A SYNTAX PARSER BASED ON THE CASE DEPENDENCY GRAMMAR AND ITS EFFICIENCY

Toru Hitaka and Sho Yoshida

Department of Electronics, Kyushu University, Fukuoka, Japan

SUMMARY

Augumented transition network grammars (ATNGs) or augumented context-free grammars are generally used in natural language processing systems. The advantages of ATNGs may be summarized as 1) efficiency of representation, 2) perspicuity, 3) generative power, and the disadvantage of ATNGs is that it is difficult to get an efficient parsing algorithm becuase of the flexibility of their complicated additional functions.

In this paper, the syntax of Japanese sentences , based on case dependency relations are stated first, and then we give an bottom-up and breadth-first parsing algorithm which parses input sentence using time $O(n^3)$ and memory space $O(n^2)$, where n is the length of input sentence. Moreover, it is shown that this parser requires time $O(n^2)$, whenever each B-phrase in input sentence is unambiguous in its grammatical structure. Therefore, the efficiency of this parser is nearly equal to the Earley's parser which is the most efficient parsing method for general context-free grammars.

1. FUNDAMENTALS OF JAPANESE SENTENCE

The Japanese sentence is ordinarily written in kana (phonetic) letters and kanji (ideographic) characters without leaving a space between words. From the viewpoint of machine processing, however, it is necessary to express clearly the units composing the sentence in such a way as to leave a space between every word as in English. We have no standard way of spacing the units though the need for this has been demanded for a long time.

We give some examples in Figure 1. The first sentence in the figure is of ordinary written form.

The second indicates a way of spacing (i.e. putting a space between every word).

The third indicates another way of spacing (i.e. putting a space between every B-phrase).

Nowadays, many other spacing methods have been tried in several institutes in Japan.

In this paper, input sentences are given in colloquial style in which a spacing symbol is placed between two successive B-phrases.

In Japanese sentences, *BUNSETSUs* (B-phrase) are the minimal morphological units of case dependency, and the syntax of Japanese sentences consists of (1) the syntax of B-phrase as a string of words, and (2) the syntax of a sentence as a string of B-phrases.

A B-phrase usually pronounced without pausing consists of two parts — main part [or equally an independent part in the conventional school grammatical term] and an annex part which is post positioned. We denote the connection of two parts in a B-phrase by a dot if necessary. A main part, which is a conceptual word [or equally an independent word] (e.g. noun, verb, adjective or adverb) provides mainly the information of the concept. On the other hand, an annex part, a possibly null string of suffix words (e.g. auxiliary verbs or particles) provides the information concerning the kakariuke relation and/or the supplementary information (e.g. the speaker's attitude towards the contents of the sentence, tense, etc.)

A word w has it's spelling W, part of speech H and inflexion K. We call (W,H,K) the word structure of w.

Suppose that a string b of length n be a B-phrase. Then, there exist an independent word w_0 and suffix words $w_1, w_2, \cdots, w_\ell$, and

$$b=w_0 w_1 \cdots w_\ell$$

$$\mathrm{Cont}(H_k,K_k,H_{k+1}) \quad (0\leq k<\ell) \quad \cdots(1)$$

$$\mathrm{Termi}(H_\ell,K_\ell) \quad \cdots(2)$$

where (W_i,H_i,K_i) is the word structure of w_i $(0\leq i\leq \ell)$, $\mathrm{Cont}(H_k,K_k,H_{k+1})$ means a word whose part of speech and inflexion are H_k, K_k respectively can be followed by a word whose part of speech is H_{k+1} in

B-phrases and Termi(H_ℓ,K_ℓ) means a word whose part of speech and inflexion are H_ℓ, K_ℓ respectively can be a right-most subword of B-phrases.

(1),(2) are called the rules of B-phrase structure, and

$$(W_0,H_0,K_0)(W_1,H_1,K_1)\cdots(W_\ell,H_\ell,K_\ell) \quad \cdots(3)$$

is called B-phrase structure of b. If (3) satisfies the condition (1), $w_0w_1w_2\cdots w_\ell$ is called to be a left partial B-phrase.

The *kakariuke* relation is the dependency relation between two B-phrases in a sentence. A B-phrase has the syntactic functions of governor and dependent. The function of governor is mainly represented by the independent word of B-phrase. The function of dependent is mainly represented by the string of particles which is the right-most substring of B-phrase and by the word in front of it (right-most non-particle word).

Every particle has the syntactic and partially semantic dependent function with its own degree of power. The particle whose power of dependent function is strongest of all particles appearing in the string of particles is called the representative particle. Therefore, the syntactic function of dependent of a B-phrase is mainly represented by the representative particle and by the right-most non-particle word.

Let $(W_0,H_0,K_0),(W_i,H_i,K_i),(W_j,H_j,K_j)$ be the word structures of independent word, right-most non-particle word and representative particle of a B-phrase, respectively. Then, $<W_0,H_0>_g$, $<W_i,H_i,H_j>_d$ are called the information of governor and the information of dependent of the B-phrase respectively, and the pair $(<W_0,H_0>_g,<W_i,H_i,H_j>_d)$ is called dependency information of the B-phrase.

There are many types of dependency relation such as agent, patient, instrument, location, time, etc. Let $\mathcal{C}$ be the set of all types of dependency relation. The set of all possible dependency relations from a B-phrase b_1 to a B-phrase b_2 is founded on the information of dependent of b_1 and the information of governor of b_2. Therefore, there is a function δ which computes the set of all possible dependency relations $\delta(\alpha,\beta)$ between a B-phrase of dependency information α and another B-phrase of dependency information β. The function δ is realized by the dependency dictionary retrieved with the key of two dependency informations.

The order of B-phrase is relatively free in a simple sentence, except for one constraint that the predicative B-phrase governing the whole sentence must be in the sentence's final position. Japanese is a post positional in this sense.

The pattern of the dependency relations in a sentence has some structural property which is called the rules of dependency structure, and the dependency relations in a sentence are called the dependency structure of a sentence. The dependency structure of a sentence is shown in figure 2, where arrows indicate dependency relations of various types. The rules of dependency structure consist of following three conditions.

i Each B-phrase except one at the sentence final is a dependent of exactly one B-phrase appearing after it.

ii A dependency relation between any two B-phrases does not cross with another dependency relations in a sentence.

iii No two dependency relations depending on the same governor are the same.

Let N be the number of B-phrases in a input sentence, and all B-phrases are numbered descendingly from right to left (see figure 2). We shall fix an input sentence, throughout this chapter. Let DI(i) be the set of all dependency informations of i-th B-phrase.

Definition: A dependency file DF of a sentence is a finite set of 5-tuples.

$$(i,j,\alpha_i,\alpha_j,c) \in DF \iff_{def} \begin{cases} N \geq i > j \geq 1, & \alpha_i \in DI(i), \\ \alpha_j \in DI(j) \text{ and } c \in \delta(\alpha_i,\alpha_j). \end{cases}$$

Definition: If a subset of DF satisfies following conditions 1) to 5), it is called a dependency structure from the ℓ-th B-phrase to the m-th B-phrase ($N \geq \ell > m \geq 1$) and denoted by DS(ℓ,m) or DS'(ℓ,m).

1) If $(i,j,\alpha_i,\alpha_j,c) \in DS(\ell,m)$, then $\ell \geq i > j \geq m$.

2) For arbitrary $i(\ell \geq i > m)$, there exists unique j,α_i,α_j,c such that $(i,j,\alpha_i,\alpha_j,c) \in DS(\ell,m)$.
(Uniqueness of Dependent)

3) If $(i,j,\alpha_i,\alpha_j,c) \in DS(\ell,m)$ and $(j,k,\alpha'_j,\alpha_k,c') \in DS(\ell,m)$, then $\alpha_j = \alpha'_j$.
(Uniqueness of B-phrase structure)

4) If $(i,j,\alpha_i,\alpha_j,c) \in DS(\ell,m)$, $(i',j',\alpha_{i'},\alpha_{j'},c') \in DS(\ell,m)$ and $i > i' > j$, then $j' \leq j$.
(Nest Structure of Dependency)

5) If $(i,j,\alpha_i,\alpha_j,c) \in DS(\ell,m)$, $(i',j,\alpha_{i'},\alpha_j,c') \in DS(\ell,m)$ and $i \neq i'$, then $c \neq c'$.
(Inhibition of Duplication of a Case)

The set of all dependency structures from ℓ-th B-phrase to m-th B-phrase is denoted by $\mathcal{F}(\ell,m)$. Any $DS(N,1) \in \mathcal{F}(N,1)$ is called a dependency structure of the input sentence. The dependency information of j-th B-phrase is unique in $DS(\ell,m)$, since 2) and 3) hold. Let $j_{DI}DS(\ell,m)$ and $j_GDS(\ell,m)$ be the dependency information of the j-th B-phrase in $DS(\ell,m)$ and the set of all the dependency relations that the j-th B-phrase governs in $DS(\ell,m)$, respectively.

$$j_GDS(\ell,m) \overset{def}{=\!=} \{c \mid (i,j,\alpha_i,\alpha_j,c) \in DS(\ell,m)\}$$

Definition: If the k-th B-phrase ($\ell \geq k \geq m$) in $DS(\ell,m)$ has the following property, k(the k-th B-phrase) is called a joint of $DS(\ell,m)$:
For any $(i,j,\alpha_i,\alpha_j,c) \in DS(\ell,m)$, $k \geq i$ or $j \geq k$.
Let $j_0(=\ell) > j_1 > j_1 > \cdots > j_p(=m)$ be the descending sequence of all the joints of $DS(\ell,m)$ (see figure 2). Then, the j_k-th B-phrase is called the k-th joint of $DS(\ell,m)$. There is a dependency relation from k-th joint (dependent) to k+1-th joint(governor) in $DS(\ell,m)$. Let $J.DS(\ell,m)$ be a set of all the joints of $DS(\ell,m)$. $DS(\ell,m/i,j)$ a subset of $DS(\ell,m)$, is defined as follows:

$$DS(\ell,m/i,j) \overset{def}{=\!=} \{(p,q,\alpha_p,\alpha_q,c) \mid (p,q,\alpha_p,\alpha_q,c) \in DS(\ell,m),\ i \geq p > q \geq j\}.$$

Lemma 1. For any positive integer ℓ, i, j, m ($N \geq \ell \geq i > j \geq m$), the following propositions hold.
(i) $DS(\ell,m/i,j) \in \mathcal{F}(i,j)$, if j is a joint of $DS(\ell,m)$.
(ii) $DS(\ell,j) \cup DS(j,m) \in \mathcal{F}(\ell,m)$, if and only if $j_{DI}DS(\ell,j) = j_{DI}DS(j,m)$.
(iii) $\{(\ell+1,j,\alpha,\beta,c)\} \cup DS(\ell,m) \in \mathcal{F}(\ell+1, m)$ if and only if $(\ell+1,j,\alpha,\beta,c) \in DF$, $\beta = j_{DI}DS(\ell,m)$, $j \in J.DS(\ell,m)$ and $c \notin j_GDS(\ell,m)$.
(iv) If $(j_k,j_{k+1},\alpha_k,\alpha_{k+1},c) \in DS(j,m)$ $(k=0,1,2,\cdots)$, then j_k is the k-th joint of $DS(j_0,m)$.

Syntax analysis of a Japanese sentence is defined as giving B-phrase structures and dependency structure of the sentence.

2. THE PARSING ALGORITHM AND ITS EFFICIENCY

In this chapter, we shall give a parsing method which will parse an input sentence using time $O(n^3)$ and space $O(n^3)$, where n is the length of input sentence. Moreover, if the dependency information of each B-phrase is unambiguous, the time variation is quadratic.

The essence of the parsing algorithm is the construction of B-phrase parse list BL and dependency parse list DL which are constructed essentially by a "dynamic programming" method. The parsing algorithm consists of four minor algorithms that are the construction of BL, the obtaining of B-phrase structure, the construction of DL and the obtaining of dependency structure.

B-PHRASE PARSE LIST

Let b be a string of n length and b(i) denote the i-th character from the left end of it.

$$b = b(1)(2) \cdots b(n).$$

The B-phrase parse list of b consists of n minor lists BL(1), BL(2), $\cdots$, BL(n).

□ Form of items in BL(j)
(i, WS, DI)

where, $1 \leq i < j \leq n$, WS is a word structure and DI is a dependency information.

□ Semantics of (i, WS, DI) $\in$ BL(j)
(i, WS, DI) $\in$ BL(j), if and only if there exists a sequence of words $w_0, w_1, \cdots, w_\ell$ satisfying following two conditions:
1) $b(1)b(2) \cdots b(i) = w_0w_1 \cdots w_{\ell-1}$, $b(i+1)b(i+2) \cdots b(j) = w_\ell$, and WS is the word structure of w_ℓ.
2) The string of word $w_0w_1 \cdots w_\ell$ is a left parcial B-phrase of dependency information DI.

ALGORITHM FOR THE CONSTRUCTION OF BL

Input. An input string $b=b(1)(2)\cdots b(n)$.
Output. The B-phrase parse list BL(1), BL(2), $\cdots$, BL(n).
Method. Step 1: Find all the independent word which are the left-most subwords of b, using independent word dictionary and for each independent word $w=b(1)b(2) \cdots b(j)$, add $(0,(W,H,K),\alpha)$ to BL(j) where, (W,H,K) is the word structure of w and $\alpha = (<W,H>_g, <W,H,->_d)$. Then, set the control word i to 1 and repeat Step 2 until $i = n$.
Step 2: Obtain all the suffix words which are the left-most subwords of $B(i+1)B(i+2) \cdots b(n)$ and for each suffix word $w = b(i+1)b(i+2) \cdots b(k)$ of word structure (W',H',K'), and for each item $(j, <W,H,K>,\alpha) \in BL(i)$, add $(i,(W',H',K'), (W',H') \circ \alpha)$ to BL(k) if

C(H,K,K'). $(W',H')\circ\alpha$ is a dependency information defined as follows.

i If H' is a auxiliary verb, then

$$(W',H')\circ\alpha \overset{def}{=\!=} (\langle\alpha\rangle_g, \langle W',H',->_d)$$

where, $\langle\alpha\rangle_g$ is the information of governor of α.

ii Let <W",H",H"'> be the information of dependent of α. When H' is a particle,

$$(W',H')\circ\alpha \overset{def}{=\!=} (\langle\alpha\rangle_g, \langle W",H",H'\rangle_d)$$

if the power of dependency function of H' is stronger than that of H"' , and else

$$(W',H')\circ\alpha \overset{def}{=\!=} \alpha.$$

There exists upper limit in the length of words and there exists upper limit in the number of dependency informations of all left partial B-phrase of a(1)a(2) ··· a(i). Therefore, there exists upper limit for the necessary size of memory space of BL(i) and the theorem 1 follows.

Theorem 1.

Algorithm for the construction of BL requires O(n) memory space and O(n) elementary operations.

We shall now describe how to find a B-phrase structure of specified dependency information from BL. The method is given as follows.

ALGORITHM FOR OBTAINING A B-PHRASE STRUCTURE OF AN INPUT STRING

Input. The specified dependency information α and BL.

Output. A B-phrase structure of dependency information α or the error signal "error".

Method. STEP 1: Search any item $(i,(W,H,K),\alpha)$ in BL(n) such as Termi (H,H). If there is no such item, then emit "error" and halt. Otherwise, output the word structure (W,H,K), set the register R to $(i,(W,H,K),\alpha)$ and repeat the step 2 until $i=0$.

STEP 2: Let R be $(i,(W,H,K),\alpha)$. Search any item $(i',(W',H',K'),\alpha')$ in BL(i) such as C(H',K',H) and $(W,H)\circ\alpha'=\alpha$. There exist at least one element which satisfies above conditions. Output the word structure (W',H',K') and $R \leftarrow (i',(W',H',K'),\alpha')$.

It is easy to know theorem 2 holds.

Theorem 2.

A B-phrase structure of specified dependency information is output by the above algorithm, if and only if the input string has at least one B-phrase structure of specified dependency information and it takes constant memory space and O(n) elementary operations to operate the above algorithm.

The set of all the dependency informations DI of input string b is obtained from BL(n), since

$$DI=\{\alpha \mid (i,(W,H,K),\alpha)\in BL(n),\ C(H,K)\}.$$

DEPENDENCY PARSE LIST DL

Let s be a input sentence of N B-phrases. The set of all the dependency informations DI(i) of the i-th B-phrase is obtained by operating the algorithm of construction of BL on the string of the i-th B-phrase.

The dependency parse list DL of s consists of N-1 minor lists DL(2), DL(3), ··· ,DL(N).

☐ Form of items in DL(i).

$$\left.\begin{matrix}(\alpha_i,j,\alpha_j,c,P)\\(\alpha_i,j,\alpha_j,\$,P)\end{matrix}\right\}$$

where, $N\geq i>j\geq 1$, $\alpha_i\in DI(i)$, $\alpha_j\in DI(j)$, $c\in\mathcal{C}$, $P\subseteq\mathcal{C}$ and \$ is a specially introduced symbol.

☐ Semantics of $(\alpha_i,j,\alpha_j,c,P)\in DL(i)$.

$(\alpha_i,j,\alpha_j,c,P)\in DL(i)$, if and only if there is a dependency structure DS(i,1) of s, where

$$(i,j,\alpha_i,\alpha_j,c)\in DS(i,1),$$
$$j_G DS(i,1)=P.$$

☐ Semantics of $(\alpha_i,j,\alpha_j,S,P)\in DL(i)$.

$(\alpha_i,j,\alpha_j,\$,P)\in DL(i)$, if and only if there is a dependency structure DS(i,1) of s, where

$$\alpha_i=i_{DI}DS(i,1),\ \alpha_j=j_{DI}DS(i,1),$$
j is a joint of DS(i,1) except O-th or 1st joint,
$$j_G DS(i,1)=P.$$

ALGORITHM FOR THE CONSTRUCTION OF DL

Input. The sequence of the sets of all dependency informations DI(1), DI(2), ··· ,DI(N).

Output. Dependency list DL(2), DL(3), ··· ,DL(N).

Method. STEP 1 (Construction of DL(2)): For each $\alpha_2\in DI(2)$, $\alpha_1\in DI(1)$ and $c\in\mathcal{C}$ such that $c\in\delta(\alpha_2,\alpha_1)$, add $(\alpha_2,1,\alpha_1,c,\{c\})$ to DL(2), set i to 2 and repeat the STEP 2 and the STEP 3 until $i=N$.

STEP 2 (Registration of items of the form $(\alpha_{i+1},j,\alpha_j,c,P)$): For any $(\alpha_i,j,\alpha_j,c,P)\in DL(i)$ and $\alpha_{i+1}\in DI(i+1)$, compute $\delta(\alpha_{i+1},\alpha_i)$ and add every $(\alpha_{i+1},i,\alpha_i,c',\{c'\})$ to DL(i+1) such that $c'\in\delta(\alpha_{i+1},\alpha_i)$. And, for any $(\alpha_i,j,\alpha_j,\Delta,P)\in DL(i)$ where $\Delta\in\mathcal{C}\cup\{\$\}$ and $\alpha_{i+1}\in DI(i+1)$, compute $\delta(\alpha_{i+1},\alpha_j)$ and add every $(\alpha_{i+1},j,\alpha_j,c',P\cup\{c'\})$ to DL(i+1) such that $c'\in\delta(\alpha_{i+1},\alpha_j)$ and $c'\notin P$. Go to Step 3.

STEP 3 (Registration of items of the form $(\alpha_{i+1},j,\alpha_j,\$,P)$): For any $(\alpha_{i+1},j,\alpha_j,c,P) \in DL(i+1)$ and $(\alpha_j,k,\alpha_k,\Delta,P') \in DL(j)$, add $(\alpha_{i+1},k,\alpha_k,\$,P')$ to DL(i+1). Then, set i to $i+1$ and go to STEP 2.

Theorem 3.

If there exist no ambiguity in the dependency information of B-phrases of input sentence, then the step 3 in the above algorithm can be replaced to the following step 3'.

STEP 3': For each $(\alpha_{K_{i+1}},j,\alpha_j,\Delta,P) \in DL(K_{i+1})$, add $(\alpha_{i+1},j,\alpha_j,\$,P)$ to DL(i+1), where

$$K_{i+1} \overset{def}{=\!=} \max\{k \mid (\alpha_{i+1},k,\alpha_k,c,P) \in DL(i+1)\}.$$

Then, set i to i+1 and go to STEP 2.

The efficiency of each step of above algorithm is as follows.

The memory size of DL(i) is O(N).

The step 1, the step 2 and the step 3 take constant, O(N) and $O(N^2)$ elementary operations, respectively.

The step 3' takes O(N) elementary operations since it takes O(N) elementary operations to compute K_{i+1}. Therefore, the theorem 4 holds.

Theorem 4.

The algorithm for the construction of DL requires $O(N^2)$ memory space and $O(N^3)$ elementary operations. Moreover, if there exist no ambiguity in the dependency information of each B-phrases, the algorithm requires $O(N^2)$ elementary operations by replacing the step 3 with the step 3'.

We shall now describe how to find a dependency structure of input sentence from DL. To begin with, we shall explain items of partial dependency structure list PDSL.

☐ Form of items in PDSL

$(i,j,\alpha_i^\#,\alpha_j^\#,P^\#)$

where, $N \geq i > j \geq 1$, $\alpha_i^\# \in DI(i) \cup \{\#\}$, $\alpha_j^\# \in DI(j) \cup \{\#\}$, $P^\#$ is a subset of $\mathcal{C}$ or $\#$ and # is specially introduced symbol.

☐ Semantics of $(i,j,\alpha_i^\#,\alpha_j^\#,P^\#)$

The item $(i,j,\alpha_i^\#,\alpha_j^\#,P^\#) \in PDSL$ means to be a dependency structure $DS(i,j) \in \mathcal{J}(i,j)$ such that following conditions 1),2) and 3) hold.

1) If $\alpha_i^\#=\alpha_i(\neq\#)$, then $i_{DI}DS(i,j)=\alpha_i$.

2) If $\alpha_j^\#=\alpha_j(\neq\#)$, then $j_{DI}DS(i,j)=\alpha_j$.

3) If $P^\#=P(\neq\#)$, then $j_G DS(i,j)=P$.

Therefore, (N,1,#,#,#) means to be a dependency structure of the input sentence.

ALGORITHM FOR OBTAINING A DEPENDENCY STRUCTURE FROM DL

Input. DL.

Output. A dependency structure of input sentence or the signal "error".

Method. STEP 1: If DL(N) is empty, emit the message "error", else, initialize PDSL to {(N,1,#,#,#)} and repeat step 2 until PDSL becomes empty.

STEP 2: Take an item freely out of PDSL and delete it from PDSL. According to the form of the item, execute 1) or 2) or 3).

1) If the item is (N,1,#,#,#) of the form and $(\alpha_N,j,\alpha_j,c,P) \in DL(N)$, then output $(N,j,\alpha_N,\alpha_j,c)$, add (N-1,j,#,$\alpha_j$,P/{c}) to PDSL if N-1 ≠ j and add (j,1,α_j,#,#) to PDSL if j ≠ 1.

2) If the item is (i,1,α_i,#,#) of the form and $(j,\alpha_i,\alpha_j,c,P) \in DL(i)$, then output $(i,j,\alpha_i,\alpha_j,c)$, add (i-1,j,#,$\alpha_j$,P/{c}) to PDSL if i-1 ≠ j and add (j,1,α_j,#,#) to PDSL if j ≠ 1.

3) If the item is $(i,j,\alpha_i^\#,\alpha_j,P)$ of the form, where $\alpha_i^\#=\alpha_i$ or #, and $(\alpha_i,j,\alpha_j,c,P) \in DL(i)$, then output $(i,j,\alpha_i,\alpha_j,c)$ and add (i-1,j,#,α_j,P/{c}) to PDSL if i-1 ≠ j. When there is not such item in DL(i), search a pair of items $(\alpha_i,k,\alpha_k,c,P') \in DL(i)$ and $(\alpha_k,j,\alpha_j,\Delta,P) \in DL(k)$, then output $(i,k,\alpha_i,\alpha_k,c)$, add (i-1,k,#,$\alpha_k$,P'/{c}) to PDSL if i-1 ≠ k and add (k,j,α_k,α_j,P) to PDSL.

PDSL needs O(N) memory space and STEP 1, STEP 2 take constant, O(N) elementary operations, respectively.

Theorem 5.

Algorithm for obtaining a dependency structure from DL requires O(N) memory space and $O(N^2)$ elementary operations.

PARSING ALGORITHM

Input. A Japanese sentence in colloquial style.

Output. A dependency structure DS(N,1) of the input sentence and a B-phrase structure of the j-th B-phrase, whose dependency information is $j_{DI}DS(N,1)$, for every j(j=1,2, ··· ,N).

Method. STEP 1: Construct N B-phrase parse lists of all B-phrases of the input sentence and get the sets of dependency informations DI(1), DI(2), ··· , DI(N).

STEP 2: Construct dependency parse list DPL from DI(1), DI(2), ··· ,DI(N).

STEP 3: Obtain a dependency structure DS(N,1) of the input sentence from DL.

STEP 4: Obtain a B-phrase structure of the j-th B-phrase, whose dependency information is $j_{DI}DS(N,1)$, for every j (j=1,2, ··· ,N) and stop.

Let n_j be the length of j-th B-phrase (j=1,2, ··· ,N), and N,n denote the number of B-phrases and the length of input sentence, respectively. Then,

$$n_1+n_2+ \cdots +n_N = n$$
$$N \leq n$$

By theorem 1, theorem 2, theorem 4 and theorem 5, next theorem holds.

Theorem 6.

The parsing algorithm requires $O(n^2)$ memory space and $O(n^3)$ elementary operations. Moreover, if the dependency information of each B-phrase is unambiguous, it requires $O(n^2)$ elementary operations.

3. CONCLUSION

Syntax of Japanese sentences is stated and a efficient parsing algorithm is given. A Japanese sentence in colloquial style is parsed by the parsing algorithm, using time $O(n^3)$ and memory space $O(n^2)$, where n is the length of input sentence. Moreover, it is parsed using time $O(n^2)$ whenever dependency information of every B-phrase is unambiguous.

REFERENCES

1. Aho, Ullman : "The Theory of Parsing, Translation, and Compiling", Prentice Hall vol. 1 (1975).
2. Woods : "Transition Network Grammars for Natural Language Analysis", Communication of the ACM, 13 (1970).
3. Pratt : "LINGOL — A Progress Report", Proc. IJCAI 4 (1975).

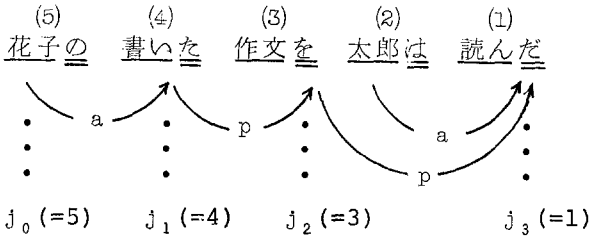
Example: Taro read the composition written by Hanako.

(1) 太郎が花子の書いた作文を読んだ

(2) 太郎 が 花子 の 書い た 作文 を 読ん だ

(3) 太郎が 花子の 書いた 作文を 読んだ

Figure 1. Ways of Spacing



———: main part a: agent

═══: annex part p: patient

j_0, j_1, j_2, j_3 : the sequence of joint

Figure 2. Dependency Structure