3J1-OS7-8

# SOL タブロー計算法に基づく命題論理の充足可能性判定器の実現

A SAT Solver based on SOL Tableau Calculus

鈴木健士郎*1　　鍋島英知*2
Kenshirou SUZUKI　Hidetomo NABESHIMA

*1山梨大学大学院医学工学総合教育部コンピュータ・メディア工学専攻
Computer Science and Media Engineering,
Department of Education Interdisciplinary Graduate School of Medicine and Engineering, University of Yamanashi

*2山梨大学大学院医学工学総合研究部
Department of Research Interdisciplinary Graduate School of Medicine and Engineering, University of Yamanashi

In this paper, we propose a new SAT Solver based on SOL tableau calculus. Most of modern SAT solvers are based on DPLL procedure and have various kinds of techniques for improving the performance of DPLL greatly, for example, conflict driven clause learning, backjumping, watched literals and so on. On the other hand, SOL tableau calculus is one of consequence finding procedures for the first order logic. Toward consequence finding for the propositional logic, we introduce a SAT solver based on SOL with SAT techniques in this work.

## 1. はじめに

本研究では，命題論理の結論発見器の実現を目標とし，結論発見手続きをベースとした命題論理の充足可能性判定問題を解くソルバーを提案する．

公理集合から自明ではない興味深い結論を導出することを**結論発見** (consequence finding) と呼ぶ．結論発見は，隠れている結論を発見する点において，定理証明とは異なる．一般に公理集合から導出可能な結論は無限に存在するため，実際の結論発見では，特定の言語バイアスを満たし包摂に関して極小な節（これを**特徴節** (characteristic clauses) と呼ぶ）を計算する．特徴節の概念は，アブダクション [4, 5]，帰納論理プログラミング [6]，生物情報科学 [2, 7] などを含む，人工知能にとって興味ある種々の推論問題に有用である．特徴節計算のための最もよく知られた手続きの 1 つが **SOL 導出** [4] (skipped ordered linear resoution) であり，SOL 導出をタブロー計算法で再定式化した **SOL タブロー計算法** [8] に基づく効率の良い実装として SOLAR が開発されている [11].

SOLAR は，一階述語論理を対象とした結論発見器である．命題論理の結論発見器としても利用可能であるが，命題論理に特化した実装技術は備えていない．一方，命題論理における充足可能性判定問題（satisfiability problem; SAT 問題）を解くソルバーは，近年劇的に性能を向上させており，いくつもの高速化技術が開発・提案されている．そこで本研究では，命題論理における高速な結論発見器の実現を目標とし，結論発見手続き SOL に最新の SAT 技術を導入したアルゴリズムを提案する．ただし本稿では結論発見ではなく反駁発見（すなわち充足可能性の判定）を対象とする．このシステムでは，充足不能の原因の発見や最適化問題を高速に解くことなどが期待される．

SAT は，最初に NP 完全性が証明された問題であり，計算理論にとって中心的である．また， SAT は，論理合成，ハードウェア・ソフトウェアの検証，制約充足問題，プランニング，自動推論・定理証明など，計算工学や人工知能における数多くの問題において基本となる．このため， SAT を解くアルゴリズムおよびそれを実装したソルバーに関して，半世紀前からこれまでに膨大な量の研究がなされてきた．こうした多くの努力が結実し，この十数年で SAT ソルバーの性能は飛躍的に向上した．

連絡先: 鈴木健士郎，山梨大学大学院医学工学総合教育部コンピュータ・メディア工学専攻，住所： 〒 400-8511 山梨県甲府市武田 4-3-11，E-mail: kenshirou@nabelab.org

## 2. SOL と SAT

### 2.1 結論発見手続き SOL

まず一階述語論理における諸定義と結論発見手続き SOL に関する諸定義を行う．**リテラル**とは，述語またはその否定である．**節**とはリテラルの選言である．**生成領域**とは，リテラル集合と特定の条件により定義され，特徴節が属する語彙を表す．形式的には生成領域は $\mathcal{P} = \langle L, Cond \rangle$ と表され，$L$ はリテラルの集合，$Cond$ は満たすべき特定の条件である．**ある節 $C$ が $\mathcal{P}$ に属する**とは，$C$ に含まれる全てのリテラルが $L$ に含まれ，かつ $C$ が条件 $Cond$ を満たしている場合をいう．また節集合 $\Sigma$ に対して， $\Sigma$ の論理的帰結で生成領域 $\mathcal{P}$ に属する節の集合を $Th_{\mathcal{P}}(\Sigma)$ と表す．$\Sigma$ を節集合とするとき，$\mu\Sigma$ は $\Sigma$ 中の任意の節により真に包摂されない節集合を表す．節集合を $\Sigma$，生成領域を $\mathcal{P}$ とした場合，**$\mathcal{P}$ に関する $\Sigma$ の特徴節集合** $Carc(\Sigma, \mathcal{P})$ を $\mu Th_{\mathcal{P}}(\Sigma)$ と定義する．すなわち特徴節とは，生成領域に属する節で包摂のもとで閉じている節である．**節タブロー** $T$ とはラベル付き順序木であり，$T$ の根以外のすべてのノードはリテラルでラベル付けされている．混乱のない限りリテラルとノードを同一視する．あるノード直下のリテラルが $L_1, \ldots, L_n$ であるとき，節 $L_1 \vee \ldots \vee L_n$ を**タブロー節**と呼ぶ．特に根の直下のタブロー節を**先頭節**と呼ぶ．**連結タブロー** $T$ とは，根以外のすべての非葉ノード $L$ が直下の子として $\neg L$ を持つ節タブローをいう．**マーク付タブロー**とは，いくつかの葉に *closed* または *skipped* で印付けされた節タブローである．マーク付けされていない葉は**サブゴール**と呼ばれる．$T$ が**解けた**とは，$T$ のすべての葉ノードがマーク付けされている場合をいう．$skip(T)$ を *skipped* でマーク付けされたすべてのリテラルの集合とする．**選択関数** $\phi$ とは，タブローからサブゴールへの写像である．本稿では，選択関数はタブロー中の最左のサブゴールを返す写像とする．

**SOL タブロー計算法** [8] を以下で定義する．$\Sigma$ を節集合，$C$ を節，$\mathcal{P}$ を生成領域，$\phi$ を選択関数とする．$\Sigma + C$ と $\mathcal{P}$ からの $\phi$ による節 $S$ の **SOL 演繹**は，次の条件を満たすタブローの列 $T_0, T_1, \ldots, T_n$ である．

1. $T_0$ は，先頭節 $C$ のみからなるタブローである．
2. $T_n$ は，解けたタブローである．
3. 各 $T_i$ において，$skip(T_i)$ は $\mathcal{P}$ に属す．
4. $T_{i+1}$ は，以下の手順で導出する．まず $\phi$ によりサブゴール $K$ を選択する．次に，以下のオペレーターを $T_i$ に適用し，$T_{i+1}$ を得る．
   (a) **Skip:** もし，$skip(T_i) \cup \{K\}$ が，$\mathcal{P}$ に属すなら，$K$ に *skipped* とラベル付けする．
   (b) **Factoring:** もし $skip(T_i)$ がリテラル $L$ を含み，$K$ と $L$ が $\theta$ によって単一化可能ならば，$K$ に *skipped* とラベル付けし，$T_i$ に $\theta$ を適用する．
   (c) **Extension:** $\Sigma \cup \{C\}$ から節 $B$ を選択し，変数の名前替えをした亜種を $B' = L_1 \vee \cdots \vee L_m$ とする．もし，$\neg K$ とリテラル $L_j$ が $\theta$ によって単一化可能ならば，サブゴール $K$ に新しい子 $L_1, \ldots, L_m$ を付加し，$L_j$ には *closed* とラベル付けする．得られたタブローに $\theta$ を適用する．
   (d) **Reduction:** もし，$K$ が祖先ノードに $L$ を持ち，$\neg K$ と $L$ が $\theta$ によって単一化可能ならば，$K$ に *closed* とラベル付けし，$T_i$ に $\theta$ を適用する．

SOL タブロー計算法の正当性は次の定理で示される．

**定理 1** [8]
**健全性:** もし $\Sigma + C$ と $\mathcal{P}$ からの $\phi$ による節 $S$ の *SOL* 演繹が存在するならば，$S$ は $Th_{\mathcal{P}}(\Sigma \cup \{C\})$ に属する．
**完全性:** もし節 $F$ が $Th_{\mathcal{P}}(\Sigma)$ には属さないが $Th_{\mathcal{P}}(\Sigma \cup \{C\})$ に属する場合，$F$ を包摂する節 $S$ の $\Sigma + C$ と $\mathcal{P}$ からの $\phi$ による *SOL* 演繹が存在する．

本稿では，SOL タブロー計算法における 4 種類の推論規則を**オペレーター**と呼ぶ．また SOL タブロー計算法における探索空間を表現するため，タブロー探索木を定義する．$\Sigma + C$ と $\mathcal{P}$ からの $\phi$ による**タブロー探索木** [8] とは，以下のように構成されるタブローでラベル付けされた木である：

1. $\mathcal{T}$ の根は先頭節 $C$ のみからなるタブローである．
2. $\mathcal{T}$ の葉ではない各ノード $T$ は，$T$ のサブゴール $\phi(T)$ にオペレーターを適用して得られるタブローを子に持つ．

## 2.2 最新 SAT ソルバーにおける技術

次に命題論理における諸定義と最新 SAT ソルバーで利用されている技術を紹介する．SAT 問題は，与えられた命題論理式を充足できるか判定する問題であり，通常，節の連言で表記された連言標準形 (conjunctive normal form; **CNF**) の式で与えられる．つまり， SAT 問題は，与えられた問題の全ての節を充足できるか判定する問題である．

SAT 問題を解く技術として以下のような技術が提案され，多くの SAT ソルバーで実装されている．

- DPLL 手続き [9]
- 監視リテラルによる高速単位伝搬 [10]
- 矛盾からの節学習とバックジャンプ法 [14]

**Input:** CNF $\psi$.
**Output:** SAT, UNSAT or UNKNOWN.

```
if ψ is empty then
  return SAT;
end if
ψ := UnitPropagation(ψ);
if φ exists in ψ then
  return UNSAT;
end if
Choose variable x in ψ by some hueristic;
if DPLL(ψ∧ x) return SAT then
  return SAT;
else
  return the result of DPLL(ψ ∧ ¬x);
end if
```

図 1: DPLL

**Input:** CNF $\psi$.
**Output:** CNF.

```
while φ doesn't exist and a unit clause l exists in ψ do
  Assign True to l and simplify ψ;
end while
return ψ;
```

図 2: Unit Propagation

- 優れた変数選択ヒューリスティックス [10]
- リスタート戦略 [12]

ここでは，基本アルゴリズムの DPLL 手続きと高速化に最も効果があると考えられる矛盾からの節学習について述べる．

## 2.3 DPLL 手続き

DPLL 手続きを図 1 に示す．DPLL 手続きは，以下を繰り返し探索する手法である．

1. **単位節**[*1]を発見し，真を割り当てる．
2. 単位節がなければ，適当な変数を選択し，真または偽を割り当てる．このことを**決定**による割り当てと呼び，決定により値が割り当てられた変数を**決定変数**，決定変数の数を**決定レベル**と呼ぶ．
3. 矛盾が発生した場合，最後の決定変数の真偽値を反転する．

## 2.4 矛盾からの節学習

矛盾から節学習を行うソルバーを **CDCL (Conflict Driven Clause Learning) ソルバー**と呼ぶ．CDCL ソルバーは DPLL 手続きに基づくソルバーで，矛盾が発生するたびに，その原因を解析し，同様の矛盾発生を避けるための節を学習する．そして矛盾の原因を解消できるレベルまでバックトラックする．バックトラックするレベルを常に “現在の決定レベル-1“とすると， DPLL ソルバーと同じ動作をする．本稿では紙面の関係上，矛盾解析手法の詳細には触れないが，詳細は論文 [14] を参考にされたい．この節学習により，矛盾を含む部分探索空間を取り除くことが可能になり，無駄な探索を抑えることが可能になる．

*1 **単位節**とは，未割り当てのリテラルが 1 つの節である．

## 3. SOL 手続きに基づく SAT ソルバー

本章では，先に述べた SOL 手続きに 最新 SAT 技術を導入したソルバーを提案する．まず SAT 問題における SOL タブロー法について述べる．

### 3.1 SAT 問題における SOL タブロー計算法

SOL タブロー計算法は，2 章で示した通り 4 つのオペレーターを使用して結論発見を行う．本稿では，反駁発見（充足可能性の判定）を対象とするので，結論発見のためのオペレータである Skip/Factoring は考慮しない．また，命題論理では変数への代入が起きないため， Reduction オペレーターを Identical reduction オペレーター [8] に置き換えることが可能である．Identical reduction は，強制的な推論規則であり，もし Identical reduction が適用可能な場合は，そのほかの推論規則を考慮する必要がなくなるため，効率改善が期待できる．

### 3.2 SOL と SAT 技術の統合

単位伝搬を組み込んだ SOL について述べる．SOL で，サブゴールを選択し Extension を行う時は，サブゴールを真と仮定し探索することに相当する．そこで，この時に単位伝搬を行い，より効率よく探索する．また，単位伝搬により矛盾が発生することがある．この場合，そのノードを真とすると矛盾することが分かるので，そのノードを閉じる．また，単位伝搬により矛盾が発生した場合は節学習を行い，以下に示す手順でバックトラックを行う：

1. 学習節からバックトラックすべき決定レベルを求める．SOL タブロー法では木の深さに相当する．
2. バックトラックするレベル以降の割り当てをキャンセルする．
3. そのノードで単位伝搬を行う．
   (a) 矛盾が起きれば，さらにバックトラックを行う．
   (b) 矛盾が起きなければ，矛盾が発生した決定のひとつ前までの決定を再現する．これは SAT ソルバーで用いられている高速化技術の 1 つである phase saving[13] と同様の手法である．もしこの途中で矛盾が発生した場合は，さらにバックトラックを行う．

SOL 手続きに，SAT 技術である単位伝搬，矛盾からの節学習とバックジャンプ法を統合したアルゴリズムを図 3 に示す．基本的なアルゴリズムは SOL タブロー計算法と同様である．提案 SAT ソルバーでは，入力として開始タブロー $T$ を受け取り，タブロー $T$ が解けていない間，以下を繰り返す．

1. 選択関数 $\phi$ によりサブゴール $K$ を選択する．
2. $K$ を真と仮定し，単位伝搬を行う．
3. 以下の中で当てはまる場合を行う．
   (a) 単位伝搬により矛盾が生じた場合， backtrack により新たなタブローを生成する．
   (b) 単位伝搬により矛盾が生じない場合，$K$ に対するオペレーター集合を作り，以下により一つのオペレーターを選択する．選択したオペレーターを適用し，新たなタブローを生成する．
      i. オペレーター集合が空の場合，以下で説明する RME によりオペレーターを生成する．生成できない場合，モデルを出力し SAT と返す．

**Input:** Tableau $T$.
**Output:** SAT, UNSAT.

```
while T is not solved do
  select subgoal K by φ;
  if UnitPropagation(K) cause conflict then
    make a new tableau T by doing backtrack();
  else
    make operators ops for K;
    if ops is null then
      if unsatisfied top clause C existed then
        print add C as children to K
      else
        print sat model;
        return SAT;
      end if
    else
      select one operator op from ops;
    end if
    make a new tableau T by applying op to T;
  end if
end while
print unsat core;
return UNSAT;
```

図 3: 提案アルゴリズム

      ii. 上記ではない場合，オペレーター集合の中から選択関数によりオペレーターを選択する．

タブローが解けた場合，充足不能の原因を出力し UNSAT と返す．

### 3.3 SAT ソルバーに求められる機能

通常， SAT ソルバーには以下の機能が求められる：

- 充足可能な問題の場合，モデルを出力する．
- 充足不能な問題の場合， UNSAT と出力する．

SOL タブロー計算法では，充足不能の原因を示す閉じたタブローを求めるので，充足不能な問題に対して UNSAT と出力することとその原因を出力可能である．それに対し， SOL タブロー計算法ではモデルの出力を考慮していないので，充足可能である問題に対してモデルの出力ができない．そこで，**RME**(Restart Model Elmination)[1] を導入することでその問題点を解決する．

**RME** とは，選択関数によって選ばれたサブゴールに対して適用できるオペレーターがない場合，そのサブゴールを新しい根頂点と考え，そこから再び SOL 手続きを開始する手法をいう．このとき先頭節としては，未充足の節を用いる．そのような未充足の先頭節がない場合，その SAT 問題は充足可能と判定できる．このとき，根からそのサブゴールまでのパスがモデルとなる．また，求まったモデルに未割り当てのリテラルがある場合先頭節から判断することができる．

上記について，以下の例を用いて説明する．

[例 1] 以下の 5 個の節からなる CNF 式を考える．

$$C_1 = \{\neg a, \neg d\} \qquad C_4 = \{\neg c, +a\}$$
$$C_2 = \{+a, +b, +c\} \qquad C_5 = \{\neg b, \neg c\}$$
$$C_3 = \{\neg b, +a\}$$

この例では，先頭節として負節 (節に含まれるリテラルがすべて負リテラルである節) を選択するものとする．まず先頭節として負節 $C_1$ を選択し，初期タブローを生成する．選択関数 $\phi$ により $\neg a$ を選択する．$\neg a$ を真とし，単位伝搬を行うと

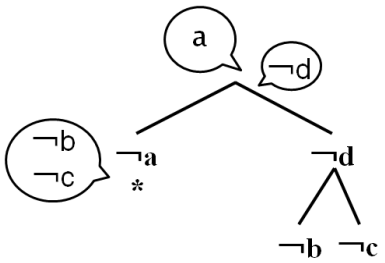


図 4: 例 1 におけるタブロー

表 1: 解けた問題数の評価

| 問題の種類 | Minisat-2.0 | 提案ソルバー |
|---|---|---|
| SAT(81 問) | 43 問 | 13 問 |
| UNSAT(118 問) | 45 問 | 14 問 |
| TOTAL(292 問) | 88 問 | 27 問 |

$\neg b, \neg c$ が真となる．すると，節 $C_2$ が矛盾するので，サブゴールを閉じる．$\neg a$ を真としたことにより，矛盾したことが分かるので，節 $+a$ を学習する．これにより $\neg d$ が単位伝搬により真となる．次に， $\phi$ により $\neg d$ を選択し，真とする．ここで， $\neg d$ に対して，適用できるオペレーターがないので RME を適用する．今回は先頭節として負節を用いているので，節 $C_5$ を先頭節とする．次に， $\phi$ により $\neg b$ を選択し，真とする．ここで， $\neg b$ に対して，適用できるオペレーターがないので RME を適用する．しかし，未充足の負節がないので，この時の根から $\neg b$ までのパスがモデルとなる．つまり，モデルは $a \wedge \neg d \wedge \neg b$ となる．未割り当ての変数として $c$ がある．今回は負節を先頭節としたので，未充足の節は少なくとも正リテラルを 1 つ含んでいる．従って， $+c$ を真とすることで充足させることができる．最終的に求まるタブローを図 4 に示す．

## 4. 評価実験

実験に用いた問題は，SAT Competition 2009 の Application 部門の問題 292 問で，1 問あたりの制限時間は 600 CPU 秒とした．実験に用いたマシンは Mac mini, Core Duo 1.66GHz, 2GB RAM である．比較対象として， Minisat[3] を用いた．

実験結果を表 1 に示す．表中の SAT は充足可能な問題の解けた問題数， UNSAT は充足不能な問題の解けた問題数，TOTAL は SAT と UNSAT と UNKNOWN の合計を示す．提案手法は Minisat に比べ 3 割程度の問題しか解けていない．一方，提案ソルバーでは Minisat が制限時間内に解けていない充足不能な問題を 4 問を解いている．提案ソルバーの性能が悪いのは， SOL への SAT 技術の組み込みの考慮不足， SAT 技術における変数選択ヒューリスティック，リスタート戦略の未導入や効率のよい実装方法などの原因が考えられる．具体的に，単位伝搬と決定の速度を表 2 に示す．表では共通に解けた各部門における単位伝搬と決定の 1 秒当たりのそれぞれの回数を示す．表 2 を見ると，提案ソルバーは miniSAT に比べて，圧倒的に速度が遅い．Total でみると単位伝搬が約 30 倍，決定が約 115 倍ほど遅い．これは，上記で書いたことが原因だと思われる．そこで，これらを改善することは今後の重要な課題の 1 つである．

## 5. まとめと今後の課題

本論文では，命題論理の結論発見器の実現を目標とし，命題論理の結論発見問題の一部である SAT 問題に対する SOL に基づく SAT ソルバーを提案した．提案 SAT ソルバーの改良として，バックトラックのアルゴリズムの改良が考えられる．

表 2: 共通に解けた問題に対する速度

| 問題の種類 | Minisat-2.0 | | 提案ソルバー | |
|---|---|---|---|---|
| | 単位伝搬 | 決定 | 単位伝搬 | 決定 |
| SAT(13 問) | 2467054.8 | 25147.0 | 61425.1 | 129.6 |
| UNSAT(10 問) | 1348495.7 | 6781.6 | 70264.7 | 189.3 |
| TOTAL(23 問) | 1980724.7 | 17633.9 | 65268.4 | 154.0 |

## 謝辞

本研究の一部は，科学研究費補助金 基盤研究 (A) (No.20240003) および基盤研究 (A) (No.20240016) による．

## 参考文献

[1] Peter Baumgartner, Ulrich Furbach, and Frieder Stolzenburg. Computing answers with model elimination. *Artificial Intelligence*, 90:135–176, 1997.

[2] Andrei Doncescu, Katsumi Inoue, and Yoshitaka Yamamoto. Knowledge based discovery in systems biology using CF-induction. In *Proceedings of the 20th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2007)*, volume 4570 of *Lecture Notes in Computer Science*, pages 395–404. Springer, 2007.

[3] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In *Proceedings of SAT-2003*, pages 502–518, 2003.

[4] Katsumi Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56:301–353, 1992.

[5] Katsumi Inoue. Automated abduction. In Antonis C. Kakas and Fariba Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, volume 2408 of *Lecture Notes in Computer Science*, pages 311–341. Springer, 2002.

[6] Katsumi Inoue. Induction as consequence finding. *Machine Learning*, 55(2):109–135, 2004.

[7] Katsumi Inoue, Taisuke Sato, Masakazu Ishihata, Yoshitaka Kameya, and Hidetomo Nabeshima. Evaluating abductive hypotheses using an em algorithm on bdds. In *Proceedings of IJCAI-09*, 2009. (to appear).

[8] Koji Iwanuma, Katsumi Inoue, and Ken Satoh. Completeness of pruning methods for consequence finding procedure SOL. In *Proceedings of FTP-2000*, pages 89–100, 2000.

[9] Davis M., Logemann G., and Loveland D. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, 1962.

[10] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of DAC-01*, pages 530–535, 2001.

[11] Hidetomo Nabeshima, Koji Iwanuma, Katsumi Inoue, and Oliver Ray. Solar: An automated deduction system for consequence finding. *Journal of AI Communications*, 23(2–3):183–203, 2010.

[12] Gomes C. P., Selman B., and Kautz H. A. Boosting combinatorial search through randomization. In *AAAI-98*, pages 431–437, 1998.

[13] Knot P. and Adnan D. A lightweight component caching scheme for satisfiability solvers. In *SAT'07 Proceedings of the 10th international conference on Theory and applications of satisfiability testing*, pages 294–299, 2007.

[14] J. P. M. Silva and K. A. Sakallah. Grasp-a search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48:506–521, 1999.