Vol.2017-AL-163 No.3
2017/5/12

決定グラフを用いた文脈自由文法の構文木集合の表現

網井 圭[1,a)]　西野 正彬[2]　山本 章博[1]

概要：本稿では，文脈自由文法の構文木集合を決定グラフで表した際の決定グラフの大きさについて考察する．文脈自由文法は記号列の背後にある構造を表現するモデルとして自然言語処理や生命情報学などの分野で広く使われる．決定グラフはブール関数を効率的に表現するためのデータ構造であり，決定グラフの大きさに比例した時間でブール関数に対する様々な操作が行えることが知られている．文脈自由文法の構文木集合はブール関数として表現することができるため，構文木集合を決定グラフで表すことで制約の追加や最良の構文木の発見などを高速に行えることが期待される．しかし，そのような決定グラフの大きさについて理論的な解析が行われていない．本稿では単純な生成規則からなる文脈自由文法における構文木集合を決定グラフを用いて表現する実験を行い，ゼロサプレス型項分岐決定グラフの大きさが他の決定グラフよりも小さくなることを示す．また，その大きさに対する理論的な上限を与える．

1. はじめに

文脈自由文法は記号列の構造を決定するモデルとして幅広く使用されている．代表的な使用例として，自然言語処理における文章の係り受け構造の解析 [1] や，生命情報学における RNA の二次構造の解析 [2] が挙げられる．

ある記号列が与えられた文脈自由文法からどのように生成されるか，すなわちどのような構文木が得られるかを求める手法として CYK(Cocke-Younger-Kasami) アルゴリズムが知られている．CYK アルゴリズムでは，記号列の長さを N とすると，動的計画法によって $O(N^3)$ の計算時間で構文木集合を生成できる．一方で，何らかの追加の制約条件を満たすような構文木の集合を求めたい場合には，CYK アルゴリズムを適用することができない．ここで，追加の制約条件とは，ある生成規則の使用回数に制限をつけたり，特定の規則の組が同時に使われないといった条件のことである．このような制約条件を用いることで，例えば自然言語処理においては，ある品詞の出現回数が制限される，特定の部分の品詞が同じであるといった背景知識を加味した構文解析を行うことができる．同様に，生命情報学においては結合位置間の距離など構造に関する背景知識が既知の場合にそれを考慮できる．本稿では決定グラフを用いることで，そのような制約をつけた解析を可能とする方法を述べる．

決定グラフとはブール関数を表現するデータ構造であり，ブール関数に対する様々な演算をグラフの大きさに比例した時間で実行できる．文脈自由文法において，どの部分記号列がどの生成規則により生成されたかを命題変数で表すことにより，ある記号列に対する構文木集合はブール関数を用いて表現することができる．そのため構文木集合を決定グラフで表現することができ，制約を追加するなどの演算や，制約を満たす最適な構文木の探索などを高速に実行することができる．

決定グラフを用いたブール関数操作の計算時間は，ブール関数を表現する決定グラフの大きさに比例することが知られている．決定グラフには二分決定図 (Binary Decision Diagram, BDD)[3]，ゼロサプレス型二分決定図 (Zero-suppressed Binary Decision Diagram, ZDD)[4]，項分岐決定図 (Sentential Decision Diagram, SDD)[5]，ゼロサプレス型項分岐決定図 (Zero-suppressed Sentential Decision Diagram, ZSDD)[6] といったいくつかの種類があり，あるブール関数を表現する際にどの決定グラフを用いると大きさが最小になるかは表現するブール関数に依存する．また，変数順序や vtree といった，決定グラフの大きさに影響を与えるパラメータが存在する．すなわち，決定グラフを用いて効率的に構文木集合を扱うためには，(1) 構文木集合を表現するのに適切な決定グラフを選び，かつ (2) その決定グラフの適切なパラメータを選択する必要がある．

本稿では上記の 4 種類の決定グラフといくつかの変数順序，vtree の組み合わせにおいて，構文木集合を表現した際

[1] 京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University, Kyoto, 606–8051, Japan
[2] NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, NTT Corporation, Japan
[a)] k.amii@iip.ist.i.kyoto-u.ac.jp

のそれぞれの決定グラフの大きさを比較し，ついて ZSDD が他の決定グラフよりも小さくなることを示した．また，実験で ZSDD の大きさを最小とした vtree に対して，その vtree を用いたときの ZSDD の大きさに対する理論的な保証を与えた．

本稿は以下のように構成される．第 2 章では準備として文脈自由文法の定義を与え，第 3 章では文脈自由文法の構文木集合をブール関数へ変換する手法について述べる．第 4 章では決定グラフについて解説し，そして第 5 章では 4 種類の決定グラフといくつかの変数順序について実験を行い，単純な文脈自由文法における構文木の集合を表す際に ZSDD が最も小さくなることを示す．第 6 章では ZSDD の大きさについて理論的な解析を行い，最後に第 7 章ではまとめと今後の展望を述べる．

2. 準備

2.1 文脈自由文法

定義 2.1 文脈自由文法 (context-free grammar, **CFG**) は，非終端記号の有限集合 V，終端記号の有限集合 Σ，生成規則の有限集合 P，開始記号 S を指定することにより規定される文法

$$G = (V, \Sigma, P, S) \tag{1}$$

のうち，生成規則が次の形式となっているものである．

$$A \to \alpha \quad (A \in V, \alpha \in (\Sigma \cup V)^*) \tag{2}$$

S から生成規則の適用を繰り返し $w \in \Sigma^*$ が生成されるとき，記号列 w は文法 G により導出されるという．

また，本稿では文脈自由文法のうち，以下の性質を満たすチョムスキー標準形に限定する．

定義 2.2 文脈自由文法 G において，どの生成規則も以下のいずれかの性質を満たすとき，文法 G はチョムスキー標準形 (Chomsky normal form, **CNF**) であるという．

- $A \to B\ C \quad (A, B, C \in V)$
- $A \to \alpha \quad (\alpha \in \Sigma)$
- $S \to \epsilon$

任意の文脈自由文法はチョムスキー標準形に変換することができ，チョムスキー標準形で表された文法は以下の CYK アルゴリズムにより構文木集合を得ることができる．

CYK アルゴリズム

CYK(Cocke-Younger-Kasami) アルゴリズムは，チョムスキー標準形の文法 G と記号列 $\alpha_1\alpha_2 \ldots \alpha_N$ が与えられたとき，記号列が文法 G により導出可能か，また導出可能な場合はどのような生成規則を用いて導出されるかを求めるアルゴリズムである．以下，記号列の i 文字目から j 文字目までからなる部分列 $\alpha_i \ldots \alpha_j$ を $\alpha_{ij} (i \leq j)$，P に含まれる j 番目の生成規則を q_j と書く．

α_{ij} に対する非終端記号 X を開始記号とする導出において，最初に利用された生成規則のインデックスの集合を $S_{ij}^{(X)}$ とする．このとき，$S_{ii}^{(X)}$ は，

$$S_{ii}^{(X)} = \{j | q_j = (X \to \alpha), \alpha = \alpha_i\} \tag{3}$$

として定義される．また，$i < j$ であるときの $S_{ij}^{(X)}$ は

$$S_{ij}^{(X)} = \bigcup_{k=i}^{j-1} \{i | r_i = (X \to YZ), S_{ik}^{(Y)} \neq \emptyset, S_{k+1,j}^{(Z)} \neq \emptyset\} \tag{4}$$

として再帰的に求めることができる．$S_{1N}^{(S)} \neq \emptyset$ であれば与えられた記号列は導出可能であり，集合に含まれるインデックスを辿ることにより構文木を構築できる．ここで，全ての $S_{ij}^{(X)}$ の計算は $O(N^3)$ 時間で実行できることが知られている．

3. 構文木からブール関数への変換

文脈自由文法の構文木集合を表すブール関数を設計するために構文木での生成規則の出現に対して命題変数 $b_{i,j,l}$ を割り当てる．終端記号の系列の i 文字目から j 文字目までからなる部分系列を α_{ij} としたとき，α_{ij} が l 番目の生成規則 q_l から生成される場合に $b_{i,j,l} = 1$ となる．

次に，各命題変数の間に制約条件を与え，正しい導出に対応する規則の集合が与えられた時に真を返すブール関数を以下のようにして設計する．それぞれの記号列 α_{ij} は高々 1 つの生成規則により導出される．そのため各命題変数 b_{ij*} のうち，二つ以上が同時に真にならない制約が必要となり，その制約を

$$H_{ij} = \bigwedge_{b_{ijk}, b_{ijl}: j \neq k} \neg b_{ijk} \vee \neg b_{ijl} \tag{5}$$

と表す．次に，各集合 $S_{ij}^{(X)}$ の規則のうちいずれかが真となる制約を，

$$F_{ij}^{(X)} = \bigvee_{k \in S_{ij}^{(X)}} b_{ijk} \tag{6}$$

とする．また非終端記号に対応する命題変数 b_{ijk} に対し，$q_k = (X \ \to \ YZ)$ であるならば，

$$D_{ijk} = \neg b_{ijk} \vee \left(\bigvee_{l=i}^{j-1} F_{i,l}^{(Y)} \wedge F_{l+1,j}^{(Z)} \right) \tag{7}$$

とする．最後に，$i < k \leq j < l$ であるような全ての b_{ijm}, b_{kln} に対して，

$$C = \bigwedge_{b_{ijm}, b_{kln}: i<k\leq j<l} \neg b_{ijm} \vee \neg b_{kln} \tag{8}$$

とする．これらの式を用いると，正しい導出に対応する規則の集合が与えられた時に真を返すブール関数は，

$$\left(\bigwedge_{i,j: 1 \leq i < j \leq N} H_{ij} \right) \wedge \left(\bigwedge_{b_{ijk}} D_{ijk} \right) \wedge C \tag{9}$$

として表現することができる．

例として，以下の文脈自由文法 $G = (V, \Sigma, P, S)$ の構文木集合を表すブール関数を設計する．$V = \{A, B, C\}$，$\Sigma = \{a, b, c\}$，P を以下とする．

- 1. $S \to A\ B$
- 2. $A \to A\ A$
- 3. $B \to B\ C$
- 4. $B \to A\ C$
- 5. $A \to a$
- 6. $A \to b$
- 7. $B \to b$
- 8. $C \to c$

abc という記号列に対する構文木集合を考える．CYK アルゴリズムにより，$S_{ij}^{(X)}$ のうち空集合でないものは以下のように求めることができる．

- $S_{11}^{(A)} = \{5\}$
- $S_{22}^{(A)} = \{6\}$
- $S_{22}^{(B)} = \{7\}$
- $S_{33}^{(C)} = \{8\}$
- $S_{12}^{(A)} = \{2\}$
- $S_{23}^{(B)} = \{3, 4\}$
- $S_{13}^{(S)} = \{1\}$

次に，それぞれの生成規則の出現に対して命題変数 $b_{115}, b_{226}, b_{227}, b_{338}, b_{122}, b_{233}, b_{234}, b_{131}$ を割り当て，制約条件を考える．

式 (5) より，$H_{22} = \neg b_{226} \vee \neg b_{227}$，$H_{23} = \neg b_{233} \vee \neg b_{234}$ となる．

式 (6) より $F_{23}^{(B)} = b_{233} \vee b_{234}$ となり，他の命題変数に対しては $F_{11}^{(A)} = b_{115}$，$F_{22}^{(A)} = b_{226}$ のようになる．

式 (7) より

- $D_{122} = \neg b_{122} \vee (F_{11}^{(A)} \wedge F_{22}^{(A)})$
- $D_{233} = \neg b_{233} \vee (F_{22}^{(B)} \wedge F_{33}^{(C)})$
- $D_{234} = \neg b_{234} \vee (F_{22}^{(A)} \wedge F_{33}^{(C)})$
- $D_{131} = \neg b_{131} \vee (F_{11}^{(A)} \wedge F_{23}^{(B)}) \vee (F_{12}^{(A)} \wedge F_{33}^{(B)})$

となる．(ここで $S_{33}^{(B)}$ は空なので $F_{33}^{(B)} = 0$ である．)

最後に式 (8) より $C = (\neg b_{122} \vee \neg b_{233}) \wedge (\neg b_{122} \vee \neg b_{234})$ となる．

以上の H_{ij}, D_{ijk}, C を用いることで式 (9) によりブール関数として構文木集合を表現できる．このブール関数が真となるのは $b_{115} = 1, b_{226} = 0, b_{227} = 1, b_{338} = 1, b_{122} = 0, b_{233} = 1, b_{234} = 0, b_{131} = 1$ のときのみであるが，これは図 1 が示す通り，正しい構文木に対応する．

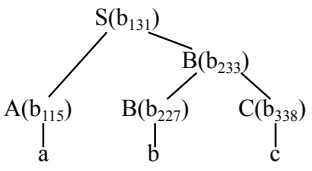


図 1 正しい構文木．括弧内はそれぞれの生成規則と対応する命題変数を表す．

4. 決定グラフ

紙面の都合上，以下では決定グラフのうち ZSDD を紹介する．なお，ZSDD は集合族を表現する決定グラフとして導入されたが，集合族とブール関数は等価であるため，ZSDD はブール関数を表現するためにも使える．

4.1 (X, Y)-分解

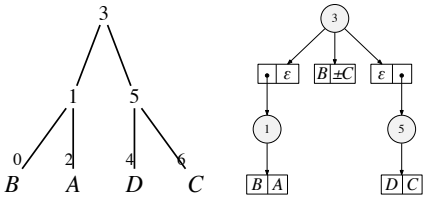
$(\mathbf{X}, \mathbf{Y})$-分解は与えられた集合族を小さな集合族に分割する手法であり，決定グラフを構成する重要な技術である．f を集合族，$\mathbf{X}, \mathbf{Y}$ をそれぞれ要素の集合であり，全体集合の分割となっているとする．集合族 f は，$\mathbf{X}, \mathbf{Y}$ を全体集合とする集合族 $p_i(\mathbf{X}), s_i(\mathbf{Y})$ によって，

$$f = [p_1(\mathbf{X}) \sqcup s_1(\mathbf{Y})] \cup \cdots \cup [p_n(\mathbf{X}) \sqcup s_n(\mathbf{Y})],$$

として表現される．記号 $\cup, \sqcup$ は集合族に対する和，結合演算を表し，それぞれ $f \cup g = \{a \mid a \in f \text{ かつ } a \in g\}, f \sqcup g = \{a \cup b \mid a \in f \text{ かつ } b \in g\}$ として定義される．$p_1, \ldots, p_n$ を $(\mathbf{X}, \mathbf{Y})$-分解における主部，$p_1, \ldots, p_n$ を副部と呼ぶ．もし主部が排他的 (すべての $i \neq j$ について $p_i \cap p_j = \emptyset$) かつ網羅的 ($\bigcup_{i=1}^{n} p_i$) かつ無矛盾 (すべての i について $p_i \neq \emptyset$) であるならば，その $(\mathbf{X}, \mathbf{Y})$-分解を $(\mathbf{X}, \mathbf{Y})$-分割と呼び，$(p_1, s_1), \ldots, (p_n, s_n)$ として表現する．さらに，すべての $i \neq j$ について $s_i \neq s_j$ を満たすならば $(\mathbf{X}, \mathbf{Y})$-分割は圧縮されていると呼ぶ．例えば，集合族 $\{\{A, B\}, \{B\}, \{B, C\}, \{C, D\}\}$ の $\mathbf{X} = \{A, B\}, \mathbf{Y} = \{C, D\}$ としたときの $(\mathbf{X}, \mathbf{Y})$-分割は

$$[\{\{A, B\}\} \sqcup \{\emptyset\}] \cup [\{\{B\}\} \sqcup \{\emptyset, \{C\}\}] \cup [\{\emptyset\} \sqcup \{\{C, D\}\}],$$

である．ここで $\{\{A, B\}\}, \{\{B\}\}, \{\emptyset\}$ が主部であり，$\{\emptyset\}, \{\emptyset, \{C\}\}, \{\{C, D\}\}$ が副部である．



(a) A vtree (b) A ZSDD

図 2 (a) vtree の例，(b) (a) の vtree を参照し，集合族 $\{\{A, B\}, \{B\}, \{B, C\}, \{C, D\}\}$ を表現する ZSDD

4.2 vtree

$(\mathbf{X}, \mathbf{Y})$-分割と並んで ZSDD を構成する重要な概念である *vtree* を導入する．ZSDD は集合族に対して再起的に $(\mathbf{X}, \mathbf{Y})$-分割を適用することで有向非巡回グラフの形に集合族を分解して表現する手法である．すなわち，与えられた集合族を，$(\mathbf{X}, \mathbf{Y})$-分割によって $\mathbf{X}, \mathbf{Y}$ を全体集合とする集合族 $p_1, \ldots, p_n, s_1, \ldots, s_n$ に分解し，さらにそれらを $(\mathbf{X}, \mathbf{Y})$-分割によって部分集合族に分割する…という手続きを表現したものが ZSDD である．vtree は再起的な $(\mathbf{X}, \mathbf{Y})$-分割の順序を与えるものであり，ある vtree に沿った $(\mathbf{X}, \mathbf{Y})$-

分割を表現する ZSDD を作成すると，一意な ZSDD を構成できる．vtree は各節が必ず 2 つの子を持つような根付き二分木であり，vtree の各葉が全体集合に含まれる各要素に対応している．図 2(a) に vtree の例を示す．図中の葉は対応する変数を表現しており，節の数字は ID を表している．各節には一意な ID が付与されている．

vtree の節は，全体集合に含まれる要素を，左側の子を根とする木に含まれる要素の集合と右側の子を根とする木に含まれる要素の集合の 2 つの集合に分割しているとみなすことができる．図中の根 v_3 は，$\mathbf{X} = \{A, B\}$ かつ $\mathbf{Y}$ であるような $(\mathbf{X}, \mathbf{Y})$-分割を表している．同様に v_3 の左側の子 v_1 も，v_1 を根とする部分木において $\mathbf{X} = \{B\}$ かつ $\mathbf{Y} = \{A\}$ であるような $(\mathbf{X}, \mathbf{Y})$-分割を表している．以下では v^l, v^r がそれぞれ v の左側の子，右側の子を表しているものとする．また，それぞれの頂点の根とする部分木を表すためにも v^l, v^r を用いる．もし v^l が葉であるとき，v をシャノンノードと呼び，そうでない場合に分解ノードと呼ぶ．図 2(a) の根は分解ノードであり，その子はどちらもシャノンノードである．また，全ての頂点がシャノンノードであるとき，その vtree を *right-linear-vtree* と呼ぶ．

なお，以下では入力グラフ，vtree，ZSDD の 3 種類のグラフを扱うため，混乱を避けるために vtree の頂点を vnode と呼び，$v, v^l, v^r, v_1, v_2, \ldots$ などと表す．同様に，入力グラフの頂点は gnode と呼び，$u, u_1, u_2, \ldots$ などと表す．後述する ZSDD の頂点は znode と呼び，$z_1, \ldots, z_n$ などと表す．

4.3 ゼロサプレス型項分岐決定図 (ZSDD)

ZSDD を以下のように再帰的に定義する．なお，α を ZSDD とし，α が表現する集合族を $\langle\alpha\rangle$ とする．

定義 4.1 α は以下のいずれかの条件を満たすとき，*vnode* v を参照する *ZSDD* と呼ぶ．

- $\alpha = \epsilon$ または $\alpha = \bot$．(解釈: それぞれ $\langle\epsilon\rangle = \{\emptyset\}$, $\langle\bot\rangle = \emptyset$)
- $\alpha = X$ または $\alpha = \pm X$ かつ v が要素 X に対応する *vtree* の葉．(解釈: それぞれ $\langle X\rangle = \{\{X\}\}$, $\langle\pm X\rangle = \{\{X\}, \emptyset\}$)
- $\alpha = \{(p_1, s_1), \ldots, (p_n, s_n)\}$，$v$ は *vnode*，$(p_1, \ldots p_n)$ がそれぞれ部分 *vnode* v^l を参照する *ZSDD*，$s_1, \ldots s_n$ が *vnode* v^r を参照する *ZSDD*，かつ $\langle p_1\rangle, \ldots \langle p_n\rangle$ が $(\mathbf{X}, \mathbf{Y})$-分割の条件を満たす集合族に対応している．(解釈: $\langle\alpha\rangle = \bigcup_{i=1}^{n} \langle p_i\rangle \sqcup \langle s_i\rangle$)

ここで，$\epsilon, \bot, X, \pm X$ を終端 ZSDD と呼ぶ．終端 ZSDD でない ZSDD は，vnode v によって表されている $(\mathbf{X}, \mathbf{Y})$-分割 $\{(p_1, s_1), \ldots, (p_n, s_n)\}$ を表しており，決定ノードと呼ぶ．図 2(b) は図 2 の vtree を参照し，集合族 $\{\{A, B\}, \{B\}, \{B, C\}, \{C, D\}\}$ を表す ZSDD である．図中の円ノードと，その子ノードである四角いノードとあわせて決定ノードを表現している．円ノード中の数字はその決定ノードが参照している vnode の ID である．四角い子ノードは $(\mathbf{X}, \mathbf{Y})$-分割に含まれる主部，副部のペアを表現しており，左側が主部，右側が副部である．

$(\mathbf{X}, \mathbf{Y})$-分割によって生成された ZSDD を，$\mathbf{X}$ に関する部分関数と呼ぶ．ZSDD の大きさを，その ZSDD に含まれる主部と副部のペアの個数と定義する．図 (2)(b) において 3 が付されたノードは $\{B, A\}$ に関する部分関数であり，この ZSDD の大きさは 5 となる．

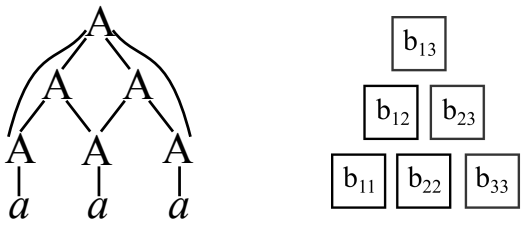
5. 実験

決定グラフのうち，BDD，SDD，ZDD，ZSDD を用いて以下に示す単純な文脈自由文法の構文木を表現した場合の大きさを比較する．BDD，ZDD は単一の変数についての分解に限定したものであり，それぞれ SDD，ZSDD で right-linear-vtree を与えたときの結果と一致する．

$$A \rightarrow AA$$
$$A \rightarrow a$$

A を開始記号かつ非終端記号，a を終端記号とし，記号列の長さ n を変化させたときの決定グラフの大きさを記す．なお，本稿で考える文法において非終端記号・終端記号ともに生成規則は一つしかないので b_{ijl} を b_{ij} と表す．

vtree を考えるにあたって，構文木における生成規則の出現場所に注目した．構文木と命題変数の対応を図 3 に示す．



(a) 構文木で使われ得る生成規則　(b) 対応する命題変数

図 3　生成規則と命題変数の対応

本実験では以下の 4 種類の vtree を与えた．それぞれ記号列の長さ 3 の場合を図 6 に示す．

vtree1　命題変数 b_{ij} について，$j - i$ の値が大きいものから順に一つずつ分解する right-linear-vtree．$j - i$ の値が等しいものは i の値が小さい順に分解する．図 3 で構文木の根に近い方から順に変数を並べることに対応する．

vtree2　$j - i$ の値が等しい変数について i の値が小さい順に right-linear-vtree を作り，それらを $j - i$ の値が大きいものから順に結合したもの．図 3 の構文木における高さが等しい変数をまとめて分解する．

	n	3	4	5	6	7	8	9	10
BDD	vtree1	14	47	92	223	416	901	1164	2087
ZDD	vtree1	6	15	34	78	177	395	874	1914
SDD	vtree2	12	44	106	202	419	803	1346	2558
	vtree3	10	40	112	227	458	929	2086	2174
	vtree4	12	42	96	218	425	723	1443	2735
ZSDD	vtree2	6	15	34	78	173	**381**	**825**	**1772**
	vtree3	6	14	32	75	173	390	868	1907
	vtree4	8	22	53	120	263	566	1205	2548

表 **1**　記号列の長さを変化させた場合の決定グラフの大きさ

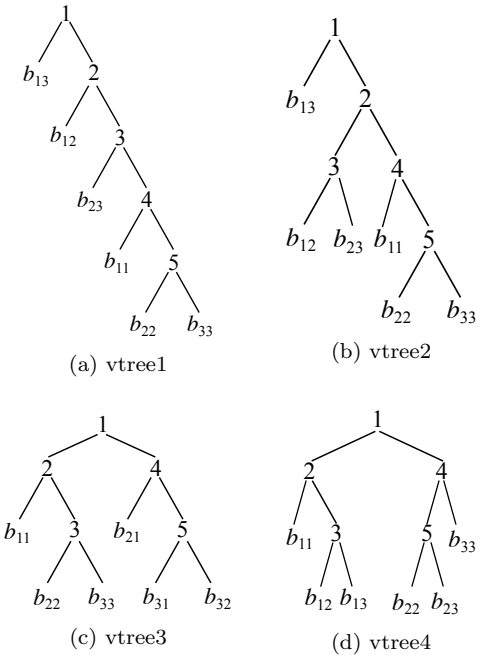


図 **4**　記号列の長さ 3 の場合の vtree

vtree3　命題変数を終端記号に対応するものと非終端記号に対応するものの 2 つに分け, それぞれ別個に vtree1 と同じ順番で right-linear-vtree を作る．vtree の根ノード v において，v^l が終端記号に，v^r が非終端記号に対応するように結合する．

vtree4　i の値が等しいものをまとめ，j が大きい順に right-linear-vtree を作る．それらを i が小さいものが根に近くなるように結合したものである．図 3 の左側から順に right-linear-vtree を作り結合することに対応する．

BDD，ZDD には vtree1 を，SDD，ZSDD には vtree2,3,4 をそれぞれ与える，

実験の結果を表 1 に示す．n が大きくなった場合，ZSDD と vtree2 の組み合わせが最も大きさが小さくなることがわかる．

6. ZSDD の大きさ

上の実験で最も小さかった ZSDD と vtree2 の組み合わせについて，n の値を変化させた時の ZSDD の大きさに対する上限を与える．

解析に入る前に，以下の説明で用いる概念をいくつか定義する．b_{ij} は，構文木において i 文字目から j 文字目までの $j-i+1$ 文字を導出する最初の規則に対応する変数である．$j-i+1$ を変数 b_{ij} の高さと呼ぶ．また，高さが r であるような命題変数 b_{ij} の集合を B_r とする．例えば長さ 5 の記号列に対する導出を考える場合，$B_3 = \{b_{13}, b_{24}, b_{35}\}$ である．一般に $|B_r| = n - r + 1$ となる．vtree2 の根から右側の枝をたどることで到達できる最初の $n-1$ 個の vnode は，その左側の子に含まれる命題変数の集合が B_r に一致している．例えば根にあたる vnode の左側の子は $B_n = \{b_{1n}\}$ に対応し，根の右側の子である vnode の左の子に含まれる命題変数の集合は B_{n-1} に対応する．このように，vtree2 では命題変数の集合をその高さごとに分解していると解釈することができる．

以下では，(1) 決定ノードの数の上限を求め，かつ (2) 各決定ノードの個ノードの数の上限を求めることで，ZSDD の大きさの上限を求める．

ZSDD 中の各決定ノードは，表現するブール関数のある部分関数を表現している．また，ZSDD の一意性より，ある ZSDD 中には等価な部分関数に対応する決定ノードは高々 1 つしか存在しないことが知られている．これらの性質を利用し，構文木集合を表現する ZSDD 中に出現する，異なる部分関数の個数の上限を見積もることで ZSDD の決定ノードの大きさの上限を与える．具体的には，決定ノードは vtree のいずれかの中間ノードに対応づけられることを利用し，vnode ごとにそれに対応づけられる決定ノードの数の上限を求めることで決定ノードの総数の上限を導出する．上限の求め方は vnode の種類によって異なる．すなわち，対応づけられる vnode の左側の子が B_r と一致する場合（高さ分解 vnode とよぶ）と，それ以外の場合とで異なる方略で上限を求める．図 6(b) では vnode1,2 が高さ分解 vnode に当たる．決定ノードの子ノードの個数についても，同様に vnode の種類ごとに異なる方法で上限を求める．以下に順に説明する．

高さ分解 vnode に対する決定ノード

vnode を参照する決定ノードは，高さ r 以下の命題変数の組合せを表すある部分関数を表現する．部分関数は，高

さが $r+1$ 以上の命題変数の値の割当と組み合わせて構文木を構成するような r 以下の命題変数のとり方の集合を表している．すなわち，高さが $r+1$ 以上の変数の選択に応じて，何通りの部分関数が存在するかを調べることによって，決定ノードの数を見積もることができる．

部分関数の種類数の上限を見積もる前に，記号列中の記号のグループ分けを以下のように定義する．いま，高さ $r+1$ 以上の命題変数の取捨が与えられたときに，各終端記号 α_i に対して，b_{jk} が $j \le i \le k$ を満たし，かつ高さが最小となるような b_{jk} をラベルとして割り当てる．$r+1$ 以上の命題変数の取捨が不正でない，すなわち高さ r 以下の命題変数との組合せで 1 つ以上の正しい構文木を与えることができるのであれば，各終端記号について必ずラベルが付与される．このラベルに基づいて終端記号をグループ分けすると以下が成り立つ．

定理 6.1 高さ $r+1$ 以上の命題変数への不正でない 2 種類の割当 I_1, I_2 が，記号列に対して等価なグループ分けをあたえるならば，それぞれの割当と組み合わせて正しい構文木の集合を与えるような高さ r 以下の 2 つの部分関数 f, g は等価である．

証明 上記の定義によるグループ分けを行うと，あるグループに属する終端記号の集合は必ず連続して出現することになる．また，終端記号 α_i, α_{i+1} が異なるグループに含まれていたとすると，正しい構文木を得るためには $j \le i < k$ であるような命題変数 b_{jk} を高さ r 以下で用いることができない．したがって，高さ r 以下の部分関数は，それぞれの連続するグループの終端記号に対する部分的な構文木の集合として定義されることになる．あるグループに対する高さ r 以下での部分的な構文木は，そのグループの作られ方によらず一致する．したがって，グループ分けが等価であるならば部分関数は等価となる． □

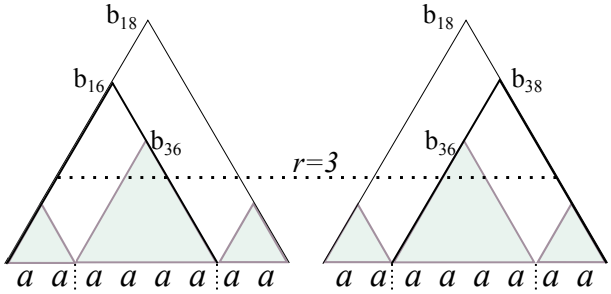


図 5 $n=8$ で 2 種類の割当に対し高さ r 以下の部分関数が等価になる例．$r=3$ としたとき，高さ $r+1=4$ 以上で左の図は $\{b_{18}, b_{16}, b_{36}\}$ を，右の図は $\{b_{18}, b_{38}, b_{36}\}$ を選択したものである．高さ $r+1$ 以上での割当は異なるが，高さ r 以下では色のついた三角形内について割当を考えることになり，部分関数が等価となる．

部分関数が等価となる例を図 5 に示す．この 2 種類は異なる構文木を表し，高さ $r+1$ 以上での変数の割当も異なるが，グループ分けは等価となる．ここで，高さ r 以下の部分関数は色のついた三角形内の部分的な構文木を表すため，2 つの部分関数は等価となる．

以上より，高さ $r+1$ 以上の変数の取捨で与えられるグループ分けの種類数から部分関数の数，すなわち決定ノードの数を見積もることができることがわかる．

次にグループ分けの種類数を考える．あるグループ分けを実現するような高さ $r+1$ 以上の変数の割当 I が存在するかどうかは，最大のグループの大きさを用いて判定することができる．最大のグループの長さについての条件を示し，それを満たすグループ分けの個数を求めることによって決定ノードの数を見積もる．

定義 6.1 各終端記号 α_i に対してラベル b_{jk} が割り当てられたとき，同じラベルを持つ最長の記号列 $\alpha_i \dots \alpha_{i+l-1}$ の文字数 l を**最大グルーピング数**と呼ぶ．

長さ n の終端記号の系列とあるグループ分けに対して，高さ $r+1$ 以上の変数の割当 I が存在する条件は最大グルーピング数を用いて表される．

補題 6.1 あるグループ分けの最大グルーピング数を l としたとき，

$$r+1 \le l \le 2r \tag{10}$$

を満たす場合に，高さ $r+1$ 以上の変数の割当 I が存在する．

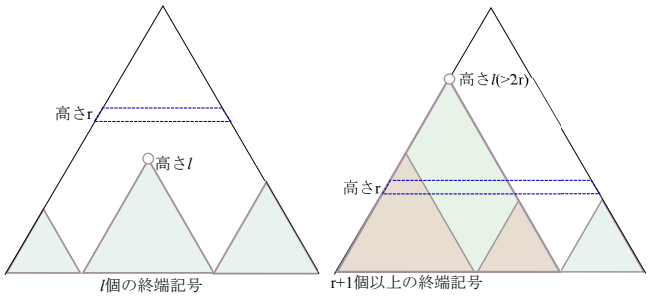
証明 最大グルーピング数が l のとき，各終端記号 α_i のラベル b_{jk} のうち最大の高さは l となる．高さ $r+1$ 以上で割当を考えるので，$r+1 \le l$ を必ず満たさなければならない．

一方，$2r < l$ の場合に不適であることを示す．$2r < l$ を満たす高さ $r+1$ 以上の割当 I が存在すると仮定する．長さ最長のグループを 2 つに分割すると，片方のグループの長さは $r+1$ 以上になる．グループの長さとラベルの高さは一致するので，このグループに含まれる終端記号に対するラベル b_{jk} の高さも $r+1$ 以上になる．これは割当 I に対する最大グルーピング数がより小さくなることを意味するため，不適である． □

次に，最大グルーピング数が l 以下となるようなグループ分けの種類数について考える．

補題 6.2 長さ n の記号列に対して最大グルーピング数が l 以下となるグループ分けの種類数を $a_{l,n}$ とおくと以下が成り立つ．

$$a_{l,n} = \begin{cases} 2^{n-1} & (n \le l) \\ \sum_{i=n-l}^{n-1} a_{k,i} & (n > l) \end{cases} \tag{11}$$



(a) $l < r+1$ の場合　　(b) $2r < l$ の場合

図 6　式 (10) を満たさないグルーピングの例

証明　$n \leq l$ のときは任意の分割が題意を満たすので，$a_{l,n} = 2^{n-1}$ となる．$n \geq l+1$ の場合，先頭のグループの長さが $1,\ldots,l$ である場合に条件を満たす．それぞれ残り $n-1,\ldots,n-l$ 個の記号に対してグループ分けを考えればよい．よって合計で $\sum_{i=n-k}^{n-1} a_{k,i}$ 個のグループ分けが存在する．□

以上より，グループ分けの種類数を定式化することができた．これを用いて，高さ分解 vnode に関する部分関数の種類数に対して上限を与えることができる．

定理 6.2　B_r に関して分解する場合の部分関数の種類数は $a_{2r,n} - a_{r,n} = O(2^n)$ で上限を与えられる．

証明　B_r に関する分解の部分関数の種類数は，高さ $r+1$ 以上の割当に対するグループ分けの種類数に対応し，補題 6.1 より最大グルーピング数 l は $r+1 \leq l \leq 2r$ を満たす．よって最大グルーピング数 $2r$ 以下のグループ分けの種類数から，r 以下のグループ分けの種類数を引いたものが上限になるため，補題 6.2 より $a_{2r,n} - a_{r,n}$ となる．□

以上の議論より，高さ分解 vnode に対する決定ノードの個数を $O(2^n)$ で見積もることができた．

高さ分解 vnode に対する子ノード

以下，B_r に対する決定ノードが持ちうる子ノードの数についての見積もりを与える．子ノードの個数は，部分関数の $(\mathbf{X},\mathbf{Y})$-分割における要素数に一致する．$(\mathbf{X},\mathbf{Y})$-分割の要素数は，主部に相当する命題変数群に対する可能な割当の総数以下となるため，高さ分解 vnode を参照する決定ノードにおいては，要素数は B_r の割当の種類数以下となる．

$|B_r| = n-r+1$ より，変数の割当は全部で 2^{n-r+1} 種類考えられるが，構文木集合を考える上では B_r のうち全ての変数が使われるわけではない．長さ $n \geq 2$ の記号列 $\alpha_1,\ldots,\alpha_n$ についての任意の構文木は $\alpha_1,\ldots,\alpha_d$ と $\alpha_{d+1},\ldots,\alpha_n$ についての部分的な構文木に分割できる．この 2 つの部分記号列に対する構文木集合を考えることにより，B_r の中で使われる変数を制限することができる．ここで $j-i+1$ 文字からなる記号列 $\alpha_i,\ldots,\alpha_j$ に対する可能な構文木の集合を部分記号列に対する構文木集合と呼ぶ．

以下では，ある構文木集合に対して 2 つの部分記号列に対する構文木集合を考え，この 2 つの集合に含まれない変数を使うことはできないことを示す．これによって B_r の中で使われる変数が制限される．次に，1 つの部分記号列に対する構文木集合の中で取りうる割当の種類数を示す．そして長さ n の記号列について，割当の種類数が最大となるような部分記号列への分割を考えることにより子ノードの数について見積もりを与える．

補題 6.3　長さ $n(\geq 2)$ の記号列に対する構文木集合 T について，ある 2 つの部分記号列に対する構文木集合 U_1, U_2 を考えると，b_{ij} が U_1, U_2 に含まれないならば $b_{ij} = 0$ となる．

証明　$b_{1,n} = 1$ と式 (7) より，$b_{1,d} = 1$，$b_{d+1,n} = 1$ となるような d が $1 \leq d < n$ に存在する．それぞれ文字列 $\alpha_1,\ldots,\alpha_d$，$\alpha_{d+1},\ldots,\alpha_n$ を導出する最初の記号と見ることができ，図 7 のように 2 つの部分記号列に対する構文木集合が考えられる．

$\alpha_1,\ldots,\alpha_d$，$\alpha_{d+1},\ldots,\alpha_n$ に対する構文木集合に含まれる変数集合をそれぞれ U_1，U_2 とおく．$b_{ij} \notin U_1, U_2$ のとき $b_{ij} = 0$ となることを示す．

$b_{ij} \in U_1$ となるとき，b_{ij} は d 文字目より前までの導出に対応するので $j \leq d$ であり，$b_{ij} \notin U_1$ のとき $d < j$ となる．また，$b_{ij} \in U_2$ となるとき，b_{ij} は $d+1$ 文字目より後の導出に対応するので $d+1 \leq i$ であり，$b_{ij} \notin U_2$ のとき $i < d+1$ となる．

以上より，$b_{ij} \notin U_1, U_2$ となるとき $d < j$ かつ $i < d+1$ を満たす．次に，$(1) i = 1$ のときと $(2) i \neq 1$ のときに場合分けして $b_{ij} = 0$ を示す．ただし $(i,j) \neq (1,n)$ とする．

$(1) i = 1$ のとき，$i < d+1 \leq j < n$ が成立するので，式 (8) より $\neg b_{i,j} \vee \neg b_{d+1,n} = 1$ であり，$b_{d+1,n} = 1$ なので $b_{i,j} = 0$.

$(2) i \neq 1$ のとき，$1 < i \leq d < j$ が成立するので，式 (8) より $\neg b_{1,d} \vee \neg b_{i,j} = 1$ であり，$b_{1,d} = 1$ なので $b_{i,j} = 0$.

以上より $b_{ij} \notin U_1, U_2$ のとき $b_{ij} = 0$ となる．□

補題 6.4　$1 < r < n$ を満たす r について，B_r の中で使われうる変数は 2 つの部分記号列に対する構文木集合のうちどちらかに含まれる変数に限定される．

証明　補題 6.3 より，b_{ij} が部分記号列に対する構文木集合に含まれない場合に $b_{ij} = 0$ が決定する．よって任意の高さの変数集合 B_r において，使われうる変数は構文木集合に含まれる変数のみに限定される．□

以上より，B_r の中で使われうる変数が限定されること

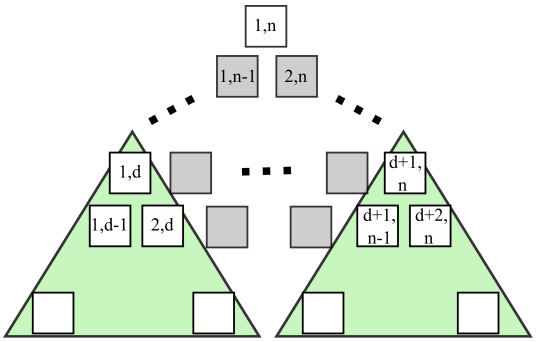


図 7　構文木集合と 2 つの部分記号列に対する構文木集合の例．それぞれの正方形は一つの命題変数を表し，塗りつぶされた正方形は値が 0 に決まるものを表す．部分記号列に対する構文木集合のどちらにも含まれない変数の値は 0 となる．

がわかった．次に，それぞれの部分記号列に対する構文木集合の中で何通りの割当が存在するかを考える．

補題 6.5　$n \geq 4$ の場合のみ，長さ n の記号列に対する構文木集合において B_r の割当の種類数が 3 となる r が存在する．

証明　$n = 2$ のとき，$B_1 = \{b_{11}, b_{22}\}$ は終端記号に対応するので，全て 1 となる 1 通りの割当しか存在しない．

$n = 3$ のとき，$B_2 = \{b_{12}, b_{23}\}$ である．両方とも 0 とした場合，終端記号である $B_1 = \{b_{11}, b_{22}, b_{33}\}$ を導出することができない．よってどちらか一方が 1 となる 2 通りの割当しか存在しない．

$n \geq 4$ のとき，$B_{n-1} = \{b_{1,n-1}, b_{2,n}\}$ に対して両方とも 1 となる場合を除いた 3 通りの割当が存在する．□

分割を繰り返すことによって任意の個数の部分記号列に分けることができる．そのため，長さ n の記号列を分割したとき，3 通りの割当が存在するような部分記号列が何個できるかによって割当の種類数を見積もることができる．

定理 6.3　長さ n の記号列に対する構文木集合について，B_r での割当の種類数は最大で $O(3^{n/4})$ である．

証明　終端記号を長さ 4 のグループに区切り，長さ 4 の部分記号列を考える．この場合が補題 6.5 を満たす部分記号列の数が最大となり，B_3 においてそれぞれの部分記号列一つあたり 3 通りの割当が存在する．$n/4$ 個の部分記号列ができるので，割当の種類数は $O(3^{n/4})$ となる．□

定理 6.4　高さ分解 vnode に対応する ZSDD の大きさは $O(n2^n3^{n/4})$ である．

証明　高さ分解 vnode における決定ノードの個数は $O(2^n)$ で，それぞれの子ノードの上限は $O(3^{n/4})$ である．よって B_r に関する大きさは $O(2^n3^{n/4})$，r は $1 \leq r \leq n$ を満たす数なので $O(n2^n3^{n/4})$ となる．□

その他の vnode

right-linear-vtree となる部分 vtree に含まれる vnode について考える．right-linear-vtree において，それぞれの決定ノードは 1 つの変数が使われる場合に対応している．そのため変数の割当と変数の個数の上限を見積もることにより，大きさに対する上限を与えることができる．

定理 6.5　B_r についての決定ノードの個数は $O(n2^n)$ より小さい．

証明　$|B_r| = n - r + 1$ であるため，変数の割当の種類数は 2^{n-r+1} より小さく，$2^{n-r+1} < 2^n$ である．B_r には n 個以下の変数が含まれ，1 つの変数につき 1 つの決定ノードに対応するので，決定ノードの個数は $O(n2^n)$ より小さい．□

right-linear-vtree の場合は決定ノードが二種類の子ノードを持つので，各 B_r における大きさは $O(n2^n)$ より小さい．高さ分解 vnode に関する大きさは $O(2^n3^{n/4})$ なので，これはオーダーに影響を与えない．

以上より，長さ n の構文木集合を表す ZSDD の大きさを $O(n2^n3^{n/4})$ と見積もることができた．

7. おわりに

本稿では単純な文脈自由文法の構文木集合を決定グラフで表した際の大きさを比較し，実験で最も小さくなった ZSDD について上限を与えた．今後の課題として以下が挙げられる．本稿では一種類の決定グラフにつき一種類の vtree しか与えていないが，それによって大きさが最小になる保証はないため，異なる種類の vtree に関しても考察を行う必要がある．また，終端記号，非終端記号ともに一種類の単純な文法を考えたが，実際に使われる文法には記号が複数あるため，一般の文法に関して考察を行う．

参考文献

[1] 黒橋禎夫：自然言語処理，放送大学教育振興会 (2015).
[2] 阿久津達也：バイオインフォマティクスの数理とアルゴリズム，共立出版 (2007).
[3] Bryant, R. E. and Bryant, R. E.: Graph-based algorithms for Boolean function manipulation, *Computers, IEEE Transactions*, Vol. 100.8, pp. 677–691 (1986).
[4] Minato, S.: Zero-suppressed BDDs for set manipulation in combinatorial problems, *Proceedings of the 30th international Design Automation Conference. ACM*, pp. 272–277 (1993).
[5] Darwiche, A.: SDD: A new canonical representation of propositional knowledge bases, *IJCAI International Joint Conference on Artificial Intelligence*, pp. 819–826 (online), DOI: 10.5591/978-1-57735-516-8/IJCAI11-143 (2011).
[6] Nishino, M., Yasuda, N., Minato, S. and Nagata, M.: Zero-Suppressed Sentential Decision Diagrams, *Proceedings of the 30th Conference on Artificial Intelligence (AAAI 2016)*, pp. 1058–1066 (2016).