F-039

モデル生成法を用いた極小モデル生成

Minimal Model Generation Using a Model Generation Theorem Prover

鹿田 憲秀† 長谷川 隆三†† 藤田 博†† 越村 三幸††

† 九州大学大学院システム情報科学府

†† 九州大学大学院システム情報科学研究院

1 はじめに

我々は，従来より一階述語論理対応の自動推論システム MGTP (Model GenerationTheorem Prover) を開発してきている [8]．MGTP は名称が示すようにモデル生成法に基づいており，与えられた論理式を真とする解釈，すなわちモデルを求めることを主要な目的として作られている．なかでも，複数のモデルがあり得る場合に，それらの中で極小なモデルのみを求めることに興味がある [5, 9]．実際，極小モデルの概念は，自動定理証明，論理プログラミング，演繹データベース，ソフトウェア検証，仮説推論等の幅広い分野において重要である．

SAT(Boolean satisfiability problem) は，与えられたブール式を真とするような変数の値の割当てがあるかどうかを判定する問題であり，計算機科学における様々な問題がこれに還元可能なため，古くから重要視されている．最近，ハードウェアの設計・検証やスケジューリング問題等において，実用規模の SAT を効率よく解くための SAT ソルバに関する研究開発が活発化している．なかでも，MiniSat[1] は 2005 年の SAT 競技会で優勝した有名な DPLL 型 SAT ソルバである．また，このような高能率 SAT ソルバはわずかに手を加えるだけで極小モデル探索機能を付加することができる．

本稿では，MGTP を改良し，極小モデル探索機能を追加した MM-MGTP (Minimal Model MGTP) を紹介し，MiniSat を java に書き改め，極小モデル探索機能を付加したもの (MM-DPLL と呼称) との比較実験を行っている．

次節で極小モデル，及び MM-MGTP について述べる．3 節では，ベンチマーク問題を用いた実験結果に基づき，MM-DPLL との比較を中心とした考察を行う．4 節でまとめと今後の課題について述べる．

2 極小モデルと MM-MGTP

2.1 極小モデル

一般に，あるブール式のモデルとは，その式を真とするように真値を割り当てられたリテラルの集合のことである．例えば，$(a \vee b) \wedge (\neg b \vee c)$ のモデルとしては，$\{a, b, c\}, \{a, \neg b, c\}, \{a, \neg b, \neg c\}, \{\neg a, b, c\}$ の 4 つが存在する．

ここで我々は，モデルの要素の中で正リテラルのみに着目する[1]．モデル M の要素から全ての正リテラルを集めた集合を M^+ と表す．2 つのモデル M_1, M_2 の間に $M_1 \subseteq M_2$ が成り立つとは $\forall x \in M_1^+ (x \in M_2^+)$ が成立することをいう．$M \subseteq M_m$ を満たす M が M_m 以外に存在しないようなモデル M_m を極小モデル (Minimal Model) という．

2.2 MM-MGTP

与えられた変数割り当てのもとに，すべての負リテラルが偽で，どの正リテラルも真でなく，真偽値の未定な正リテラルを 1 個以上含む節を**違反節**という．特に，未定正リテラルがただ 1 個ならば単位違反節といい，さもなければ，選言的違反節という．1 個の節において正リテラルには予め順序 $\prec$ が定められているとする．

MM-MGTP の手続きの概略を以下に示す[2]．

1. **[単位違反節の充足]**
   (a) 単位違反節が存在すれば 1 個選択し，その未定正リテラルに真値を割り当てる．存在しなければ，2 へ．
   (b) (1.a) の結果，評価値が偽となる節が生じた（充足失敗）ならば 4 へ．さもなければ，(1.a) へ．
2. **[選言的違反節の充足（分岐点設定）]**
   (a) 選言的違反節 C が存在すれば 1 個選択し，探索の分岐点を設定．存在しなければ，3 へ．
   (b) C 中の正リテラルで順序 $\prec$ に関して最小のもの p を選び，p に真値を割り当てる．
   (c) (2.b) の結果，評価値が偽となる節が生じた（充足失敗）ならば 4 へ．さもなければ，1 へ．
3. **[モデルの非極小性検査]**
   違反節がない，すなわち全節充足可能であるから，ここまでに真偽値を割り当てられた正リテラルの集合として，モデル $M^+ = \{p_1, \cdots, p_n\}$ が得られる．M^+ とこれまでに得られたすべての極小モデルとの間で包摂検査を行う．その結果，M^+ がこれまでのどの極小モデルにも包摂されなければこれを新たな極小モデルとして保存し，さもなければこれを棄却する．その後，4 へ．
4. **[後戻り]**
   探索経路上の適切な分岐点に戻り，$\prec$ に関して次の順位の正リテラルに真値を割り当てて，1 へ．

[1] モデル生成法における自然な解釈となる．

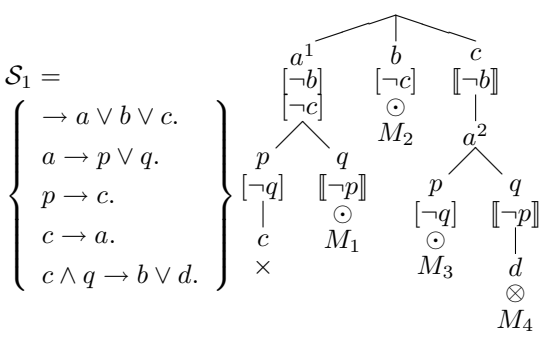

図 1: MG-tree for clause set $\mathcal{S}_1$ with branching assumption and lemma.

2.3 分岐仮定と分岐補題

実際の MM-MGTP においては，そもそも非極小なモデルの生成自体を抑制し，不要な探索と極小性検査を省くため，さらなる工夫が施されている．

図 1 において，入力節集合 $\mathcal{S}_1$ は，CNF の各節を含意式（すべての負リテラルを $\rightarrow$ の左辺に移項して $\wedge$ で結合したもの）で表現している．また，手続きの結果を表現した木において，各ノードは真値を割り当てられた正リテラルでラベルづけされている．$\times$ は充足失敗の枝，$\otimes$ は非極小モデルの枝，$\odot$ は極小モデルの枝を表す．

選言 $a \vee b \vee c$ は $(a \wedge \neg b \wedge \neg c) \vee (b \wedge \neg c) \vee c$ と等価である．そこで，ノード a^1(右上の数字はノードを区別するためのもの．a^2 も同様．) に $[\neg b], [\neg c]$ を付加する．これを，**分岐仮定** *(branching assumption)* と呼ぶ．これにより，モデル $\{a^1, p, c\}$ を棄却する．実はこれは極小モデルなのだが，後で $M_3 = \{c, a^2, p\}$ として再発見されることになる．

分岐仮定を適用した際，最初に得られるモデル（証明木中最も左のモデル）が常に極小であることは，Bry らが証明した [3]．それ以降（証明木中右側）のモデルは必ずしも極小ではない．たとえば，後に得られるモデル M_4 は極小でない．

ノード a の下の $p \vee q$ の分岐では，等価式 $(p \wedge \neg q) \vee q$ を採ってノード p に付加した分岐仮定 $[\neg q]$ が以下の推論に関与しなかったので，改めてもう一つの等価式 $p \vee (q \wedge \neg p)$ を採ることができ，ノード q に $[\![\neg p]\!]$ を付加する．これを，**分岐補題** *(branching lemma)* と呼ぶ．

ノード c に付加された分岐補題 $[\![\neg b]\!]$ は，$c \wedge q \rightarrow b \vee d$ をあたかも $c \wedge \neg b \wedge q \rightarrow d$ として適用するべく効果的に用いられた．ところが，同じノード c に分岐補題 $[\![\neg a]\!]$ が付加できないのは，分岐仮定 $[\neg c]$ が兄ノードの a^1 以下で枝の棄却に関与していたからである．かくして，M_4 が M_1 に対して極小でないことは陽に検査してのみ判ることとなった．さりとて，もし $[\![\neg a]\!]$ を用いたとすると，ノード a^2 の段階で枝が棄却され，極小モデル M_3 をも見落とすことになったであろう．

[2]ここで示すのはモデル記憶方式だが，MM-MGTP には再計算方式も存在し，両者には一長一短あって問題の特性により使い分けられる．

3 実験と考察

MM-MGTP と MM-DPLL との性能比較を行った．MGTP については，先に説明したモデル記憶方式 (MCHK) と，再計算方式 (RCMP) の 2 通りの方式で実験を行っている．また Industrial カテゴリーのインスタンス ferry7 については，MGTP に学習節生成機能を付加して実行している．実行環境は次の通り．

CPU: Intel Core 2 Duo E6600 (2.4GHz Dual Core)

Memory: DDR2 SDRAM PC6400 2GB

OS: Microsoft Windows XP Service Pack 2

Java VM: Java(TM) 2 Runtime Environment, Standard Edition (build 1.6.0_01-b06)

SAT インスタンスとしては，ベンチマークとして近年一般的となった Industrial, Random, Craft の 3 カテゴリーからそれぞれ数問ずつ選んだ．各カテゴリーの特徴は以下のとおり．

Industrial: 回路検証やネットワーク検証など現実の工業分野から生じた問題. (MiniSat1.14 が最優秀の成績をおさめた部門)

Random: ランダムに生成された問題．主に 3SAT.

Crafted: N-Queens，ラテン方陣等パズル系の問題.

実験結果を表 1 に示す．制限時間は 2000 秒とした．T.O. は時間切れを表す．Vars は変数の数，Cls は節数，Minimal Models は極小モデル数で，この値が 0 の欄は UNSAT インスタンスであることを示している．Conflicts は充足失敗の回数を表す．

3.1 考察

Random カテゴリーの特徴は，充足不可能，あるいは Conflict 数は多いが極小モデル数が少ないことである．これらのほとんどのインスタンスで DPLL が MGTP に優っている．その理由は，MGTP は負リテラルに関する推論が不十分，かつ，充足失敗の際に限定的な補題しか生成しないため探索空間の刈り込みが不十分だからである．これに対し，DPLL では上手く生成・管理された学習節による探索空間の刈り込みが，著しく功を奏しているといえる．

Industrial，Craft カテゴリーでは，ほとんどのインスタンスで MGTP が DPLL に優っている．それらにおいては，一般に問題記述に冗長性が少ない．ゆえに充足失敗数が多くてもそれらの原因に重複がない．言い換えると学習節が有効に働かないということが考えられる．こうして，DPLL では無駄な学習節の生成・管理のために計算コストが浪費されることとなる．加えて，極小モデル数が多い場合には反モデル節

表 1: Results on standard SAT benchmarks.

Instance	Vars	Cls	Minimal Models	Conflicts			CPU time (sec)		
				DPLL	MGTP		DPLL	MGTP	
					MCHK	RCMP		MCHK	RCMP
Industrial									
ferry7	1,946	22,336	5,040	28,375	13,200	13,200	9.97	7.64	**7.11**
depots1	161	496	2	13	15	15	**0.02**	**0.02**	**0.02**
driverlog1_v01i.r	128	493	338,100	96,387	1	1	823.05	3.05	**2.58**
driverlog1_v01a.r	128	489	676,269	—	1	1	T.O.	5.78	**4.72**
driverlog1_v01i.s	128	493	149,184	18,185	170,741	166,101	381.59	4.22	**2.47**
rovers1_ks99i.r	439	5423	17	56	47	47	0.06	**0.02**	**0.02**
rovers1_ks99i.s	439	5423	7992	56	10676	10676	1.44	**0.50**	3.48
Random									
uf100-04	100	430	2	598	3,041	3,041	**0.05**	0.13	0.13
uuf125-01	125	538	0	933	7,255	7,255	**0.06**	0.28	0.28
uf100-06	100	430	4,730	3,811	27,870	27,881	0.61	**0.56**	1.13
uuf150-01	150	645	0	3,203	64,178	64,178	**0.06**	2.59	2.56
uf200-01	200	860	3	14,992	832,882	832,882	**0.53**	42.14	42.10
Crafted									
qg7-09	729	22,060	4	28	4	4	0.11	**0.02**	**0.02**
qg6-10	1000	33,466	0	230	63	63	0.16	**0.05**	**0.05**
qg5-09	729	28,540	0	19	19	19	0.11	**0.02**	0.03
qg4-08	512	9,685	0	735	605	605	**0.13**	0.14	0.15
qg4-09	729	15,580	194	37,660	29,559	29,559	**7.92**	8.50	8.40

の蓄積により，性能はさらに悪化する．これに対し，MGTP における失敗学習の情報や極小モデル集合は木構造に縮約されているので，記憶量の増大に関して DPLL より有利であるし，再計算方式に切り替えてこの問題を回避することも可能である．

しかしながら，もっと根本的な要因は分岐探索の違いにあると考えられる．

- MGTP の分岐：
  3 節の手続きの [**分岐点設定**] において，選言的違反節 C が分岐ノードをラベルづけし，C 中の未定値の正リテラル数 N に対応して N 分岐する．
- DPLL の分岐：
  4 節の手続きの [**探索の分岐**] において，まず，すべての未定変数 p_i に対して，ある基準に基づき評価値 $E(p_i)$ が計算される．これに基づいて最良の変数 p^1 が選ばれ，これが分岐ノードをラベルづけし，p^1 に対する真偽値割り当てに対応して 2 分岐する．

たとえば，節集合 $\{a_1 \vee a_2 \vee a_3,\ b_1 \vee b_2,\ c_1 \vee c_2 \vee c_3 \vee c_4\}$ に対して MGTP では，そもそも各節から 1 個ずつのリテラルを選んだ $\{a_i, b_j, c_k\}$ の形のモデルしか探索しない．言い換えれば，MGTP はできる限り各節を排他的論理和として充足させようと試みる．さらに分岐仮定と分岐補題により，この考え方をより徹底させている．

一方，DPLL では $\{a_1, a_2, a_3, b_1, b_2, c_1, c_2, c_3, c_4\}$ のような非極小なモデルを特別視しないし，実際，変数選択に用いられる評価関数 $E(\cdot)$ がこれを避けるようには作られていない．かくして，DPLL と MGTP とでは探索木のサイズが大きく異なる可能性がある．

この違いを端的に実証する例として，次の節集合族 $\mathcal{S}_2^n$ を考える．

$$\{\ p_1^i \vee p_2^i \vee p_3^i \vee p_4^i \vee p_5^i \vee p_6^i \vee p_7^i \vee p_8^i \vee p_9^i \vee p_{10}^i \mid 1 \leq i \leq n\}$$

$\mathcal{S}_2^n$ の MGTP による探索木は高さが n の完全 10 分木となり，生成されるモデルの総数は 10^n で，全て極小である．すべての分岐で分岐補題が適用可能であり，極小性検査を一切行わずに済むという MGTP には有利なインスタンス群である．一方，DPLL の探索

表 2: Results on MM-hard problems.

Instance	Vars	Cls	Minimal Models	Conflicts			CPU time (sec)		
				DPLL	MGTP		DPLL	MGTP	
					MCHK	RCMP		MCHK	RCMP
$\mathcal{S}_2^n$									
n=4	40	4	10,000	1,193	0	0	0.45	0.05	**0.03**
n=5	50	5	100,000	20,475	0	0	59.17	0.22	**0.20**
n=6	60	6	1,000,000	—	0	0	T.O.	22.17	**17.16**
$\mathcal{S}_3^n$									
n=17	69	83	131,072	17,848	1	1	269.08	**1.02**	4.03
n=18	73	88	262,144	59,461	1	1	1,310.30	**2.13**	8.89
n=19	77	93	524,288	—	1	1	T.O.	**4.42**	19.53

木は最悪の場合，高さ $10n$ の完全 2 分木となる．実験結果を 表 2 に示す.

もう一つの節集合族 $\mathcal{S}_3^n$ は，すべての分岐で分岐補題が適用不可能であり，常に極小性検査を強いられるという MGTP にも極めて不利なインスタンス群であるが，これらにおいてさえ，上と同様の理由によって DPLL の性能劣化の方が顕著となっている.

4 まとめ

実験の結果，標準 SAT ベンチマークの Industrial, Crafted の SAT インスタンス群に対して極小モデル生成器 MM-MGTP は MM-DPLL よりも優位な性能を示した．これは，考察で述べたように，分岐探索の原理が異なることと，動的に付加される節の数の爆発的増大が主要な原因である．MGTP は極小モデル生成のために特別に効率化されているが，充足失敗に関する探索空間削減能力が限定的である等の弱点がある．ただし，負リテラルに関する推論や学習節の取り扱いについて強化すれば，SAT ソルバとしてさらなる性能向上の余地がある．これに対し，Random のインスタンス群では MM-DPLL のほうが優位な結果となっている．これは，充足失敗の学習による探索空間の削減能力等，基本的性能において DPLL 型 SAT ソルバが MGTP に優っているためと考えられる．また，DPLL 型 SAT ソルバに若干の修正し，全ての極小モデルだけを得るように変更を施すことは比較的容易であり，有意義である.

両者の改善と比較の再評価が今後の課題である.

謝辞

本研究は科研費（20240003）の助成を受けたものである.

参考文献

[1] Eiklas Eén, Niklas Söresson. An Extensible SAT-solver [extended version 1.2], *SAT2003.*

[2] Elazar Birnbaum and Eliezer L. Lozinskii. The Good Old Davis-Putnam Procedure Helps Counting Models. *Journal of Artificial Intelligence Research,* **10**, pp.457–477, 1999.

[3] François Bry, Adnan Yahya. Minimal Model Generation with Positive Unit Hyper-Resolution Tableaux. *Proc. of 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods,* LNAI 1071, pp.143–159, 1996.

[4] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM,* **5(7)**, pp.394–397, July 1962.

[5] Ryuzo Hasegawa, Hiroshi Fujita, and Miyuki Koshimura. Efficient Minimal Model Generation Using Branching Lemmas, *Proc. of 17th International Conference on Automated Deduction,* LNAI 1831, pp.184–199, Springer, June 2000.

[6] MiniSat Page. http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/

[7] SAT Competitions. http://www.satcompetition.org/

[8] 長谷川 隆三，藤田 博．Java によるモデル生成型定理証明系 MGTP の開発．情報処理学会論文誌，Vol.41，No.6，pp.1791–1798，2000 年 6 月.

[9] 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，16 巻，2 号，2001 年.