九州大学学術情報リポジトリ
Kyushu University Institutional Repository

極小モデル導出法に基づく解集合計算の効率化

白井，康之
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 ： 博士後期課程 | (株)三菱総合研究所情報技術研究部

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/10.15017/1525459

出版情報：九州大学大学院システム情報科学紀要. 7 (2), pp.119-124, 2002-09-26. 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2号 平成14年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.7, No.2, September 2002

極小モデル導出法に基づく解集合計算の効率化

白井康之* ・ 長谷川隆三**

Efficient Implementation of Answer Set Semantics Based on Minimal Model Generation

Yasuyuki SHIRAI and Ryuzo HASEGAWA

(Received June 14, 2002)

Abstract: Answer set programming (ASP), that computes answer sets from extended logic programs including "negation as failure", has been a focus not only in the field of logic programming but also in application areas. The basic algorithm to compute answer sets based on model generation theorem proving has been proposed by Inoue, et al. The implementation of the basic algorithm, however, generates enormous combination of hypotheses inefficiently. In this paper, we propose a new implementation method based on a minimal model generator, MM-MGTP, including some techniques to prune redundant branches. Some experimental results are shown for comparison with the naive implementation and other systems.

Keywords: Answer set, Negation as failure, Extended logic program, Model generation, Minimal model

1. はじめに

失敗による否定を含む論理プログラミングに対して提唱されている複数の意味論の中で，解集合意味論がその応用性の観点から注目されており，近年非単調推論の分野で活発な研究が行なわれている．井上ら[7]は，解集合意味論に基づく定理証明系として，一階述語論理を対象としたボトムアップ型の定理証明システムMGTP (Model Generation Theorem Prover) 上での失敗による否定の取り扱い方法を明らかにした．ここでは，失敗による否定を含む節は，様相オペレータを含む一般の一階述語論理節に変換され，様相オペレータを述語記号と考えることにより，MGTPの証明手続きを変更することなく，解集合の計算が可能である．ただし，MGTP 上での単純な実現は，仮説の組合せによる探索空間の爆発や解集合ではないモデルを大量に生成してしまうなどの問題があり，応用性の観点からは，必ずしも効率的なものではなかった．一方，長谷川[3]らは，MGTP 上で極小モデル生成を行うための効率化手法(MM-MGTP)を提案しているが，こうした極小モデル生成の枠組は，解集合計算においても有効な推論手続きを提供する．本稿では，MM-MGTP をベースにした解集合計算の枠組とその効率化方法を示す．また，実装されたシステムによる実験結果を通じて，本アプローチの有効性を示す．

平成14年6月14日受付

* 知能システム学専攻博士後期課程, (株) 三菱総合研究所 情報技術研究部
** 知能システム学部門

2. MGTP 上の極小モデル導出法

2.1 MGTP

MGTP は，一階述語論理を対象としたボトムアップの定理証明系である．MGTP の入力プログラムは，以下のようなルールの集合からなる．ルールは節とも呼ばれる．

$$A_1, \ldots, A_n \rightarrow C_1; \ldots; C_m \ (n \geq 0, m \geq 0) \qquad (1)$$

$A_1, \ldots, A_n, C_1, \ldots, C_m$ は述語記号を用いて表された正または負リテラルで，$n(\geq 0)$ 引数の述語記号p と項 $t_1, \ldots, t_n$ により，$p(t_1, \ldots, t_n)$ と表される．項は，関数記号と定数，ならびに変数により構成される．変数を含まない項を基底項と呼び，基底項のみから構成される節を基底節という．また $A_1, \ldots, A_n$ は前件部 (antecedent)，$C_1; \ldots; C_m$ を後件部 (consequent) と呼ばれる．前件部の並びは連言であり，ルールが適用される条件を表す．また，後件部の並びは選言であり，ルールが適用された時の帰結を表す．特に，$n = 0$ の場合を正節，$m = 0$ の場合を負節と呼ぶ．また，MGTP のルールは，値域限定条件，すなわち，後件部に出現する項変数は，必ず前件部に出現するという条件が満たされなければならない．これゆえ，前件部が空であるような値域限定ルールは，基底でなければならない．

2.2 MGTP の推論手続き

MGTP の証明手続きは，正節から始めて，モデルとなる候補をボトムアップに生成していく過程である．

1. init : $\mathcal{M} := \{\emptyset\}$, $result := unsat$.
2. do while $\mathcal{M} \neq \emptyset$:
   for $M \in \mathcal{M}$:
   - if there exists a clause $A_1, \ldots, A_n \rightarrow .$ in P, and a substitution σ s.t., $A_1\sigma, \ldots, A_n\sigma \in M$, then :
     $\mathcal{M} := \mathcal{M} - \{M\}$ (model rejection),
     continue while-loop.
   - else if there exists a clause $A_1, \ldots, A_n \rightarrow C_1; \ldots; C_m.$ $(m > 0)$ in P, and a substitution σ s.t.,
     $A_1\sigma, \ldots, A_n\sigma \in M, C_1\sigma, \ldots, C_m\sigma \notin M$, then :
     $\mathcal{M} := (\mathcal{M} - \{M\}) \cup \{\{M \cup \{C_j\sigma\}\} \mid j = 1, \ldots, m\}$ (model extension),
     continue while-loop.
   - else : output M as a model. $result := sat$, $\mathcal{M} := \mathcal{M} - \{M\}$, continue while-loop.
3. if $\mathcal{M} = \emptyset$: return result.

Fig.1　The algorithm of the MGTP procedure

MGTP の推論手続きを**Fig. 1** に示す.

ここで, $\mathcal{M}$ をモデル候補集合と呼び, その要素をモデル候補と呼ぶ. モデル候補は, リテラルの集合であり, 以下, M と表記する. モデル候補の拡張では, ルールの前件部が現在保持されているモデル候補 M により充足され, かつ, 後件部がモデル候補 M に包摂されないとき, その後件部を M に追加する. 後件部が選言である場合には, ケース分割を行なうような拡張が行なわれる. なお, 終了したときに $result$ が sat であれば, 与えられたプログラムは充足可能であり, $unsat$ であれば, プログラムは充足不能である.

2.3 MGTP 上の極小モデル計算

あるモデル M において, $M' \subset M$なるモデル M'が存在しないとき, モデルMは極小であるという. MGTP により導出されたモデル集合は, 与えられたプログラムのすべての極小モデルを含むことが保証されるが, 半面, 選言を含むプログラムにおいてはタブロー法や Davis-Putnum 法と同様に不要な非極小モデルも生成される可能性がある. 長谷川ら [3]は, 分岐補題, 分岐仮定と呼ぶ概念を導入することで, 非極小モデルが導出過程で枝刈りされるような拡張を行った. このように拡張されたMGTPを以降MM-MGTPとよぶ. MM-MGTPにおいては, 導出されるモデルは極小モデルであることが保証される.

3. MGTP 上の解集合計算

3.1 拡張論理プログラムと解集合意味論

(1) の節表現の拡張として, 以下のような節からなる拡張論理プログラムを考える.

$$A_1, \ldots, A_n,\ not\ B_1, \ldots, not\ B_l \rightarrow C_1; \ldots; C_m \quad (n \geq 0, m \geq 0, l \geq 0) \tag{2}$$

ここで, B_i は, A_i, C_i 同様, 正または負のリテラルをあらわす. また, not は失敗による否定(以下, NAF と略記する)をあらわし, 通常の否定とは厳密に区別される.

NAFを含む拡張論理プログラムの代表的な意味論として, 解集合[†1]意味論[6]やWell Founded Semantics [5]があるが, 法的推論等など現実的な応用[10]を考慮したとき, 論証可能なすべての解を生成することが望ましいため, 以降では解集合意味論を対象とすることとする.

解集合意味論は以下のように定義される. 拡張論理プログラム P に対して, P を基底化したものをP_G とする. P_G に対して, リテラル集合 S に関する既約 (reduction) とは, P_G に対して以下の処理によって得られるものであり, $red(P_G^S)$で表す.

- $b \in S$ であるような $not\ b$ を前件部に含むようなルールを消去.
- 残ったルール集合からすべての NAF付きリテラルを消去.

ここで, S がPの解集合であるとは, S が $red(P_G^S)$ の演繹閉包であることをいう. 拡張論理プログラムは一般に複数の解集合を持つ.

3.2 失敗による否定の MGTP 節への変換

前節で示した拡張論理プログラムは, NAF を含まないプログラム, すなわちMGTP の入力プログラムへの変換が可能である[7]. NAFを含むルール (2)は, 以下のように様相オペレータ Kを用いた MGTP 節へ変換される.

$$A_1, \ldots, A_n \rightarrow \neg \mathsf{K}B_1, \ldots, \neg \mathsf{K}B_l, C_1;\ \ldots\ ; \neg \mathsf{K}B_1, \ldots, \neg \mathsf{K}B_l, C_m;\ \mathsf{K}B_1; \ldots; \mathsf{K}B_l. \tag{3}$$

†1 [6] では "安定モデル" と呼ばれているが, 否定を含む場合は, "解集合" と呼ばれて区別されることが多い. 本論文では, "解集合" を用いる.

以下 K の付いたリテラルをK項と呼ぶ．KL は直観的には L が成り立つという仮説を表すリテラルである．K項は以下のような一貫性制約を持つ．

$$\neg \mathsf{K}L, L \rightarrow . \quad (4)$$
$$\neg \mathsf{K}L, \mathsf{K}L \rightarrow . \quad (5)$$
$$\mathsf{K}L, \neg L \rightarrow . \quad (6)$$
$$\mathsf{K}L, \mathsf{K}\neg L \rightarrow \quad (7)$$
$$\neg \mathsf{K}L, \neg \mathsf{K}\neg L \rightarrow . \quad (8)$$

3.3 MGTP による解集合計算

上記変換により求められたプログラムは，K を述語記号と考え，MGTP によりモデルを求めることが可能である．一般に，K を一つの述語記号と考えて実行させた場合には，解集合に加え，解集合以外のモデルも生成される．MGTP のモデル$M \in \mathcal{M}$ から，K項を取り除いたものを $obj(M)$とすると，M が下記2つの条件を満足するとき，$obj(M)$は与えられたプログラムの解集合となる．

1. M は極小解である．すなわち，$M' \subset M$ なる $M' \in \mathcal{M}$が存在しない．[†2]
2. リテラル L に対して，K$L \in M$ ならば $L \in M$が成り立つ(T-条件)．

モデル集合 $\mathcal{M}$の極小解のみを取り出す操作を min とし，T-条件を満足するモデルのみを取り出す操作を $tcond$とすると，$tcond(min(\mathcal{M}))$は，与えられたプログラム P のすべての解集合と一致する．

3.4 冗長性

3.2，3.3で示したナイーヴな実装は，解集合の条件を満足しない(解集合ではない)モデルを大量に生成する．

例：3色問題：
3色問題は，rgb の3色を用いて，ネットワーク上のノードを隣接するノード間で互いに異なるように色分けする問題で，解集合計算におけるベンチマークとして知られている[9]．ノード数2の場合のプログラムは以下の通りである．

$$\rightarrow node(1), node(2), edge(1,2). \quad (9)$$
$$node(X) \rightarrow color(X,r), \neg \mathsf{K}color(X,b), \neg \mathsf{K}color(X,g); \mathsf{K}color(X,b); \mathsf{K}color(X,g). \quad (10)$$
$$node(X) \rightarrow color(X,g), \neg \mathsf{K}color(X,r), \neg \mathsf{K}color(X,b); \mathsf{K}color(X,r); \mathsf{K}color(X,b). \quad (11)$$
$$node(X) \rightarrow color(X,b), \neg \mathsf{K}color(X,g), \neg \mathsf{K}color(X,r); \mathsf{K}color(X,g); \mathsf{K}color(X,r). \quad (12)$$
$$edge(X,Y), color(X,C) \rightarrow \neg color(Y,C). \quad (13)$$
$$edge(X,Y), color(Y,C) \rightarrow \neg color(X,C). \quad (14)$$

ノード数10のケースで，MGTP に一貫性制約 (4)〜(8) のみを追加したバージョン (以下，ナイーブな実装) で実行したところ，最終的な解集合486個に対して，309020もの枝が生成された (うち失敗枝が118496，解集合ではないモデルが190038)．失敗枝が大量に生成される要因は，K 項に関する枝刈りが十分に行なわれていないためである．また，解集合ではないモデルが生成される冗長性に関しては，以下のような原因がある．

- 同じ解集合となるが，仮説が異なるために，複数のモデルとして生成されてしまうこと．たとえば，$M_1 = \{\mathsf{K}color(1,r), color(1,r), color(2,g)\}$, $M_2 = \{\mathsf{K}color(2,g), color(1,r), color(2,g)\}$ は $obj(M_1) = obj(M_2) = \{color(1,r), color(2,g)\}$ であるにも関わらず，異なるモデルとして生成されてしまう．
- T-条件を満足しない仮説のみからなるモデルが大量に生成されてしまうこと (例: $\{\mathsf{K}color(1,r), \mathsf{K}color(2,r)\}$)．

4. MM-MGTP による解集合計算と効率化

解集合計算においては，極小解以外は不要であること，また，MGTP により導出されるモデルが極小であることが保証されていれば，T-条件のみを判定すれば済むことから，解集合を求めるにあたって，MM-MGTP をベースにすることは極めて自然である．

4.1 MM-MGTP プログラムへの変換

MM-MGTP においては，非極小モデルを枝刈りするために，後件部に出現する選言において，自身より右側に出現する要素の否定を分岐仮定として自動的に含むように実装されている．このため，(3) で示した MGTP プログラムへの変換は，以下のように簡略化することが可能である．

$$A_1, \ldots, A_n \rightarrow C_1; \ \ldots \ ; C_m; \mathsf{K}B_1; \ldots; \mathsf{K}B_l. \quad (15)$$

例： プログラム $\{not\ a \rightarrow b, \rightarrow c\ ;\ d, c \rightarrow d, c \rightarrow a\}$ に関して MM-MGTP の生成する証明木は，**Fig. 2**のようになる．ここで，[ ] は分岐仮定を表す．また，⟦ ⟧ は分岐補題を表す．**Fig. 2**では，分岐仮定である $[\neg \mathsf{K}a]$ が b 以降の枝において棄却されていないので，分岐補題 $[\![\neg b]\!]$ を Ka の枝に追加することができる．MM-MGTP でモデルとして導出されるのは，極小解である M_2, M_4 であり，通常の MGTP において導出される M_1, M_3 は導出されない．M_2, M_4 において T-条件を満足するものは M_2 のみであり，実際，M_2 が上記プログラムの唯一の解集合である (通常の MGTP において導出される M_4 は極小であるが T-条件は満たさない．一方 M_3 自体は T-条件を満たすが，$M_4 \subset M_3$ であるため，解集合とはならない)．

MM-MGTP を用いる場合には，M_1, M_3 は非極小モ

†2 もとのプログラムが選言を含まない場合には，極小性の条件は不要である[8]．

デルとして事前に枝刈りがなされている．このため，M_2, M_4 に関する T-条件の検査のみを行えば良い．

一般に，NAF の数が多く，仮説が大量に生成されるようなケースでは，仮説のみからなるケース分割が多発し，全く同様の証明木を繰り返す可能性がある．MM-MGTP による非極小モデルの枝刈りにより，同一の解集合を多数生成するようなケースを削減することができる．

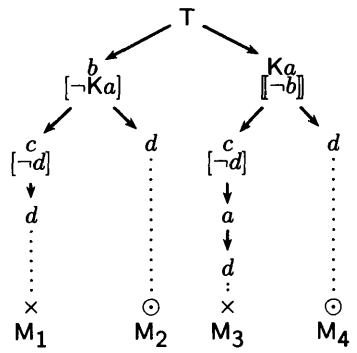

Fig.2 Answer set computation using MM-MGTP procedure

4.2 単位反駁と選言簡約の拡張

MGTP においては，モデル候補内に相補リテラルが存在する場合に即座に棄却する単位反駁，ならびにモデル候補中に存在する相補リテラルを用いて，モデル拡張候補として保持している選言を簡約化する機能 (**Fig. 3**参照．ここで，M はモデル候補，D はモデル拡張候補をあらわす) が組み込まれており，特に正負リテラルが混合する制約充足問題では大きな枝刈り効果が確認されている [2)]．

一方，K 項に関しては，(4)〜(8) に示したような単位反駁としての一貫性制約があるが，これらの制約関係を用いて選言簡約を以下のように拡張することできる．すなわち，選言 $\mathcal{C} = L_1; \ldots; L_i; \ldots; L_n \in D$ に対してある $L \in M$ が存在して，L, L_i が制約 (4)〜(8) のいずれかを満足するとき，選言 $\mathcal{C}$ を，$\mathcal{C}' = L_1; \ldots; L_{i-1}; L_{i+1}; \ldots; L_n$ と書き換えることができる．

$$\frac{L \in M \quad \neg L \in M}{\bot} \qquad \frac{L \in M \quad \neg L; \mathcal{C} \in D}{\mathcal{C}}$$

Fig.3 Unit refutation and disjunction simplification

4.3 選言に関する拡張包摂検査

選言 $\mathcal{C} = L_1; \ldots; L_i; \ldots; L_n \in D$ に対して $L_i \in M$ であるとき，L_i は選言 $\mathcal{C}$ を包摂するといい，M の要素によって包摂される選言は，D から削除される．このような包摂検査は，冗長な枝を生成する選言を事前にモデル拡張候補から除外するという意味において，極めて重要である．

K 項に関しても，それを述語記号だと考えれば，全く同様の包摂検査が行なわれるが，加えて，解集合の定義に基づき，以下のように拡張することができる．すなわち，$L_i \in M$ かつ $\mathcal{C} = L_1; \ldots; \mathsf{K}L_i; \ldots; L_n \in D$ であるとき，$\mathcal{C}$ は，L_i に包摂されるものとして，モデル拡張候補から除外することができる．拡張された包摂検査が解集合計算に影響を与えないことは明らかである．実際，解集合意味論の定義によれば，L_i を含む解集合の演繹閉包においては，$not\ L_1$ を前件部に含むようなルール，すなわち，後件部の選言要素として $\mathsf{K}L_1$ を含むようなルールは除外されるので，こうした選言によりケース分割を行なうことは明らかに冗長である．拡張包摂検査を用いることにより，冗長であることが明らかである枝の分岐，特に，K 項の冗長な組合せを回避することに効果がある．

4.4 拡張連言照合

例：3.4の例において，(10)(11)(12) から，以下のような具体化された選言をモデル拡張候補集合として保持していたとする．

$$\mathcal{C}_1 = color(1,g); \mathsf{K}color(1,b); \mathsf{K}color(1,r)$$
$$\mathcal{C}_2 = color(1,b); \mathsf{K}color(1,r); \mathsf{K}color(1,g)$$
$$\mathcal{C}_3 = color(2,r); \mathsf{K}color(2,g); \mathsf{K}color(2,b)$$
$$\mathcal{C}_4 = color(2,g); \mathsf{K}color(2,b); \mathsf{K}color(2,r)$$
$$\mathcal{C}_5 = color(2,b); \mathsf{K}color(2,r); \mathsf{K}color(2,g)$$

このとき，新たにリテラル $color(1,r)$ がモデル候補に加えられるとする．4.3で示した拡張包摂検査によれば，$\mathcal{C}_1, \mathcal{C}_2$ は包摂検査により取り除くことができる．また，ルール (13) により，$\neg color(2,r)$ が生成されるので，4.2 で示した選言簡約により，選言の枝刈りを行なうことができる．さて，ここで，モデル候補に加えられるのが $color(1,r)$ ではなく，$\mathsf{K}color(1,r)$ であるとする．このとき，$\mathcal{C}_1, \mathcal{C}_2$ は同様に包摂検査により取り除くことができるが，ルール (13) が適用できないので，$\mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5$ については簡約化することができない．結果として，冗長な枝を生成してしまう．このような冗長性は，直観的には $\mathsf{K}color(1,r)$ が生成された時点で，ルール (13) を

$$edge(X,Y), \mathsf{K}color(X,C) \rightarrow \mathsf{K}\neg color(Y,C).$$

と解釈して，$\mathsf{K}\neg color(2,r)$ を生成することにより回避することができる．一般に以下のことが成り立つ．

モデル候補 M に対し，あるホーン節 $A_1, \ldots, A_n \rightarrow C$

があって，すべての $i(1 \leq i \leq n)$ に対して，$KA_i \in M$ または $A_i \in M$ が成り立つとき，$M = M + \{\mathsf{K}C\}$ とする操作を拡張連言照合とする．今，あるプログラム P に対し，MM-MGTP により生成されるモデル集合を $\mathcal{M}$ とし，拡張連言照合を加えた MM-MGTP により生成されるモデル集合を $\mathcal{M}'$ とすると，$tcond(\mathcal{M}) = tcond(\mathcal{M}')$ が成り立つ (証明略).

拡張連言照合を用いることにより，K 項が選択された場合でも，関連する制約伝搬処理を同様に行うことができる．

4.5 K 項に関連する枝分岐抑制

解集合を生成しないような仮説のみの組み合わせが冗長に生成される例として，以下のプログラムを考えてみる．

$$\rightarrow c, d; \mathsf{K}a; \mathsf{K}b. \quad (16)$$
$$\rightarrow a, b; \mathsf{K}c; \mathsf{K}d. \quad (17)$$

このプログラムでは，2 つの安定モデルに加えて，仮説の組合せとして 4 つの冗長なモデル $\{\mathsf{K}a, \mathsf{K}c\}$, $\{\mathsf{K}a, \mathsf{K}d\}$, $\{\mathsf{K}b, \mathsf{K}c\}$ $\{\mathsf{K}b, \mathsf{K}d\}$ が生成される．4.4で示したような拡張連言照合を除けば，K 項は節の前件部に出現することはないので，連言照合には何ら影響を与えない．このため，複数の K 項からなる選言はそれぞれ分岐させる必要は無く，別途選言バッファとして保持したまま分岐させる方法が考えられる．例えば，(16) において，$\mathsf{K}a$, $\mathsf{K}b$ をそれぞれ分岐させるのではなく，$\mathsf{K}a; \mathsf{K}b$ を選言バッファに追加し，上位のモデル候補をそのまま継承した枝を新規に 1 つ作成する．この場合，選言バッファに対しては，モデル拡張候補と同様に，簡約や包摂テスト，単位反駁が行われる．上の例の証明木は **Fig. 4**のようになる．□ は選言バッファを表す．

モデル候補 M に L が含まれるとき，選言バッファに含まれる $\mathsf{K}L$ を含む選言は，包摂テストにより削除される．したがって，モデル M が生成されたとき，選言バッファが空でないとすると，選言バッファに含まれる $\mathsf{K}L$ に対して，$L \notin M$ である．すなわち，モデル M は T-条件を満足しない．これより，T-条件の判定は，モデルが生成された時点で選言バッファが空であるか否かに置き換えることができる．

5. 実 験 結 果

4.で述べた解集合計算の枠組みは，java で実装された MM-MGTP をベースに開発されている．ただし，4.5で述べたケース分割の抑制は，K項を用いた連言照合が起こらないことを前提としている．このため，現在2つのバージョンが開発されており，これらを ASP1, ASP2 とする．ASP1 は，4.1, 4.2, 4.3, 4.4を実装したものである．また，ASP2 は4.1, 4.2, 4.3, 4.5を実装したものである．

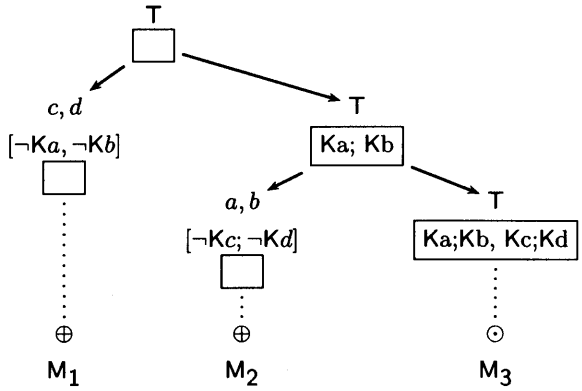

Fig.4 Suppression of case splitting on K literals

以下の実験結果はいずれも Intel Pentium III(266MHz)上の Vine Linux 2.1CR において実行された結果である(java の VM として J2RE 1.3.1 を使用). **Table–1**では準群(QG)問題と呼ばれる有限代数問題(群の要素数(オーダ)として8と11で実行)，また，**Table–2**では3色問題に関するASP1 ならびに ASP2 の実行結果，ならびに比較検討のため，ナイーブな実装(表中の naive)と dlv システムの実行結果を示した．dlv[4]は，NAF や通常の否定，選言を扱うことができ，解集合計算においては高速な処理系†3として知られている代表的なシステムである．表中，AS は解集合の数，MD は総モデル数をあらわす．また，実行時間単位は msec であり†4，TimeOut は300秒以上経過したものを表す．

Table 1 Results for QG problems using NAF

	order	Failed	AS	MD	RunTime
ASP1	8	9	1	1	635
ASP2	8	9	1	1	569
naive	8	23	1	1	813
dlv	8	–	1	–	4500
ASP1	11	112	5	5	8669
ASP2	11	112	5	5	5895
naive	11	> 18000	–	–	TimeOut
dlv	11	–	–	–	TimeOut

Table–1に見られるように，ASP1，ASP2ともに同様の枝刈り効果を持ち，naive 版で見られるような冗長性が回避されていることがわかる．この効果は，オーダを上げた際により顕著である．特に，オーダ11においては，naive 版や dlv では300秒以内で解が導出されなかったのに対し，ASP1,ASP2ともに数秒程度で計算が完了している．

†3 dlv は C++で実装されている．
†4 dlv の実行結果は，time コマンドによる．

Table 2 Results for 3 colors problems using NAF

	Failed	AS	MD	RunTime
ASP1	0	486	540	2885
ASP2	0	486	486	1773
naive	118496	486	190524	99902
dlv	–	486	–	80

3色問題(**Table–2**)についても同様の結果が見られるが，dlv が圧倒的なパフォーマンスを示している．3色問題は準群問題に比較して，扱われる仮説の量が圧倒的に少なく，また，枝刈りのための制約伝搬処理も複雑ではないことから，結果として処理系自体のパフォーマンスが強く反映されたものと思われる．

なお，準群問題，3色問題は，いずれもNAFを用いずに通常の有限領域の制約伝搬問題として記述することが可能である．参考までに通常の MGTP により上記問題を選言を用いて実行させた結果を**Table–3**に示す．ASP1, ASP2による枝刈り効果は，選言表現に対するMGTPにおける枝刈り効果と全く同一であり，実行時間の差は，K 項に関する処理のオーバーヘッドであると推定される．

Table 3 Results using disjunction representation

Problem	Order	Failed	MD	RunTime
QG	8	9	1	408
QG	11	112	5	1425
3colors	10	0	486	596

6. ま　と　め

本稿では，極小モデル導出をベースにした拡張論理プログラムの解集合計算の枠組み，ならびに効率化手法について示し，ベンチマークテストを通じて，従来版と比較して大きな枝刈り効果が達成できることを確認した．

関連するシステムとして，dlv[4]，smodels[9]が挙げられる．dlv では，依存関係に基づく部分計算をベースにしているので，部分構造が定義できるような問題に対しては有効だが，一方，準群問題のような制約伝搬が複雑に絡み合うような問題においては，逆に依存関係の解析がネックになる．また，smodels でははじめに可能性のある仮説集合を生成し，次にこれに基づき，モデル生成を行う．本システムでは，仮説の生成とモデル生成を同時に行っているところが大きな相違であり，モデル候補の情報と仮説集合の情報を互いに制約伝搬させ，枝刈りを行うことができる．

また，MGTPをベースにしたものでは，島尻[1]らによるT-条件に関して関連性テストを用いたアプローチがある．ただし，島尻らのアプローチは，一般論理プログラムを対象としたもので，選言が含まれるような拡張論理プログラムにおいては，関連性テストによる枝刈りによりモデルの極小性が保証されなくなってしまう．本アプローチは，選言を含む拡張論理プログラムにおいても健全性が保証される効率化の枠組である．

最後に今後の課題を簡単にまとめる．dlv, smodels において用いられているような仮説間の依存関係解析は問題によっては非常に有益である．MGTP においてもこうした解析が実行途上においてコストをかけずに計算することができないか，検討する必要がある．また，3色問題で表されるような問題は，本来 NAF を用いて表現するよりも排反な選言として表現し，選言に対する制約伝搬として定式化する方が自然であり，かつ余分な仮説を生成しなくても済むので，効率的である．NAF 表現を排反な選言として表現するための条件，すなわち有限領域の制約伝搬と解集合計算の関係を明らかにすることを今後の課題とする．

参　考　文　献

1) 島尻優香，世木博久，伊藤英則．モデル生成法に基づく安定モデル計算の効率化．人工知能学会誌, 12(5):101–107, 1997.
2) 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, J80-D-II(1):224–236, 1997.
3) 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．人工知能学会誌, 16(2):234–245, 2001.
4) T. Eiter, W. Faber, C. Koch, N. Leone, and G. Pfeifer. Dlv - a system for declarative problem solving. In *Proc. of the 8th Int. Workshop on Non-Monotonic Reasoning (NMR'2000)*, 2000.
5) A. Van Gelder, K. A. Ross, and J. S. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the Association for Computing Machinery*, 38(3):620–650, 1991.
6) M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. 5th Int. Conf. and Symp. of Logic Programming*, pages 1070–1080, Seattle, WA, 1988.
7) K. Inoue, M. Koshimura, and R. Hasegawa. Embedding Negation as Failure into a Model Generation Theorem Prover. In *Proc. 11th Int. Conf. on Automated Deduction*, pages 400–415. Springer-Verlag, 1992. LNAI 607.
8) K. Inoue and C. Sakama. A Fixpoint Characterization of Abductive Logic Programs. *Journal of Logic Programming*, 27(2):107–136, 1996.
9) I. Niemela and P. Simons. Efficient Implementation of the Well-founded and Stable Model Semantics. Research report 7-96, University of Koblenz-Landau, 1996. `http://www.tcs.hut.fi/~ini/papers/`.
10) Y. Shirai, R. Hasegawa, and K.Nitta. Legal Reasoning with Dynamic Rule Preferences Based on Theorem Proving Technologies. In *Proc. of Int. Workshop on Informations & Electrical Engineering 2002*, pages 178–183, 2002.