汎化された係り受け文脈自由文法の構文解析法

トウシンバット, D.
九州大学大学院システム情報科学研究科知能システム学専攻 : 博士後期課程

冨浦, 洋一
九州大学大学院システム情報科学研究科知能システム学専攻

日高, 達
九州大学大学院システム情報科学研究科知能システム学専攻

https://doi.org/10.15017/1515687

出版情報：九州大学大学院システム情報科学紀要. 5 (2), pp.223-227, 2000-09-26. 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5巻 第2号 平成12年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.5, No.2, September 2000

汎化された係り受け文脈自由文法の構文解析法

D. トウシンバット* ・ 冨浦 洋一** ・ 日高 達**

An Efficient Parsing Algorithm for CFG with Robust Dependency Constraints

Toushinbatto D., Yoichi TOMIURA and Toru HITAKA

(Received June 16, 2000)

Abstract: A context-free grammar (CFG) in which dependency constraints are tucked into production rules has been proposed, where syntactic categories are subdivided by the headword of the phrase. It's difficult to collect all necessary rules expressing dependency constraints in the grammar, so taking superordinate-subordinate relations in the word hierarchy into the grammar as production rules has been considered. However, because there are huge number of syntactic trees for an input sentence in this grammar, how to select a preferable syntactic tree is very important. This paper shows an efficient parsing algorithm, specialized to the heuristics for selecting the most plausible syntactic tree.

Keywords: Dependency constraints, Superordinate-subordinate relations, Parse table , Parsing algorithm

1. はじめに

自然言語処理の構文解析では，一つの入力文に対して，得られる構文木が一般に複数存在するという構文構造の曖昧さの問題がある．構文木の中には意味的に不適格なものも含まれるため，どのようにして適格な構文構造に絞り込むかが，構文解析における重要な問題である．

構文構造の曖昧さの絞り込み手法として，係り受け制約を利用することが考えられる．統語範疇をそれから導出される句の主辞あるいは主辞の意味カテゴリーで細分化して，係り受け制約を確率文脈自由文法の生成規則として表現する手法が提案されている[2),3)]．しかし，詳細に係り受け制約を記述すると，係り受け制約を表現する生成規則数が膨大となり，その一部が文法から洩れてしまうという問題が生じる．係り受け制約を表現する生成規則が洩れた文法による構文解析では，本来得られるべき係り受け制約を満たすような構文木が，得られない場合がある．この解決法の一つとして，概念間の上位－下位関係を下位概念から上位概念への書き換え規則として文法に取り込むことが考えられる[4)]．

このような文法では，入力文に対する構文木の数が膨大になり，これらから尤もらしい構文木を選択する必要がある．また，構文木の数が膨大であるため，すべての部分構文木の部分解析情報 (item) を表 (parse table) に書き込む方式の構文解析アルゴリズムであるアーリー法やチャート法[5)] では，当然 item の数が膨大となり効率が悪い．

本論文では，どのような構文木が尤もらしいかの傾向を考え，構文木の絞り込みのための，ヒューリスティックを設定し，これに特化した効率的な構文解析アルゴリズムを提案する．

2. 係り受け文脈自由文法

2.1 係り受け規則

我々は，係り受け制約を文脈自由文法の生成規則として表現する手法に関する研究を行なっている[2),3)]．本節では，文献 3) に従って，その概要を説明する．

「昨日買ったリンゴ」における「リンゴ」のように，句の中心的な意味を担う単語をその句の主辞 (head) と呼ぶ．句Aが句Bを修飾しているとき，句Aの主辞と句Bの主辞の間に係り受け関係が成立している (句Aの主辞が句Bの主辞に係る)．一般に句Aの中の語が係り受け関係の種類を規定する情報を持ち，これを句 A の function と呼ぶ[†1]．たとえば,「昨日買ったリンゴを食べる」において，後置詞句「昨日買ったリンゴを」の head は「リンゴ」，function は「を」であり，連体修飾句「昨日買った」の head は「買う」，function は末尾の活用語「た」の活用形「連体」である．

単語 α が単語 β にfunciton f で規定される関係で係るとき，これが意味的に適格であるためには，α, β, f の間にある一定の制約がある．これを係り受け制約と呼ぶ．文脈自由文法の生成規則で係り受け制約を表現することを考えよう．生成規則

平成12年6月16日受付

* 知能システム学専攻博士後期課程
** 知能システム学部門

†1 英語の主格や目的格のように，句Aと句Bの位置関係が係り受け関係の種類を規定する場合もある．

$$X \rightarrow Y_1 \cdots Y_i \; X \; Y_{i+1} \cdots Y_l \tag{1}$$

において，Y_j から導出される句が右辺の Xから導出される句を修飾するとする．文脈自由文法において，規則 (1) を用いた導出の後，Y_j からの導出と右辺の X からの導出は独立に行なわれるため，Y_j から導出される句の head および function と右辺の X から導出される句の head の間の係り受け関係が意味的に適格であることを規定できない．そこで，統語範疇を head，function で細分化して，生成規則 (1) を

$$X(h) \rightarrow Y_1(-h) \cdots X(h) \cdots Y_l(-h) \tag{2}$$

と

$$Y_j(-h) \rightarrow Y_j(h_j, f_j) \; (j = 1, 2, \cdots, l) \tag{3}$$

の形式の生成規則で表現し直す．$X(h)$ は head が h である統語範疇 X の句を導出する非終端記号であり，$Y(-h)$ はheadが h である句に係り得る統語範疇 Y の句を導出する非終端記号であり，$Y(h, f)$ は head が h，function が f である統語範疇 Y の句を導出する非終端記号である．規則 (3) は，h_j が f_j で規定される関係で h に係る係り受け関係の適格性，つまり，係り受け制約を表している．本論文では (3) の形式の生成規則を係り受け規則と呼ぶ．

たとえば，「食べる」を head とする動詞句 (VP) に後置詞句 (PP) が係ることを表現する生成規則，および，「リンゴ」を head，「を」を function とする後置詞句 (PP) が「食べる」に係り得る後置詞句であることを表現する生成規則は，それぞれ以下のようになる．

$$VP(\text{食べる}) \rightarrow PP(-\text{食べる}) \; VP(\text{食べる})$$
$$PP(-\text{食べる}) \rightarrow PP(\text{リンゴ}, \text{を})$$

このように，従来自然言語文法で用いられてきた統語範疇をそれから導出される句の head と function で細分化して係り受け制約を表現した文法を，係り受け文脈自由文法と呼ぶ．

2.2 係り受け文脈自由文法の汎化

文献 3) では，係り受け制約の記述の効率を考慮し，概念間の上位－下位関係を上位概念から下位概念への書き換え規則として文法中に取り込み，係り受け規則中の headとして概念も許すようにしている．たとえば

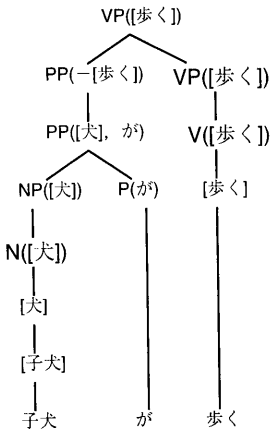

Fig.1　Syntactic tree for "a puppy dog walks"

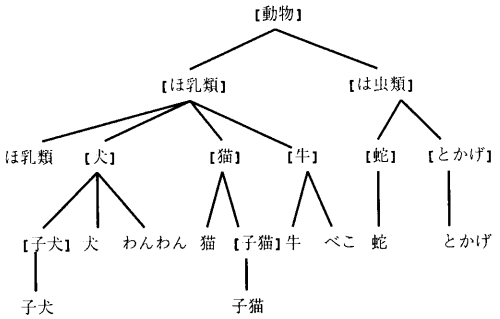

Fig.2　Thesaurus

$$P(-[\text{歩く}]) \rightarrow PP([\text{子犬}], \text{が})$$

が文法になく，

$$P(-[\text{歩く}]) \rightarrow PP([\text{犬}], \text{が})$$

が文法に存在する場合，「子犬が歩く」に対する構文木は **Fig.1**のようになる（**Fig.2**に示すように［子犬］は［犬］の下位概念である）．ここで，$[w]$ は単語w の概念を意味する．しかし，**Fig.2** のいかなる概念 C （$C \in \{$[動物]，[ほ乳類]，[猫] [子猫]$\}$）に対しても

$$PP(-[\text{歩く}]) \rightarrow PP(C, \text{が}) \tag{4}$$

なる規則が文法に存在しなかったとすると，「子猫が歩く」に対する構文木は得られない．

その解決法として，文献 4) では，概念間の上位－下位関係を下位概念から上位概念への書き換え規則として文法に取り込むことを考えた．

たとえば，“［犬］が［歩く］に「が」で係り得る”ならば，［犬］を意味的に近い［子猫］に置き換えた“［子

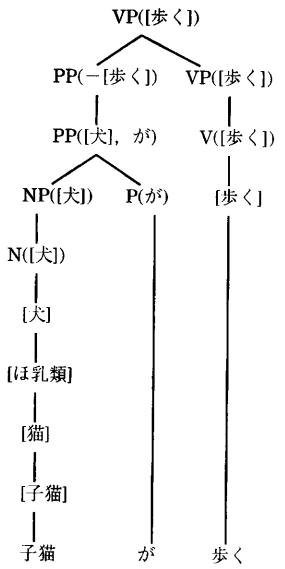

Fig.3　Syntactic tree for "a kitten cat walks"

猫］が［歩く］に「が」で係り得る"が成立する可能性も高いと期待できる．そこで，［犬］と［ほ乳類］の間の上位－下位関係を生成規則

$$[犬] \rightarrow [ほ乳類]$$

として追加し，「子猫が歩く」の解析に

$$PP(-[歩く]) \rightarrow PP([犬], が)$$

を利用できるようにする．このようにして得られる「子猫が歩く」に対する構文木を**Fig.3**に示す．このように，概念間の上位－下位関係を下位概念から上位概念への書き換え規則

$$h \rightarrow h_H \tag{5}$$

として文法に取り込むことで，意味的に近い係り受け制約を，入力文の構文解析に利用できる．ただし，h_H は h の直接の上位概念の一つである．

3. 汎化された係り受け文脈自由文法における構文解析

3.1 問題点とその対策

Fig.2に示すシソーラス中の上位－下位関係を生成規則として取り込んで汎化した係り受け文脈自由文法では

$$PP(-[歩く]) \rightarrow PP([子猫], が) \tag{6}$$
$$PP(-[歩く]) \rightarrow PP([犬], が) \tag{7}$$
$$PP(-[歩く]) \rightarrow PP([とかげ], が) \tag{8}$$

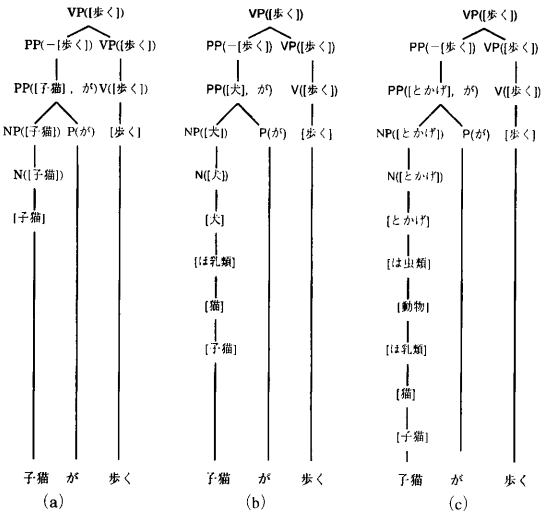

Fig.4　Syntactic trees for "a kitten cat walks"

なる係り受け規則が存在するとすると，「子猫が歩く」に対して，**Fig.4**に示す3つの構文木が得られる．

さらに，head，function で細分化していない通常の統語範疇レベルの文脈自由文法で複数の構文木が得られる文（たとえば「コスモスが咲いた丘に登る」では「コスモス」が「咲いた」に係る構文木と「登る」に係る構文木が得られる）では，汎化した係り受け文脈自由文法では意味的に不適格な係り受け関係を含む多数の構文木も得られ，係り受け制約を表現した本来の目的が失われる．

このように (5) の形式の生成規則を組み込んで汎化した係り受け文脈自由文法では，入力文に対して膨大な構文木が得られるため，尤もらしい構文木を選択する必要がある†2．

また，構文木の数が膨大であるため，すべての部分構文木の部分解析を表に書き込む方式の構文解析アルゴリズムであるアーリー法やチャート法[5]では，効率が悪い．

それでは，どのような構文木が尤もらしいのであろうか．先の「子猫が歩く」の例では **Fig.4 (a)** の構文木が尤もらしい．そこで，これを一般化し，

> 下位概念から上位概念への書き換え規則の適用が少ない構文木を優先する．

という仮定を設ける．(5) の形式の書き換え規則には，それを使用した構文木の尤もらしさが極端に低いものとそうでないものがあるため，(5) の形式の書き換え規則にコストを付与し，上記の仮定をさらに次のように一般化する．

> 適用されている下位概念から上位概念への書き換え規則のコストの和が小さい構文木を優先する．

(5) の形式の書き換え規則以外の規則に関しては通常

†2　文献 4) では，(5) の形式の生成規則の適用確率を正の微小確率とするような確率文法化で，この問題に対処している．

の最尤推定法により適用確率を付与して確率文法化する．

以上をまとめると，汎化された（確率）係り受け文脈自由文法における構文解析では，文法を

$$G =< \Sigma, V, P_1, P_2, S, f_p, f_c >$$

$$\begin{cases} \Sigma: \text{ 終端記号の有限集合} \\ V: \text{ 非終端記号の有限集合} \\ P_1: \text{ 生成規則の有限集合} \\ P_2: \text{ 生成規則の有限集合，ただし，} P_1 \cap P_2 = \emptyset \\ S: \text{ 開始記号 } (S \in V) \\ f_p: P_1 \to (0,1] \\ f_c: P_2 \to N \ (N: \text{自然数の集合}) \end{cases}$$

とし，入力文 w に対し，文脈自由文法 $G' =< \Sigma, V, P_1 \cup P_2, S >$ により得られる構文木集合 $\Im(w)$ に対し

$$\neg \exists\, T' \in \Im(w) \;\; Cost(T') < Cost(T)$$

なる $T \in \Im(w)$ のうち，$Pr(T)$ を最大にする T を優先（選択）する．ただし，

$$Cost(T) = \sum_{\delta \in P_2} n(\delta, T) \times f_c(\delta)$$

$$Pr(T) = \prod_{\delta \in P_1} f_p(\delta)^{n(\delta,T)}$$

$n(\delta, T)$ ：T 中で δ が使用されている回数

である．

3.2 構文解析アルゴリズム

構文解析アルゴリズムの分かりやすさのために生成規則は次の2つの形式だけであるとする †3．

$$X \to a \ (a \in \Sigma)$$
$$X \to \alpha \ (\alpha \in V^+)$$

提案する構文解析アルゴリズムは，以下の PT (parse table) 作成と最左導出抽出の2つの部分からなる．以下の $\star$ は $\star \notin \Sigma \cup V$ なる特殊記号とする．また，$[A \to \alpha \star \beta, i, j, r, c]$ ($[A \to \alpha\beta \in P_1 \cup P_2, i, j, c$は整数$, 0 < r \leq 1)$ を item と呼ぶ．

PT 作成アルゴリズム

入力： $G =< \Sigma, V, P_1, P_2, S, f_p, f_c >$，文字列 $w = a_1 a_2 \cdots a_n$

出力： PT

(I) $C = 0$

(II) $j = 1, 2, \cdots, n$ として以下を繰り返す．

1) **(a)** $A \to a_j \in P_1$ ならば，$[A \to a_j \star, j-1, j, f_p(A \to a_j), 0]$ を PT に登録する．

(b) $A \to a_j \in P_2$ かつ $f_c(A \to a_j) = C$ ならば，$[A \to a_j \star, j-1, j, 1, f_c(A \to a_j)]$ を PT に登録する †4．

2) 以下の操作を新たに追加される item がなくなるまで繰り返す．

(a) $[B \to \gamma \star, k, j, r_1, c_1] \in PT$，$[A \to \alpha \star B\beta, i, k, r_2, c_2] \in PT$，かつ $c_1 + c_2 = C$ ならば，$[A \to \alpha B \star \beta, i, j, r_1 \times r_2, c_1 + c_2]$ を PT に登録する．

(b) b1) $[B \to \gamma \star, i, j, r, c] \in PT$，$A \to B\beta \in P_1$ ならば，$[A \to B \star \beta, i, j, r \times f_p(A \to B\beta), c]$ を PT に登録する．

b2) $[B \to \gamma \star, i, j, r, c] \in PT$，$A \to B\beta \in P_2$ かつ $c + f_c(A \to B\beta) = C$ ならば，$[A \to B \star \beta, i, j, r, c + f_c(A \to B\beta)]$ を PT に登録する．

(III) $[S \to \alpha \star, 0, n, r, c] \in PT$ ならば，終了．そうでないならば，$C \leftarrow C + 1$ として，$C \leq K$ †5 ならば，再び **(II)** から繰り返す．そうでないならば，終了．

ただし，PT 作成段階での **(II)** の **2)** で $[A \to \alpha \star \beta, i, j, r, c]$ の形式の item を PT に登録するとき，$[A \to \alpha \star \beta, i, j, r', c']$ がすでに PT に存在するならば，$c' = c$ かつ $r' < r$ のときのみ，$[A \to \alpha \star \beta, i, j, r', c']$ を削徐し，$[A \to \alpha \star \beta, i, j, r, c]$ を加えるものとする．

上記の PT 作成アルゴリズムが終了した時点で

$$[A \to \alpha \star \beta, i, j, r, c] \in PT$$

は，

†3　この形式以外の生成規則を持つような文法に対しても，後述のアルゴリズムをわずかに修正するだけで済むため，本質的な制限ではい

†4　P_2 の要素がすべて (5) の形式ならば，**(II)** の **1)** の **(b)** は不要．ここでのアルゴリズムはより一般的な文法に対して記述している．

†5　たとえば，P_2 の生成規則をシソーラスの下位概念から上位概念への書き換え規則のみからなるとすると，シソーラスの root から leaf までの最大深さを L，P_2 に含まれる生成規則のコストの最大値を C_{max} として，$K \leq L \times C_{max} \times n$ とすると，統語範疇レベルの文法で非文となる入力に対しては，本構文解析アルゴリズムでも非文となる．

$$\begin{cases} A \to \alpha\beta \in P_1 \cup P_2 \\ \alpha \overset{*}{\Rightarrow} a_{i+1} \cdots a_j \\ A \to \alpha\beta \in P_1 \text{のとき} \\ \quad r = f_p(A \to \alpha\beta) \times \max\limits_{\pi \in D(\alpha,i,j)} \{P_r(\pi)\} \\ \quad c = \min\limits_{\pi \in D(\alpha,i,j)} \{Cost(\pi)\} \\ A \to \alpha\beta \in P_2 \text{のとき} \\ \quad r = \max\limits_{\pi \in D(\alpha,i,j)} \{Pr(\pi)\} \\ \quad c = f_c(A \to \alpha\beta) + \min\limits_{\pi \in D(\alpha,i,j)} \{Cost(\pi)\} \end{cases}$$

と等価であることが示せる．ただし，

$$D(\alpha,i,j) \overset{def}{=} \{\pi | \alpha \overset{\pi}{\Rightarrow} a_{i+1}a_{i+2}\cdots a_j\}$$

$$Cost(\pi) = \sum_{\delta \in P_2} n(\delta,\pi) \times f_c(\delta)$$

$$Pr(\pi) = \prod_{\delta \in P_1} f_p(\delta)^{n(\delta,\pi)}$$

$n(\delta,\pi)$ ： 導出 π 中で δ が使用されている回数

である．

PT 作成アルゴリズムでは，$[S \to \alpha\star, 0, n, r, c] \in PT$ の場合，c より大きなコストを持つ構文木の抽出に必要な item $[A \to \star\beta, i, j, r', c']$ $(c' > c)$ は登録しない．

最左導出抽出アルゴリズム

入力： $G =< \Sigma, V, P_1, P_2, S, f_p, f_c >$，文字列 $w = a_1a_2\cdots a_n$ に対する PT．

出力： 文字列w に対する 最左導出（構文木），あるいは error．

$[S \to \alpha\star, 0, n, r, c] \notin PT$ ならば，error を出力して終了．$[S \to \alpha\star, 0, n, r, c] \in PT$ ならば，$\neg \exists\, [S \to \alpha\star, 0, n, r', c], r' > r$ なる item $[S \to \alpha\star, 0, n, r, c] \in PT$ に対し，関数 $F([S \to \alpha\star, 0, n, r, c])$ を適用する．

$F([A \to \alpha\star, i, j, r, c])$ の定義

(I) $\alpha = a$ $(a \in \Sigma)$ の場合，$return(num(A \to \alpha))$

(II) $\alpha = X_1X_2\cdots X_m$ $(X_i \in V)$ の場合，

1) $k \leftarrow m, v \leftarrow j,\ r' \leftarrow r,\ c' \leftarrow c,\ \pi \leftarrow \varepsilon$.

2) $k > 0$ の間，以下を繰り返す．

(a) $k = 1$ ならば，$[X_k \to \beta\star, i, v, r', c'] \in PT$ を満たす β を求めて，

$\pi \leftarrow F([X_k \to \beta\star, i, v, r', c'])\oplus\pi$,

$k \leftarrow k - 1$.

(b) $k > 1$ ならば，

$$\begin{cases} [X_k \to \beta\star, u, v, r_1, c_1] \in PT \\ [A \to X_1\cdots X_{k-1} \star X_k \cdots X_m, i \\ \quad , u, r_2, c_2] \in PT \\ c' = c_1 + c_2 \\ r' = r_1 \times r_2 \end{cases}$$

を満たす $\beta, u, r_1, r_2, c_1, c_2$ を求め，

$\pi \leftarrow F([X_k \to \beta\star, u, v, r_1, c_1])\oplus\pi$,

$k \leftarrow k - 1, v \leftarrow u, r' \leftarrow r_2, c' \leftarrow c_2$.

3) $return(num(A \to \alpha))$

ただし，

$\oplus$：生成規則番号の連接

$num(\delta)$:生成規則 δ の番号

とする．

4. おわりに

本論文は，シソーラス上の上位－下位関係を下位概念から上位概念への書き換規則として取り込んだ汎化された係り受け文脈自由文法に対する構文解析アルゴリズムを提案した．P_2 の規則（なるべく適用することを避けたい規則）が少ない（あるいはコストが小さい）構文木から順次作成していくことにより，余分な item を PT になるべく登録せずに解析を終了でき，効率的である．

参考文献

1) 日高 達: 確率文法, 情報処理学会学会誌, Vol. 36, No. 2, pp. 169–176 (1995)

2) 田辺利文, 冨浦洋一, 日高 達: 係り受け制約を含む文脈自由文法, 情報処理学会研究報告, Vol. 95, No. 69, pp. 95–101 (1995).

3) 田辺利文, 冨浦洋一, 日高 達: 係り受け文脈自由文法とその日本語への適用, 情報処理学会論文誌, Vol.41 No. 1 pp. 36–45 (2000).

4) D. トウシンバット, 冨浦, 日高:係り受け文脈自由文法の強化法, 情報処理学会研究報告, Vol. 98,No.99,pp. 39–44 (1998).

5) Aho and Ullman: The Theory of Parsing, Translation and Compiling, Vol. 1 ,Prentice Hall(1975).