science – An EATCS Series),Springer(1997)
2) K.Iwanuma and K.Kishino: Lemma Generalization and Non-unit Lemma Matching for Model Elimination, ASIAN'99,LNCS 1742,(1999),163–176.
3) 越村三幸;長谷川隆三:"証明の依存性解析によるモデル生成法の冗長探索の削除,"人工知能学会誌,Vol.15,No.6,(2000),1064–1073.
4) Geoff Sutcliffe and Christian Suttner. Thousands of Problems for Theorem Provers, v2.5.0, (2002). http://www.tptp.org/ .
5) 清水純一;越村三幸;長谷川隆三:"数独パズルにおける補題の解析- 補題の一般化に向けて,"電子情報通信学会,(2001),33–38.