1J3-3

結論発見手続き SOL タブロー計算法の分割統治法に基づく効率化

A divede and concuer strategy for consequence finding procedure SOL tableau calculus

寄特勇紀[*1] 鍋島英知[*2]
Yuki KIDOKU 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 divide and couquer strategy for a consequence finding procedure SOL tabuleau caluculs and effectiveness of our method. Consequence finding is a useful in many applications of Artificial Intelligence. In general, consequence finding requires exhaustive search process to find out all the interesting consequences. To enumerate consequences, SOL tableau calculus sometimes causes the repeated computation of sub-tableaux. The experimental results show the effectiveness of the proposed approach.

1. はじめに

本稿では，一階述語論理の結論発見手続き SOL タブロー計算法 [1] の分割統治法に基づく効率化手法を提案する．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と呼ぶ．結論発見は演繹・帰納・発想推論を実現する手段として，多くの応用分野に適用できる有用な枠組みである．その中で最もよく知られている手続きの一つが SOL 導出 [2] であり，SOL タブロー計算法 [1] は SOL 導出をタブロー計算法で再定式化したものである．

本研究では，SOL タブロー計算法における部分タブローの繰り返し計算を避けるため，分割統治法に基づく新しい探索戦略を示し，その正当性を証明する．また本手法の導入により，効率改善が得られることを示す．

2. SOL タブロー計算法 [1]

本章では SOL タブロー計算法とその問題点を述べる．

2.1 SOL タブロー計算法の定義

まず諸定義を行う．述語とその否定を**リテラル**という．また，リテラルの選言を**節**という．**生成領域** $\mathcal{P}$ とは求めたい結論 (**特徴節**という) の条件であり，リテラルの集合 L と特定の条件 $Cond$ によって定義される．ある節 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 $Cond$ を満たしているとき，C は $\mathcal{P}$ に属するという．また，節集合 Σ に対して，Σ 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．また，$\mu\Sigma$ は Σ 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 Σ，生成領域を $\mathcal{P}$ とすると，$\mathcal{P}$ に関する Σ 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，生成領域に属する節で包摂のもとで閉じている節が特徴節である．

節タブロー T とはラベル付き順序木であり，T の根以外の全てのノードはリテラルでラベル付けされている．本稿では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 $L_1, ..., L_n$ であるとき，節 $L_1 \vee ... \vee L_n$ を**タブロー節**と呼ぶ．根の直下のタブロー節は**先頭節**と呼ぶ．**連結タブロー**とは，根以外の全ての葉でないノード 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**と呼ぶ．T が**解けた**とは，T の全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 ϕ は，タブローからサブゴールへの写像である．

SOL タブロー計算法を以下で定義する．Σ を節集合，C を節，$\mathcal{P}$ を生成領域，ϕ を選択関数とする．$\Sigma + C$ と $\mathcal{P}$ からの ϕ による節 S の **SOL 演繹**は，次の条件を満たすタブローの列 $T_0, T_1, ..., T_n$ 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 ϕ によりサブゴール K を選択する．次に，以下のいずれかの操作を適用し，T_{i+1} を得る．
   (a) **Skip:** もし，$skip(T_i) \cup \{K\}$ が，$\mathcal{P}$ に属するなら，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 θ 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 θ を適用する．
   (c) **Extension:** $\Sigma \cup \{C\}$ から節 B を選択し，変数の名前変えをした亜種を $B' = L_1, ..., L_m$ をとする．もし $\neg K$ とリテラル L_j が θ によって単一化可能ならば，サブゴール K に新しい子 $L_1, ..., L_m$ を付加し，L_j には *closed* をラベル付けする．得られたタブローに θ を適用する．

連絡先: 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，〒 400-8511 山梨県甲府市武田 4-3-11, E-mail：kidoku@nabelab.org

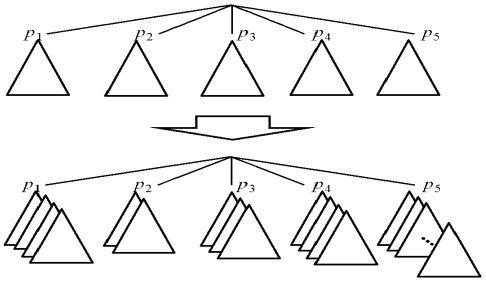

図 1: 探索空間の爆発

(d) **Reduction:** もし，K が祖先ノードに L を持ち，$\neg K$ と L が θ 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 θ 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. [1]
健全性: もし $\Sigma + C$ と $\mathcal{P}$ からの ϕ による節 S の SOL 演繹が存在するのならば，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 ϕ による SOL 演繹が存在する．

2.2 問題点：部分タブローの繰り返し計算

SOL タブロー計算法では，タブローの分岐を順番に解いていく．例として図 1 を考える．なお，三角形は解けた部分タブローを表すものとする．また，タブローの分岐は左から解かれていく (選択関数 ϕ は最左のサブゴールを選択する) ものとする．ここで，分岐 P_1 から分岐 P_5 までの全ての分岐を解くことができた場合，タブロー全体が解けたことになり，結論 $skip(T)$ を出力する．次に，別の解を探索するため P_5 の分岐に対して行った探索をリセットし，別の部分結論を求めて探索する．このような処理を繰り返し，分岐 P_5 について全ての可能性を網羅的に探索した後，分岐 P_4 までバックトラックして分岐 P_4 について別の部分結論を探索する．その後，再び分岐 P_5 について全ての部分結論を求めていく．これは，P_4 の解き方によって P_5 の解が異なってくることがあるからである．以上の様な処理を，分岐 P_1 から分岐 P_5 の全ての解き方の組み合わせを試すまで繰り返す．ここで，分岐 P_1 から分岐 P_5 がそれぞれ 10 通りの解き方を持つと仮定すると，最悪で 10^5 通りの解き方を試す事になる．この様に，SOL タブロー計算法では分岐が増えると考慮すべきタブローの解き方が増加し，繰り返し計算によって探索空間が指数関数的に爆発してしまう可能性がある．

3. SOL タブロー計算法における分割統治法

2.2 節で紹介した部分タブローの繰り返し計算を回避するため，SOL タブロー計算法に分割統治法を適用する．なお，本稿では結論発見問題の特殊形である反駁発見問題に対して分割統治法の導入を検討する．まず，タブローの分岐を分割してそれぞれを独立して探索し，各サブタブローから得た部分問題の解を合成して全体の解を得る．ここで部分問題の解は，代入により表すことができる．代入については 3.1 節で述べる．

従来の SOL タブロー計算法では，バックトラックを繰り返しながら，各サブゴールを何度も繰り返し解き直す必要がある．一方，分割統治法を適用した場合，各サブゴールを一度全解探索するだけでよくなり，同じサブゴールを何度も解き直すことを避けることが可能になる．その結果，サブゴールの繰り返し計算による探索空間の爆発を回避することができる．しかしながら，分割統治を行わない場合は代入が適用された特殊なサブゴールを解くのに対し，分割統治法を適用した場合はもっとも一般的なサブゴールを解くことになる．よって，サブゴールを探索する際の探索空間が増大してしまう可能性がある．

3.1 代入

部分問題の解は代入によって定義される．代入とは，変数と表現の関係付けの有限の集合である．例として以下を考える．

$$\{X/A,\ Y/b,\ Z/f(c)\}$$

この例では，変数 X に変数 A，変数 Y に定数 b，変数 Z に関数 $f(c)$ が関係付けられている (代入されている)．ここで，表現が変数に関係付けられることを**束縛する**といい，表現によって束縛された変数を**束縛変数**という．変数は高々一つの表現と関係付けられ，表現と関係づけられた変数はその代入のどの表現にも現れてはならない．

3.2 分割統治法のための代入合成アルゴリズム

SOL タブロー計算法に分割統治法を適用する場合，部分問題の解の合成として代入の合成を考える必要がある．しかし，AI の教科書にみられる既存の代入合成手法は，独立して解いた部分タブローの解を合成する分割統治法には適さない．よって，分割統治法のための代入合成アルゴリズムを提案する．代入合成アルゴリズムを図 2 に示す．アルゴリズムは，代入を合成する関数 Compose と 合成後の代入が代入の条件に反していないかを検査する Check に分かれている．例として以下の二つの代入 γ と τ を考える．

$$\gamma = \{X/f(a),\ Y/Z\}$$

$$\tau = \{X/f(Z)\}$$

γ と τ を既存の合成手法で合成すると以下のようになる．

$$\gamma \circ \tau = \{X/f(a),\ Y/Z\}$$

一方，提案する合成アルゴリズムで合成した結果は以下のようになる．

$$\gamma \bullet \tau = \{X/f(a),\ Y/a,\ Z/a\}$$

γ の要素 $X/f(Y)$ と τ の要素 $Y/f(a)$ は Y/a で単一化可能である．よって，二つの要素から Y/a を求め，γ に合成する．

3.3 代入合成アルゴリズムの正当性

従来の SOL タブロー計算法で得られる解集合を Θ，部分タブローを解いて得られる解集合を Γ，T とする．提案した代入合成アルゴリズムの正当性は以下で示される．

定理 2
健全性： $\forall \gamma \in \Gamma, \forall \tau \in T(\exists \gamma \bullet \tau \rightarrow \exists \theta \in \Theta \quad \theta = \gamma \bullet \tau)$
完全性： $\forall \theta \in \Theta(\exists \gamma \in \Gamma, \tau \in T \quad \theta = \gamma \bullet \tau)$
健全性の証明
証明は，$\gamma \in \Gamma$ と $\tau \in T$ に共通して現れる束縛変数の数を n として数学的帰納法を用いて行う．以下のような節を考える．

$$p(X_1, ..., X_n) \vee q(Y_1, ..., Y_m)$$

$n=0$ の場合は，自明であるため割愛する．$n=k-1$ まで健全と仮定する．$n=k$ の場合，γ と τ に共通して現れる束縛変数 k 個のうちの一つを X_j とする．このとき，$\{X_j/s\} \in \gamma$, $\{X_j/u\} \in \tau$ とする．また，$\{X_j/v\} \in \gamma \bullet \tau$ とする．ここで，$p(X_1,...,X_n)$ と $q(Y_1,...,Y_m)$ に引数としてあらわれる X_j を v で置換した問題を考える．$\gamma \bullet \tau$ が存在することから X_j を v に置換してもそれぞれの分岐は解ける．このときの解を $\gamma' \bullet \tau'$ とする．ここで，

$$\gamma \bullet \tau = \gamma' \bullet \tau' \cup \{X_j/v\}$$

と書ける．このとき $n=k-1$ なので帰納法の仮定より，従来手法で解いても

$$\theta' = \gamma' \bullet \tau'$$

となる解 θ' が存在する．ここで，置換を行う前の $p(X_1,...,X_n)$ と $q(Y_1,...,Y_m)$ を従来手法で解いた時を考えると，X_j を v に置き換えて解けているので，$\theta' \cup \{X_j/v\}$ となる解は明らかに存在する．このとき，

$$\gamma \bullet \tau = \gamma' \bullet \tau' \cup \{X_j/v\} = \theta' \cup \{x_j/v\} = \theta$$

と書くことができるので $n=k-1$ まで健全なとき，$n=k$ の場合でもアルゴリズムは健全である．□

完全性の証明

証明は，$p(X_1,...,X_n)$ と $q(Y_1,...,Y_m)$ の共通の引数の内 $\theta \in \Theta$ に束縛変数として現れる変数の数を n とし，数学的帰納法を用いて行う．健全性の場合と同様に以下のような節を考える．

$$p(X_1,...,X_n) \vee q(Y_1,...,Y_m)$$

$n=0$ の場合は自明なため割愛する．$n=k-1$ まで完全と仮定する．$n=k$ の場合，$p(X_1,...,X_n)$ と $q(Y_1,...,Y_m)$ の共通の引数のうち，$\theta \in \Theta$ に束縛変数として現れる k 個の変数のうち一つを X_j とする．また，$\{X_j/v\} \in \theta$ とする．ここで，$p(X_1,...,X_n)$ と $q(Y_1,...,Y_m)$ に引数としてあらわれている X_j を v に置換した問題を考える．仮定より θ が存在し，$\{X_j/v\} \in \theta$ であるので，あらかじめ代入しても明らかに解ける．このときの解を θ' とする．ここで，

$$\theta = \theta' \cup \{X_j/v\}$$

と書ける．このとき $n=k-1$ であるので，帰納法の仮定より提案手法で解くと

$$\gamma' \bullet \tau' = \theta'$$

となるような解 $\gamma' \bullet \tau'$ が存在する．ここで，$\{X_j/s \in \gamma\}$, $\{X_j/u \in \tau\}$ とする．X_j を v に置き換えて $\gamma' \bullet \tau'$ が存在することから，

$$v = s\lambda(\gamma' \bullet \tau') = u\lambda(\gamma' \bullet \tau')$$

となるような λ が存在する．ここで，

$$\gamma \bullet \tau = (\gamma' \bullet \tau') \cup \{X_j/s\lambda(\gamma' \bullet \tau')\}$$

と書ける．よって，

$$\gamma \bullet \tau = (\gamma' \bullet \tau') \cup \{X_j/s\lambda(\gamma' \bullet \tau')\} = \theta' \cup \{X_j/v\} = \theta$$

と書くことができるので，$n=k-1$ まで完全なとき，$n=k$ の場合も完全である．□

Algorithm 1 Compose

Input: Substitution γ, τ.
Output: Composition result, or Φ.

$\gamma = \{v_1/t_1, \ldots, v_n/t_n\}$ $\tau = \{w_1/u_1, \ldots, w_m/u_m\}$
$\tau_h = \{w_1/u_1\}$ $\tau_t = \{w_2/u_2, \ldots, w_m/u_m\}$
$\tau'_h = \mathbf{Check}(\tau'_h\gamma)$
if $\tau'_n = \Phi$ **then**
  return Φ
end if
$\gamma' = \mathbf{Check}(v_1/t_1\tau'_h, \ldots, v_n/t_n\tau'_h)$
if $\gamma' = \Phi$ **then**
  return Φ
end if
if $\exists i(w_1 = v_i)(1 \le i \le n)$ **then**
  $\lambda = \mathrm{mgu}(u_1, t_i)$
  if $\lambda = \Phi$ **then**
    return Φ
  end if
  $\gamma' = \gamma' \circ \lambda$
else
  $\gamma' = \gamma' \cup \tau'_h$
end if
return **Compose**(γ', τ_t)

Algorithm 2 Check

Input: Substitution σ.
Output: checked substitution σ, or Φ.

$\sigma = \{x_1/y_1, \ldots, x_n/y_n\}$
for all $\{x_i/y_i\} \in \sigma$ **do**
  CHECK;
  if $x_i = y_i$ **then**
    remove $\{x_i/y_i\}$ from σ
    continue
  else if y_i contains x_i **then**
    return Φ
  else if y_i contains x_j **then**
    $y_i := y_i\{x_j/y_j\}$
    goto CHECK;
  end if
end for
return σ

図 2：代入合成アルゴリズム

4. 評価実験

SOL タブロー計算法を実装した結論発見システム SOLAR [3] に分割統治法を実装し，評価実験を行った．ベンチマーク問題として TPTP-v3.5.0 を用い，通常の SOLAR と解けた問題数を比較した．実験に使用したマシンは Mac mini(Core 2 Duo 1.83GHz，2GB RAM) である．また，1 問あたり 1 分の制限時間を設けた．実験は，分割する分岐の深さや分岐数が結果に影響するか調べるため以下の 4 種類を行った．

- 深さが 1 以下の分岐（先頭節）を分割.
- 深さが 2 以下の分岐を分割.
- 全ての分岐を分割.
- 長さが 4 以上である全ての分岐を分割.

実験結果を表 1 に示す．表 1 は，それぞれの手法でどれだけの問題を解くことができたかを表している．各列は問題を解く

表 1: SOLAR との比較

	SOLAR	depth ≤ 1	depth ≤ 2	all nodes	all nodes len ≥ 4
♯ of solvedunit EQ	162	162	154	141	162
♯ of solved non-unit EQ	696	656	547	647	666
♯ of solved non-EQ	1249	1249	1228	1173	1244
Total	2107	2067	2029	1961	2072

表 2: 並列処理シミュレーション

	SOLAR	depth $\leq 50\%$ ♯ substs ≤ 500	depth $\leq 70\%$ ♯ substs ≤ 700	depth $\leq 70\%$ ♯ substs ≤ 2000 len ≥ 4
♯ of solved unit EQ	162	182	217	162
♯ of solved non-unit EQ	696	755	740	697
♯ of solved non-EQ	1249	1227	1230	1253
Total	2107	2156	2187	2112

のに用いた手法であり，最も左の列が SOLAR，右の 4 列が分割統治法である．各行は問題の種類を表し，unit EQ は単位節のみで構成された等号を含んだ問題である．non-unit EQ は unit EQ 以外の等号を含んだ問題である．non-EQ は等号を含まない問題である．“Total” は解けた問題の総数を表している．表 1 より，どの分割方法を用いても分割統治法のほうが解けた問題数が少ないことが分かる．特に，タブローの分岐を分割すればするほど解けた問題数が少なくなっている．よって，タブローの分岐を過剰に分割することは適当ではないといえる．さらに，従来の SOLAR は反駁を 1 つ発見したら処理を停止することが可能であるのに対し，分割統治を適用した場合は，各サブゴールについて全解探索をした後に部分問題の解集合を統合するため，必ず全解探索を行う. このことも原因であるといえる．

分割統治法を適用した場合，各サブゴールを独立して解くことが可能である．すなわち，本手法は並列処理に向く．そこで，並列処理を適用した提案手法をシミュレートするためもうひとつの実験を行った．

シミュレーションでは各ノードが保持する代入の数を制限し，保持する代入が制限を超えたら処理を打ち切ることで並列処理を模倣している．実験は，以下の 3 つの制限を用いて行った．

- 深さ 50 % 以下の分岐を分割し，ノードが保持する代入は 500 以下
- 深さ 70 % 以下の分岐を分割し，ノードが保持する代入は 700 以下
- 深さ 70 % 以下で長さが 4 以上の分岐を分割し，ノードが保持する代入は 2000 以下

実験結果を表 2 に示す．表 2 より，一部を除いて提案手法のほうが SOLAR より多くの問題を解いていることが分かる．また，どの条件でも “Total” では SOLAR より多くの問題を解いている．この結果から，並列処理を適用することで，分割統治法によるいっそうの効率化が見込める．さらに，解けた問題数が少なくなっている部分は，どちらも non-EQ 問題である．EQ 問題は non-EQ 問題に比べて，探索の際にタブローが複雑になりやすいという特徴がある．このことから，提案手法は等号を含む問題を解く際により効果を発揮すると考えられる．

5. まとめと今後の課題

本研究では分割統治法による SOL タブロー計算法の効率化手法を提案した．また，分割統治法のための代入合成アルゴリズムを提案し正当性を証明した．さらに, 実験結果より分割統治法の適用による効率改善が見込めることを示した．

定理証明問題は反駁を発見することを目的とする．定理証明問題では，反駁を 1 つでも発見すれば探索を直ちに終了することができる．しかし，分割統治法を適用した場合，定理証明問題であったとしても，各サブゴールについて全解探索をした後に部分問題の解集合を統合するため，全解探索をすることになる．結論発見問題は基本的に全解探索を必要とするので，分割統治法は結論発見問題を解く場合により効果的だと考える．よって，分割統治法で結論発見問題への拡張は今後の重要な課題の 1 つである．また，実験結果より並列処理の適用による効率化の可能性が示されたことから，並列処理の実現も今後の課題である．

謝辞

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

参考文献

[1] Koji Iwanuma, Katsumi Inoue, and Ken Satoh, A Tableau-based Reconstruction of Consequence Finding Procedure SOL. *IEICE technical report*, Vol. 100, No. 321, pp.17-24. 2000.

[2] Katsumi Inoue, Linear resolution for consequense finding. *Artificial Intelligence*, Vol.56,pp.301-353, (1992).

[3] Hidetomo Nabeshima, Koji Iwanuma, Katsumi Inoue, and Oliver Ray, SOLAR: An automated deduction system for consequence finding, *AI Communications*, Vol. 23, No. 2-3. pp.183-203, 2010.