Systems and Computers in Japan, Vol. 36, No. 2, 2005
Translated from Denshi Joho Tsushin Gakkai Ronbunshi, Vol. J86-D-II, No. 1, January 2003, pp. 84–97

Incremental Dependency Parsing Based on Headed Context-Free Grammar

Yoshihide Kato,[1] Shigeki Matsubara,[2] Katsuhiko Toyama,[3] and Yasuyoshi Inagaki[4]

[1]Graduate School of International Development, Nagoya University, Nagoya, 464-8601 Japan

[2]Information Technology Center, Nagoya University, Nagoya, 464-8601 Japan

[3]Graduate School of Information Science, Nagoya University, Nagoya, 464-8603 Japan

[4]Faculty of Information Science and Technology, Aichi Prefectural University, Aichi, 480-1198 Japan

SUMMARY

This paper proposes the incremental dependency parsing method based on the context-free grammar with dependency information. In the proposed method, the reachability, which represents the relation between categories, is used. In parallel to the inputting of sentence, the dependency, which is the relation between the modifying word and the modified word, is computed. In the proposed method, the parse tree to cover the inputted fragment of the sentence is not needed. Consequently, the method can compute the dependency more efficiently than the method that computes the dependency from the parse tree. As a result of a parsing experiment using ATIS corpus, the proposed method utilizing the reachability is demonstrated to be effective in reducing the parsing time. © 2005 Wiley Periodicals, Inc. Syst Comp Jpn, 36(2): 63–77, 2005; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/scj.10524

Key words: incremental parsing; dependency; dependency parsing; reachability; incremental interpretation.

1. Introduction

The analysis of natural language is to clarify the syntactic relation and the semantic relation among the constituents of the sentence. Most of the methods proposed to date consider the analysis for the whole sentence. In the analysis of spoken language, however, such an approach is not always adequate. The reason for this is as follows. The speech appears as a time continuation of phonemes, and depending on the spoken language processing system, the analysis must sometimes proceed in parallel to the speech input.

In the real-time interactive speech system, for example, which should generate the word of agreement or interruption during the utterance of the user with an adequate timing [8, 19], the system must understand the speech even in the course of the user's utterance [1, 16, 20]. As another scene, in the simultaneous translation system that outputs the translated sentence in parallel to the utterance of the speaker [13, 17], it is required to convert the syntax to the target language, even in the course of utterance of the original sentence [3, 12].

In those scenes, where the language must be processed simultaneously with the speech input, the incremental parsing procedure for the natural language—that is, "the method to generate successively the result of parsing for the

sentence fragments read up to that moment, in the course of scanning the input sentence following the order of its occurrence"—is required.

One of the representations, for the result generated by the natural language analysis, is the dependency structure. The dependency structure is the analysis focusing on the modifying and modified relations among the words. The importance of such an approach has been widely recognized in recent years, and there are studies of the analysis procedure [4, 7, 9, 21]. Proposed have been a method to resolve the syntactic ambiguity based on the dependency structure [5], and a method to generate the translated sentence based on the dependency structure [2, 14]. There have been few studies, however, on the incremental analysis of the dependency structure.

The dependency parsing procedure can largely be divided, in general, into the following two approaches.

(1) The method where the dependency is generated based on the dependency constraint [7, 21]

(2) The head information is attached to the grammatical rule, and the parse tree is converted to the dependency structure based on that information [4, 5, 9]

In either method, there arises a problem when the incremental dependency parsing is to be realized.

When method (1) is simply applied to the fragments of the sentence in the course of input, the incremental dependency parsing is not realized. The reason for this is that, even if the constraint used in method (1) is adequate as the constraint for the dependency structure of the sentence, it is not always adequate as the constraint for the dependency structure of the sentence fragment. The situation is as follows.

Method (1) utilizes the constraint that "Except for the word representing the semantics of the whole sentence, each word always modifies another word, and the word to be modified is unique" (uniqueness constraint). For some words appearing in the fragment of the sentence in the course of input, it may happen that the word to be modified is not yet inputted. In that case, the dependency structure of the sentence fragment does not always satisfy the uniqueness constraint. It is not clear in such a situation what processing should be applied in the approach of method (1).

In method (2), on the other hand, the parse tree for the whole sentence must be constructed before computing the dependency structure. If the dependency structure is to be computed after generating such a parse tree, the incremental parsing property will be degraded greatly.

From such a viewpoint, this paper proposes the following method. Each time a word is inputted successively from the left, the dependency structure is computed for the sentence fragment, which is inputted up to that stage. The method proposed in this paper is based on the context-free grammar with attached dependency information, and generates all grammatically possible dependency structures for the sentence fragments in the course of input. It is shown first that, by combining the incremental parsing based on the context-free grammar with the dependency computation from the parse tree, such an incremental dependency parsing can be realized.

As the next step, a method is proposed to realize the dependency parsing, which is equivalent to or more efficient than the above method, namely, an incremental parsing method based on the reachability. In the proposed method, it is not necessary to generate the parse tree for the inputted sentence fragment. Consequently, efficient parsing can be realized, and the method is suited to the incremental spoken language processing.

The validity of the proposed method is shown theoretically. A parsing experiment is run using ATIS corpus contained in Penn Treebank [11], and it is shown that the proposed method is effective in reducing the parsing time. Compared to the direct dependency parsing procedure described in Section 2, based on the incremental chart parsing, the parsing time is reduced to approximately 1/1000 on the average.

This paper is structured as follows. In Section 2, the parsing method based on the incremental chart parsing is described. Section 3 proposes the incremental dependency parsing method based on reachability. For Theorem 8 asserting the validity of the method, the detailed proof is given in the Appendix. Section 4 outlines the experiment, and evaluates the proposed method based on the result of experiment.

2. Incremental Dependency Parsing

The dependency is the relation between the modifying word and the modified word. It can be calculated from the parse tree, if the dependency information among phrases is attached to each rule of the context-free grammar (CFG), which defines the phrase structure [5]. In order to compute the dependency among the words incrementally in the order of word occurrence, the following procedure should be executed.

(1) The CFG-based incremental parsing, which is similar to the incremental chart parsing, the partial parse tree for the input sentence fragment is generated.

(2) Following the method in Ref. 5, the dependency among the words is computed from the partial parse tree generated in (1).

In this section, as the first step, the computation for the dependency from the parse tree is described. Then, the incremental chart parsing method, which is one of the

incremental parsing methods, as well as the dependency computation based on the method are described.

2.1. Dependency structure

In discussing the dependency, the modified word is called the head, and the modifying word is called the dependent. In the following, the dependency, in which the head is w_h and the dependent is w_d, is written as $\langle w_d \rightarrow w_h \rangle$. The set of dependencies among the words in the inputted sentence is called the dependency structure for the sentence. Figure 1(b) shows an example of the dependency structure.

2.2. Dependency computation based on parse tree

The dependency structure for the sentence can be computed from the parse tree for the sentence, by defining the category for each rule in CFG, called head child [5]. The only one category on the right-hand side of each grammatical rule is called head child. It implies that the phrase with the category being the head child is modified by other phrases. In the following, the head child is indicated using an asterisk. In the grammar of Fig. 2, for example, the head child in the rule $s \rightarrow np\ vp^*$ is vp.

By defining the head child in each grammatical rule, the head word, which is the word representing the semantics of the whole parse tree, is defined as follows.

[Definition 1] (head word of parse tree) Let σ be the parse tree.

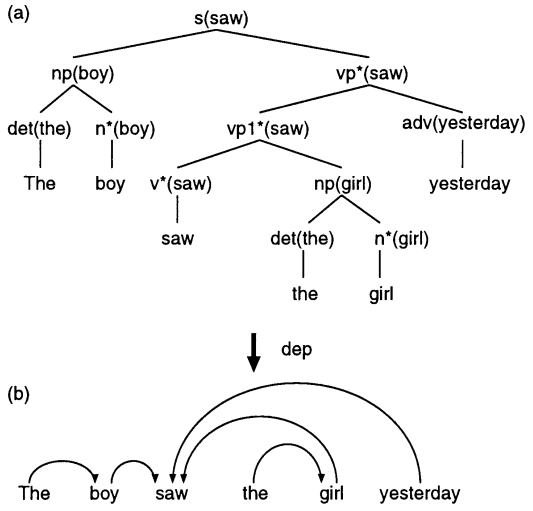


Fig. 1. An example of computing dependencies from parse tree.

s	$\rightarrow np\ vp^*$	det	$\rightarrow$ the
np	$\rightarrow det\ n^*$	n	$\rightarrow$ girl /boy
vp	$\rightarrow vp1^*\ adv\ /\ vp1^*\ pp$	v	$\rightarrow$ saw
$vp1$	$\rightarrow v^*\ np\ /\ v^*\ np\ np$	adv	$\rightarrow$ yesterday

Fig. 2. Grammar and lexicon.

(1) If $\sigma = [w]x$, the head word of σ is w.

(2) If $\sigma = [[\cdots]\,x_1 \cdots [\cdots]\,x_h\ \cdots [\cdots]\,x_m]_A$ and if the head child of the grammatical rule $A \rightarrow X_1 \cdots X_h \cdots X_m$ used in composing σ is X_h, that is, if $A \rightarrow X_1 \cdots X_h^* \cdots X_m$, then the head word of σ is the head word of $[\cdots]x_h$. □

In the following, the head word of parse tree σ is written as $head(\sigma)$. When the head word of parse tree is defined, the dependency structure can be computed from the parse tree.

[Definition 2] (dependency structure of parse tree) Let σ be a parse tree. The function dep is defined as follows.

(1) If $\sigma = [w]x$, $dep(\sigma) = \phi$.

(2) Let $\sigma = [[\cdots]x_1 \cdots [\cdots]x_h \cdots [\cdots]x_m]_A$. Then, the head child of the grammatical rule $A \rightarrow X_1 \cdots X_h \cdots X_m$ is X_h. In other words, if $A \rightarrow X_1 \cdots X_h^* \cdots X_m$, then

$$dep(\sigma) = d(\sigma) \cup \bigcup_{i=1}^{m} dep([\cdots]_{X_i})$$

In the above, $d(\sigma)$ is defined as follows:

$$d(\sigma) = \{\langle w_d \rightarrow head([\cdots]_{X_h})\rangle \mid w_d = head([\cdots]_{X_j})(1 \leq j \leq m, j \neq h)\}$$

□

$dep(\sigma)$ is the dependency structure, which is computed from σ. Figure 1 indicates that, when the computation is executed from parse tree of (a) according to Definition 2, the dependency structure is obtained as in (b). The word in parentheses next to the category is the head word of parse tree with that category as the root.

2.3. Incremental chart parsing and dependency structure computation

The incremental dependency parsing can be realized by applying the dependency structure computation from the parse tree described in the previous section to the partial parse tree generated by the incremental parsing based on CFG. This section describes, as the first step, the incremental chart parsing, which generates the partial parse tree for the sentence fragment incrementally. Then, a method is

shown where the dependency structure is computed based on the incremental chart parsing.

2.3.1. Incremental chart parsing

The incremental chart parsing retains the result of analysis, using the graph called chart, as in the conventional chart parsing [10]. The chart is composed of the set of nodes and the set of edges. The node is placed between words in the inputted sentence. The node between the i-th word w_i and the $(i + 1)$-th word w_{i+1} is labeled as number i. In the following, the node with label i is simply called node i. The edge connects nodes, with the parse tree for the part covered by that edge in the inputted sentence as the label.

The parse tree is represented by the data structure called term, and is represented by the notation $[\alpha]_X$, where X is a category and α is a word, symbol ? or a list of terms. For the term σ, which is $[\alpha]_X$, X is called the category of σ. The term including ?, such as $[?]_X$, is called the undecided term, indicating that the structure is not yet determined. Among the undecided terms appearing in the term, the leftmost term is called the leftmost undecided term. When an undecided term appears in the terms, which are attached as the label to the edge, the edge is called active edge; if not, the edge is called inactive edge.

Incremental chart parsing is the procedure where a new manipulation is introduced into the standard bottom-up chart parsing. In the standard bottom-up chart parsing, the following three operations are executed. In the following, the edge with label σ, connecting nodes i and j of the chart, is written as (i, j, σ).

(Operation 1) dictionary consultation: If the category of i-th word w_i is X, the inactive edge $(i - 1, i, [w_i]_X)$ is added to the chart.

(Operation 2) application of grammatical rule: When the inactive edge $(i, j, [\cdots]_X)$ exists in the chart, if there exists the grammatical rule $A \rightarrow XY \cdots Z$, the edge $(i, j, [[\cdots]_X[?]_Y \cdots [?]_Z]_A)$ is added to the chart.

(Operation 3) replacement of term: Let (i, j, σ) be an active edge in the chart, and the leftmost undecided term of σ be $[?]_X$. Then, if there exists an inactive edge (j, k, σ') in the chart, such that the category of term σ' is X, the edge with the term, which is obtained by replacing the leftmost undecided term of σ by σ', as the label, is added between the nodes i and k of the chart.

In the incremental chart parsing, the following two operations are further added, in order to realize the application of the grammatical rule to the active edge and the replacement of the term by active edge.

(Operation 4) application of grammatical rule to active edge: When there exists active edge (i, j, σ) in the chart, if the category of σ is X and there exists the grammatical rule $A \rightarrow XY \cdots Z$, the edge $(i, j, [\sigma[?]Y \cdots [?]_Z]_A)$ is added to the chart.

(Operation 5) replacement of term by term of active edge: Let (i, j, σ) be an active edge in the chart, and let the leftmost undecided term of σ be $[?]_X$. Then, if there exists an active edge (j, k, σ'), such that the category of term σ' is X, the edge with the term, which is obtained by replacing the leftmost term of σ by σ', as the label, is added between the nodes i and k of the chart.

In other words, the incremental chart parsing introduces the application of the grammatical rule to the active edge, as well as the operation to replace the leftmost undecided term of the term labeled to the active edge by the term labeled to another active edge.

As an example, consider the parsing of sentence fragment "The boy saw." In the standard bottom-up chart parsing, the following terms are generated:

$$[[[the]_{det}[boy]_n]_{np}[?]_{vp}]_s \tag{1}$$

$$[saw]_v \tag{2}$$

$$[[saw]_v[?]_{np}]_{vp1} \tag{3}$$

The term for "The boy saw," however, is not generated.

In the incremental chart parsing, on the other hand, the grammatical rule $vp \rightarrow vp1\ adv$ is applied to term (3), and the term

$$[[[saw]_v[?]_{np}]_{vp1}[?]_{adv}]_{vp} \tag{4}$$

is generated. By replacing the leftmost undecided term of (1) by term (4), the term

$$[[[the]_{det}[boy]_n]_{np}[[[saw]_v[?]_{np}]_{vp1}[?]_{adv}]_{vp}]_s \tag{5}$$

is generated for "The boy saw."

2.3.2. Dependency structure computation

The dependency structure for the sentence fragment can be computed by applying the function *dep*, which is defined in Definition 2 to compute the dependency struc-

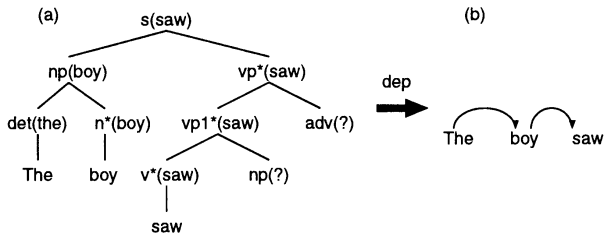


Fig. 3. An example of computing dependencies for initial fragment.

ture, to the term generated by the incremental chart parsing. For the term contained in term (5), for example, the head word is derived as in Fig. 3(a), and the dependency structure for the sentence fragment "The boy saw" is computed as in Fig. 3(b).

3. Incremental Dependency Parsing

By the procedure of the previous section, the dependency structure for the sentence fragment can be computed incrementally. The procedure, however, is not efficient, and there is a possibility that many terms with the same dependency structure are generated. In the incremental chart parsing for "The boy saw," not only the term of Eq. (5), but also the following terms are generated:

$$[[the]_{det}[boy]_n]_{np}[[[saw]_v[?]_{np}]_{vp1}[?]_{pp}]_{vp}]_s \quad (6)$$

$$[[the]_{det}[boy]_n]_{np}[[[saw]_v[?]_{np}[?]_{np}]_{vp1}[?]_{adv}]_{vp}]_s \quad (7)$$

$$[[the]_{det}[boy]_n]_{np}[[[saw]_v[?]_{np}[?]_{np}]_{vp1}[?]_{pp}]_{vp}]_s \quad (8)$$

The same dependency structure is derived from terms (5) to (8). If only the determination of the dependency structure is the concern, it is desirable from the viewpoint of the parsing efficiency that the dependency structure is correctly computed without generating many such terms with the same dependency structure. One of the reasons for generating the terms with the same dependency structure is the application operation of the grammatical rule. In other words, even if there is more than one application of grammatical rules to a term, it can happen that some of the terms obtained by applying the rules have the same dependency structure.

In the incremental chart parsing, the grammatical rules are applied, not only to the inactive edge, but also to the active edge, and the above situation has a larger effect. From such a viewpoint, this section proposes the connection operation for the terms based on the concept of reachability, as a substitute for the application of grammatical rule to the active edge. It is shown that the dependency structure can be computed incrementally by such an approach.

3.1. Reachability

Reachability is the relation between categories. It is defined as follows.

[Definition 3] (reachability) If there exists grammatical rule $A \rightarrow XY \cdots Z$, it is written that $X \rightsquigarrow A$. $\rightsquigarrow^*$ is defined as the reflexive transition closure of $\rightsquigarrow$. If $X \rightsquigarrow^* Y$, X is called reachable to Y. □

When X is reachable to Y, it implies that the parse tree with Y as the root can have X as the left-end descendent.

In order to compute the dependency structure, the head word of parse tree must be determined. In order to determine the head word without actually applying the grammatical rule, the reachability is classified in the proposed method. The reachability of X to Y implies that by applying several grammatical rules to term $[\cdots]_X$, a term with Y as the category is obtained. The classification is based on the relation between the head word of the term obtained in the above way and the head word of $[\ldots]_X$. In the following, the position of the head child of grammatical rule r is written as $hc(r)$.

[Definition 4] Let $\overset{h}{\rightsquigarrow}$ and $\overset{d}{\rightsquigarrow}$ be the relation between categories. They are defined as follows. If there exists the grammatical rule $r = A \rightarrow XY \cdots Z$ such that $hc(r) = 1$, it is written that $X \overset{h}{\rightsquigarrow} A$. If there exists the grammatical rule $r = A \rightarrow XY \cdots Z$ such that $hc(r) \neq 1$, it is written that $X \overset{d}{\rightsquigarrow} A$. □

$X \overset{h}{\rightsquigarrow}{}^* Y$ implies that, when several grammatical rules are applied to term $[\ldots]_X$, a term with Y as the category is obtained, and the head word of $[\ldots]_X$ propagates to that term. $X \rightsquigarrow^* \overset{d}{\rightsquigarrow} \rightsquigarrow^* Y$, on the other hand, implies that, when several grammatical rules are applied to term $[\ldots]_X$, a term with Y as the category is obtained, and the head word of $[\ldots]_X$ does not propagate to that term.

3.2. Incremental dependency parsing based on reachability

By the above preparation, the dependency parsing procedure based on reachability is proposed. The method is composed of the following three procedures.

(Procedure 1) The term is generated by the standard bottom-up chart parsing.

(Procedure 2) The generated terms are connected through reachability.

(Procedure 3) The dependency structure is computed from the connected terms.

In the proposed method, it is theoretically guaranteed that the dependency structure derived by the method is the same as the structure derived by the method described in Section 2. Furthermore, a much more efficient incremental dependency parsing is realized. Those properties are shown in the following.

In Procedure 2, the terms, which are connected through the reachability, are called the connecting terms. They are defined as follows.

[Definition 5] (connecting terms) Let $\sigma_i (i = 1, \ldots, n)$ be the terms which are generated by the standard bottom-up chart parsing. For each i, let the category of σ_i be X_i, and the category of the leftmost undecided term be Y_i. Let $R_i \in \{\&_h, \&_d\}$ (where $\&_h$ and $\&_d$ are the notations indicating that the terms are connected through $\overset{h}{\rightsquigarrow}^*$ and $\rightsquigarrow^* \overset{d}{\rightsquigarrow} \rightsquigarrow^*$, respectively). The sequence $\sigma_1 R_1 \cdots R_{i-1} \sigma_i R_i \cdots R_{n-1} \sigma_n$ satisfying the following (i) and (ii) is called the connecting terms.

(i) If $R_i = \&_h$, $X_{i+1} \overset{h}{\rightsquigarrow}^* Y_i$.
(ii) If $R_i = \&_d$, $X_{i+1} \rightsquigarrow^* \overset{d}{\rightsquigarrow} \rightsquigarrow^* Y_i$. □

In Procedure 2, the terms which are generated by the standard bottom-up chart parsing are connected. The manipulation is as follows.

(Procedure 2) Let (i, j, σ) be an active edge in the chart, and let the leftmost undecided term of the term σ be $[?]_Y$. Let $(j, k, \sigma_1 R_1 \cdots R_{n-1} \sigma_n)$ be the edge with the connecting term in the chart as the label, and let the category of σ_1 be X. Then, if $X \overset{h}{\rightsquigarrow}^* Y$, the edge $(i, k, \sigma \&_h \sigma_1 R_1 \cdots R_{n-1} \sigma_n)$ is added to the chart. If $X \rightsquigarrow^* \overset{d}{\rightsquigarrow} \rightsquigarrow^* Y$ the edge $(i, k, \sigma \&_d \sigma_1 R_1 \cdots R_{n-1} \sigma_n)$ is added to the chart.

Terms (1) and (2), for example, are connected as in Fig. 4.

In Procedure 3, the dependency structure is computed based on the connecting terms. The head word of connecting term is defined first.

[Definition 6] (head word of connecting term) Let τ be a connecting term.

(1) If τ is a term which is generated by the standard bottom-up chart parsing, the head word of τ is $head(\tau)$.

(2) Let $\tau = \sigma_1 R_1 \sigma_2 R_2 \cdots R_{n-1} \sigma_n$ and let $\sigma_1 = [[\cdots]_{X_1} \cdots [\cdots]_{X_{k-1}} [?]_{X_k} \cdots {}_{X_m}]_A$. Then, if $R_1 = \&_h$, and X_k is the head child, the head word of τ is the head word of $\sigma_2 R_2 \cdots R_{n-1} \sigma_n$. If otherwise, the head word of τ is $head(\sigma_1)$. □

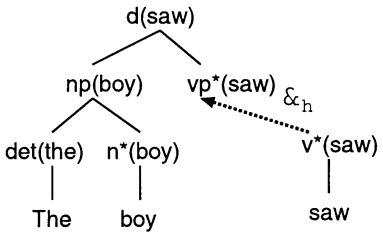


Fig. 4. Connecting terms.

In the following, the head word of the connecting term τ is written as $head_C(\tau)$. In Procedure 3, the dependency structure is computed based on the connecting terms, by the following definition.

[Definition 7] (dependency structure of connecting term) Let τ be the connecting term. The function dep_C is defined as follows.

(1) If τ is the term which is generated by the standard bottom-up chart parsing, $dep_C(\tau) = dep(\tau)$.

(2) Let $\tau = \sigma_1 R_1 \sigma_2 R_2 \cdots R_{n-1} \sigma_n$, and let $\sigma_1 = [[\cdots]_{X_1} [\cdots]_{X_{k-1}} [?]_{X_k} \cdots [?]_{X_m}]_A$. Then,

$$dep_C(\tau) = d_C(\tau) \cup dep_C(\sigma_2 R_2 \cdots R_{n-1} \sigma_n) \cup \bigcup_{i=1}^{k-1} dep([\cdots]_{X_i})$$

where $d_C(\tau)$ is defined as follows. Let $h = hc(A \rightarrow X_1 \cdots X_m)$.

(a1) If $R_1 = \&_h$ and $h = k$,

$$d_C(\tau) = \{\langle w_d \rightarrow head_C(\sigma_2 R_2 \cdots R_{n-1} \sigma_n)\rangle \mid w_d = head([\cdots]_{X_j})(1 \le j \le m, j \ne h)\}$$

(a2) If $R_1 = \&_h$ and $h \ne k$,

$$d_C(\tau) = \{\langle w_d \rightarrow head([\cdots]_{X_h})\rangle \mid w_d = head([\cdots]_{X_j})(1 \le j \le m, j \ne h, j \ne k) \text{ or } w_d = head_C(\sigma_2 R_2 \cdots R_{n-1} \sigma_n)\}$$

(b) If $R_1 = \&_d$, $d_C(\tau) = d(\sigma_1)$. □

In the proposed method, if terms are connected through $\&_h$, the head word of the term in connected terms is propagated [Fig. 5(a)]. If terms are connected through $\&_d$, the head word is not propagated [Fig. 5(b)]. By thus utilizing the reachability, the dependency structure of the sentence fragment can be computed incrementally, without

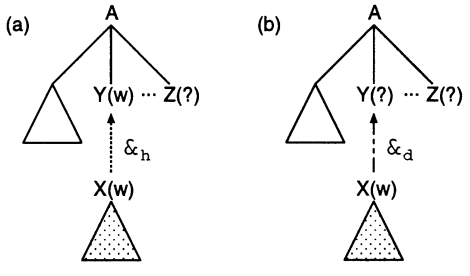


Fig. 5. Head word propagation based on reachability relation.

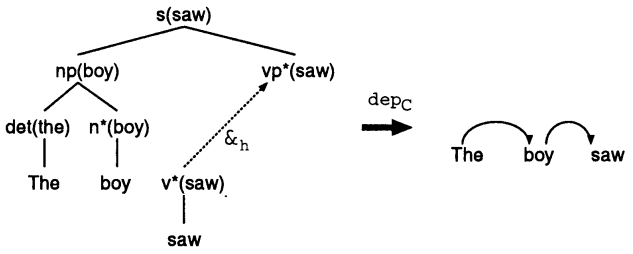


Fig. 6. An example of head word propagation.

applying the grammatical rule to the active edge. Even though there are many ways to apply the grammatical rule, the number of classified cases concerning the reachability is at most 2. Consequently, the dependency structure can be computed more efficiently. The dependency structure computed by the proposed method is equivalent to the structure computed by the method in Section 2. In other words, the following theorem applies.

[Theorem 8] Let **w** be the word sequence, let $I(\mathbf{w})$ be the set of terms for **w** generated by the incremental chart parsing, and let $C(\mathbf{w})$ be the set of connecting terms for **w**. Then, $dep(I(\mathbf{w})) = dep_C(C(\mathbf{w}))$. □

The proof is shown in the Appendix.

3.3. Example of parsing

As an example, consider the parsing for the sentence fragment "The boy saw." For this fragment, the proposed method applies the standard bottom-up chart parsing, and generates terms (1) and (2). The category of (2) is *v*, and the category of the leftmost undecided term of (1) is *vp*. Since $v \overset{h}{\rightsquigarrow}{}^{*} vp$, (1) and (2) are connected, and the following connecting term is generated:

$$[[[the]_{det}[boy]_n]_{np}[?]_{vp}]_s \&_h [saw]_v \qquad (9)$$

The head word of this connecting term (9) is computed as in Fig. 6. Using $v \overset{h}{\rightsquigarrow}{}^{*} vp$, "saw" as the head word of (2) is propagated to the leftmost undecided term of (1). By applying the function dep_C to determine the dependency structure of connecting term (9), the dependency structure of "The boy saw" is determined as $\{\langle the \rightarrow boy\rangle, \langle boy \rightarrow saw\rangle\}$. It is the same as the dependency structure, which is computed by the method of Section 2 from terms (5) to (8).

4. Experiment and Discussion of Results

In order to evaluate the proposed method, a dependency parsing experiment is run as follows. The proposed method, the method in Section 2, and the straightforward method based on the standard bottom-up chart parsing (described later) are implemented on Linux PC (Pentium IV 2 GHz, with main memory 2 GB) using GNU Common Lisp. As the object of the experiment, all 578 sentences in ATIS corpus with parse tree, recorded in Penn Treebank [11], are used. There are 509 grammatical rules used in the experiment, which are extracted from the parse tree of the corpus. The position of the head child in the grammatical rule is defined, following the method in Ref. 6.

4.1. Experiment for parsing time

In order to evaluate the parsing efficiency of the proposed method, the parsing time is compared between the proposed method and the method in Section 2. Figure 7 shows the average parsing time per sentence as a function of the sentence length.[†] The average parsing time per sentence is 0.017 s in the proposed method. On the other hand, it is 11.081 s in the method of Section 2. Thus, the proposed method based on the reachability effectively reduces the parsing time in the incremental dependency parsing.

The number of edges to cover the inputted sentence fragment is less in the proposed method than in the method of Section 2. This helps to reduce the parsing time. Figure 8 shows, for example, the number of edges, which are generated for the English sentence

I need to have dinner served. (10)

The number of edges with the generated connecting terms as the labels is obviously less than the number of edges generated by the incremental chart parsing.

Examining the case where the 5th word "dinner" is inputted, approximately 6×10^4 edges are generated in the incremental chart parsing, but only about 100 edges are generated in the proposed method. Figure 9 shows the parsing time for the same sentence. It is seen that the parsing time is greatly reduced by the time "dinner" is inputted.

4.2. Accuracy of parsing under time limitation

When the incremental dependency parsing procedure is introduced into the spoken language processing system, such as real-time speech conversation and simultaneous translation, it is necessary that the parsing time for the user's utterance be less than the time of the user's utterance. If the parsing time is longer than the time of utterance, simultaneous progress with the sentence input cannot be continued

[†]When the parsing time exceeds 60 s per word, the parsing is terminated. The result shown here is for 154 sentences, for which parsing is not terminated in the method of Section 2. In the proposed method, parsing is not terminated for all of the 514 sentences, including the above 154 sentences.

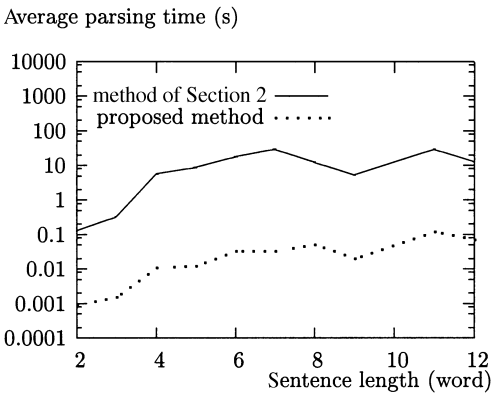


Fig. 7. Parsing time per sentence.

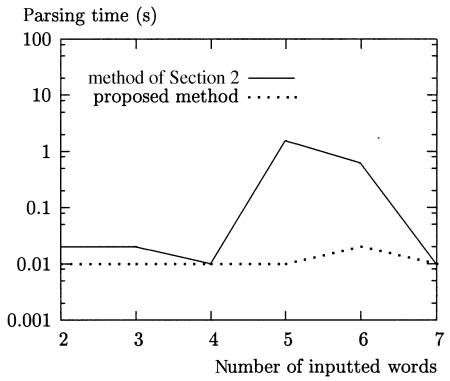


Fig. 9. The parsing time per word for the sentence, "I need to have dinner served."

eventually, even if the incremental dependency parsing is applied.

This section analyzes the accuracy of parsing with such a time restriction. As the first step, the correct dependency structure for the fragment x of sentence s is defined as follows.

(1) Parse tree for sentence s given by corpus is converted to the dependency structure based on the head information of the grammar (this is called the correct dependency structure of s).

(2) Among the dependency structures that compose the correct dependency structure of s, those composed only of words appearing in x are extracted (the dependency structure composed of those dependency structures is called the correct dependency structure of x).

For English sentence "The boy saw the girl yesterday.", for example, the correct dependency structure is given as $\{\langle the \rightarrow boy\rangle, \langle boy \rightarrow saw\rangle, \langle the \rightarrow girl\rangle, \langle girl \rightarrow saw\rangle, \langle yesterday \rightarrow saw\rangle\}$. For the fragment "The boy saw the," the correct dependency structure is $\{\langle the \rightarrow boy\rangle, \langle boy \rightarrow saw\rangle\}$.

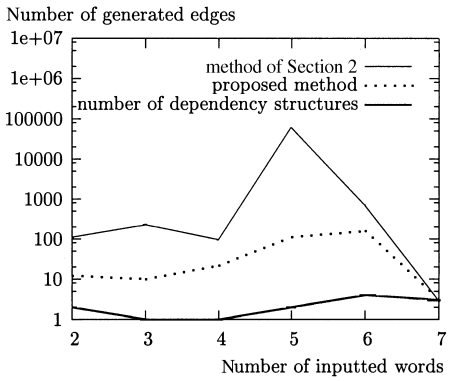


Fig. 8. The number of edges for the sentence, "I need to have dinner served."

The proposed method and the method of Section 2 are equivalent in the sense of Theorem 8. Consequently, the proposed method can always compute the correct dependency structure when the grammatical rules, which are needed to compose the parse tree, are given. In fact, for all sentence fragments in the experiment, for which the result of parsing is obtained, the proposed method could compute the correct dependency structure (see Table 1).

The above property is guaranteed, however, only when the parsing time is not limited. When the parsing time is limited, it is not always true that the correct dependency structure is computed. In this case, it is not clear to what extent of accuracy the proposed method can achieve the parsing. From such a viewpoint, a time limit is set in parsing and an experiment is run. The ratio of the sentence fragments for which the correct dependency structure is generated under the time limit, is defined as the accuracy of the incremental parsing.

In the experiment, the utterance speed in natural English speech is considered, and the time limit for parsing of a word is set as 0.3 s.[†] In other words, the words are inputted from the left successively with an interval of 0.3 s. The computation of the dependency structure for the sentence fragment $w_1 \cdots w_i$ is started when word w_i is inputted (when the parsing for $w_1 \cdots w_{i-1}$ is not completed at that time, the computation is started when that parsing is completed). The precision of parsing is calculated as the "ratio of sentence fragments for which the correct dependency structure is computed within the time limit." The average

[†]Examining the English speaker utterance in CIAIR simultaneous translation database [15], the average utterance time per word is about 0.32 s.

length of 578 sentences examined in the experiment is 8.5 words. There are 4931 sentence fragments as the object of incremental dependency parsing.

As a method to generate incrementally the dependency structure, a more straightforward method can be considered, in which the dependency structure is computed from all terms generated by the standard bottom-up chart parsing. When, for example, the sentence fragment "The boy" is analyzed by the standard bottom-up chart parsing using the grammar of Fig. 2, the term $[[the]_{det}[boy]_n]_{np}$ is generated. The dependency structure of this term is $\{\langle the \rightarrow boy\rangle\}$, which is considered as the result of computation for the dependency structure of "The boy."

In this method, however, it can happen that the correct dependency structure is not generated. In fact, when the sentence fragment "The boy saw" is parsed, using the same grammar, terms such as $[[the]_{det}[boy]_n]_{np}$ and $[[saw]_v[?]_{np}]_{vpt}$ are generated. The dependency structure for those terms is $\{\langle the \rightarrow boy\rangle\}$ or ϕ, and the correct dependency structure $\{\langle the \rightarrow boy\rangle, \langle boy \rightarrow saw\rangle\}$ is not generated. The method, which is based on the bottom-up chart parsing, does not connect terms. Consequently, the correct dependency structure is not always generated for any sentence fragment, but has the feature that a fast parsing is executed. In the following, this straightforward method is called the baseline method, for which the same experiment is run for comparison.

Table 1 shows the precision of the proposed method, the method of Section 2, and the baseline method. In the following, the proposed method and the method of Section 2 are compared first, and then the proposed method and the baseline method are compared.

4.2.1. Comparison of proposed method and method of Section 2

The method of Section 2 requires a very long parsing time compared to the proposed method, and cannot generate the result of parsing for a considerable number of sentence fragments. The precision is low, being 8.9%. This result indicates that the parsing speed of the method of Section 2 is insufficient. On the other hand, the precision of the proposed method is 77.0%, which is higher than the method of Section 2 by as much as 68.1%. This result indicates that the parsing efficiency is effectively improved by the proposed method.

Table 1. Precision of incremental dependency parsing

	Precision (%)	Ratio of sentence fragments for which the parsing result is obtained (%)
Proposed method	77.0	77.0
Method of Section 2	8.9	8.9
Baseline method	64.2	88.9

Table 2. The number of correctly parsed fragments

Success/failure of correct dependency structure		Number of sentences
Proposed method	Baseline method	
Success	Success	2829
Success	Failure	968
Failure	Success	335
Failure	Failure	799
Total		4931

4.2.2. Comparison of proposed method and baseline method

The proposed method and the baseline method differ in that the latter does not include the term connection based on the reachability. Consequently, the comparison of the proposed method and the baseline method can be interpreted as the evaluation of the effect of the term connection based on the reachability. From such a viewpoint, 4931 sentence fragments are examined and are divided into classes according to the method by which the correct dependency structure is computed. Table 2 shows the results.

The number of sentence fragments for which the correct dependency structure is computed only by the proposed method is 968, which is considerable. For all of those sentence fragments, the baseline method computed the result of parsing within the time limit. In other words, for those 968 fragments, the correct dependency structure cannot be computed unless the connecting term is generated.

On the other hand, there are 335 sentence fragments for which the correct dependency structure is calculated only by the baseline method. Those are the case for which the dependency structure cannot be computed by the proposed method within the time limit. This is due to the fact that the parsing time is shorter in the baseline method, by the amount that the term connection based on the reachability is not included.[†]

Since the set of sentence fragments, for which the correct dependency structure can be computed by one of the methods, does not include that for the other, it is not possible to decide simply which method is better, but it is

[†]In the baseline method, the average parsing time per sentence is 0.009 s.

seen from Table 1 that the rate of correct result is higher in the proposed method.

5. Conclusion

This paper considered the situation where the words of a sentence are successively inputted from the left, and proposed a method which incrementally computes all dependency structures that are grammatically probable. It is shown that, by including a process to compute the dependency structure from the parse tree in the CFG-based incremental parsing, the incremental dependency parsing is realized.

As a method to realize a more efficient incremental dependency parsing, a method based on the reachability, as a relation between categories, is proposed. It is theoretically shown that the method based on the reachability can generate the dependency structure, which is equivalent to that obtained by the incremental parsing. It is shown experimentally that the method is effective in reducing the parsing time.

Up to the present, a method has been reported which tries to improve the efficiency of the incremental parsing by utilizing the dependency [18]. In that method, however, the dependency structure is computed from the partial parse tree. It is expected that, by combining the dependency structure computation proposed in this paper with such a method, the efficiency of the incremental parsing will be improved. Such an approach will be investigated at a later time.

Acknowledgments. The authors thank members of the laboratory for assisting in this study. They also thank the reviewer for providing useful comments, which helped to improve this paper.

REFERENCES

1. Allen J, Ferguson G, Stent A. An architecture for more realistic conversational systems. Proc Intelligent User Interfaces, p 1–8, Santa Fe, NM, 2001.
2. Alshawi H, Bangalore S, Douglas S. Learning dependency translation models as collections of finite state head transducers. Computational Linguistics 2000;26:45–60.
3. Amtrup JW. Incremental speech translation. Lecture Notes in Artificial Intelligence No. 1735. Springer-Verlag; 1999.
4. Bröker N. Separating surface order and syntactic relations in a dependency grammar. Proc 17th Int Conf on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, p 174–180, Montreal, 1998.
5. Collins MJ. A new statistical parser based on bigram lexical dependencies. Proc 34th Annu Meeting of the Association for Computational Linguistics, p 184–191, Santa Cruz, 1996.
6. Collins M. Head-driven statistical models for natural language parsing. PhD dissertation, University of Pennsylvania, 1999.
7. Eisner JM. Three new probabilistic models for dependency parsing: An exploration. Proc 16th Int Conf on Computational Linguistics, p 340–345, Copenhagen, 1996.
8. Ehsani F, Hatazaki K, Noguchi J, Watanabe T. Interactive speech dialogue system using simultaneous understanding. Proc 3rd Int Conf on Spoken Language Processing, p 879–882, Yokohama, 1994.
9. Lombardo V, Lesmo L. An Earley-type recognizer for dependency grammar. Proc 16th Int Conf on Computational Linguistics, p 723–728, Copenhagen, 1996.
10. Kay M. Algorithm schemata and data structures in syntactic processing. Technical Report, CSL-80-12, Xerox PARC, 1980.
11. Marcus MP, Santorini B, Marcinkiewicz MA. Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics 1993;19:310–330.
12. Matsubara S, Asai S, Toyama K, Inagaki Y. Chart-based parsing and transfer in incremental spoken language translation. Proc 4th Natural Language Processing Pacific Rim Symposium, p 521–524, Phuket, Thailand, 1997.
13. Matsubara S, Toyama K, Inagaki Y. Sync/Trans: Simultaneous machine interpretation between English and Japanese. Lecture Notes in Artificial Intelligence 1999;1747:134–143.
14. Matsubara S, Watanabe Y, Toyama K, Inagaki Y. Incremental sentence production for English–Japanese spoken language translation. Tech Rep NL, Inf Process Soc, NL-132, p 95–100, 1999.
15. Matsubara S, Takagi A, Kawaguchi N, Inagaki Y. Bilingual spoken monologue corpus for simultaneous machine interpretation research. Proc 3rd Int Conf on Language Resources and Evaluation, Vol. I, p 153–159, Canary Islands, 2002.
16. Milward D, Cooper R. Incremental interpretation: Applications, theory, and relationship to dynamic semantics. Proc 15th Int Conf on Computational Linguistics, p 748–754, Kyoto, 1994.
17. Mima H, Iidam H, Furuse O. Simultaneous interpretation utilizing example-based incremental transfer. Proc 17th Int Conf on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, p 855–861, Montreal, 1998.
18. Murase T, Matsubara S, Kato Y, Inagaki Y. Incremental CFG parsing with statistical lexical depend-

encies. Proc 6th Natural Language Processing Pacific Rim Symposium, p 353–358, Tokyo, 2001.
19. Nakano M, Miyazaki N, Hirasawa J, Dohsaka K, Kawabata T. Understanding unsegmented user utterances in real-time spoken dialogue systems. Proc 37th Annual Meeting of the Association for Computational Linguistics, p 200–207, College Park, USA, 1999.
20. Nakano M, Dosaka K. Language and dialogue processing in spoken dialogue system. J Soc Artif Intell 2002;17:271–278.
21. Sleator D, Temperley D. Parsing English with a link grammar. Carnegie Mellon University Computer Science Tech Rep, CMU-CS-91-196, 1991.

APPENDIX

Proof of Theorem 8

Several definitions to be used in the proof are given as the first step. Let $T(\mathbf{w})$ be the set of terms for the word sequence $\mathbf{w}$, which is generated by the standard bottom-up chart parsing. Let σ be a term. Then, $cat(\sigma)$ represent the category of σ. Let $\tau = \sigma_1 R_1 \cdots R_{n-1}\sigma_n$ be a connecting term, and let $cat_C(\tau) = cat(\sigma_1)$. When the grammatical rules are applied successively to term σ following Operation 4, let the obtained term be $apply(\sigma, r_1 \cdots r_q)$. In the above representation, it is tacitly assumed that $r_1, r_2, \ldots, r_q$ can be applied successively to σ. The term that is obtained by replacing the leftmost undecided term of term σ by σ' is written as $rep(\sigma, \sigma')$.

For the word sequence $\mathbf{w}$, the set $I(\mathbf{w})$ of terms, which is generated by the incremental chart parsing, is defined as follows.

[Definition 9] (term generated by incremental chart parsing). For the word sequence, the function I_n that returns the set of terms is defined as follows.

(1) For any word sequence $\mathbf{w}$, let $I_0(\mathbf{w}) = T(\mathbf{w})$.

(2) For any word sequence $\mathbf{w}$, let $I_{i+1}(\mathbf{w}) = I_{op4,i+1}(\mathbf{w}) \cup I_{op5,i+1}(\mathbf{w})$, where $I_{op4,i+1}(\mathbf{w})$ and $I_{op5,i+1}(\mathbf{w})$ are defined as follows.

$I_{op4,i+1}(\mathbf{w})$
$= \{\sigma | \sigma' \in I_i(\mathbf{w})$ and there exists a grammatical rule r such that $\sigma = apply(\sigma', r)\}$.

$I_{op5,i+1}(\mathbf{w})$
$= \{\sigma |$ there exists $\sigma_1 \in T(\mathbf{w}_1), \sigma_2 \in I_i(\mathbf{w}_2)(\mathbf{w} = \mathbf{w}_1\mathbf{w}_2)$ such that $\sigma = rep(\sigma_1, \sigma_2)\}$.

Furthermore, using I_n, $I(\mathbf{w})$ is defined as

$$I(\mathbf{w}) = \bigcup_{n=0}^{\infty} I_n(\mathbf{w})$$

□

$I_n(\mathbf{w})$ is the set of terms which is obtained by applying Operations 4 and 5 for n times to the terms which are generated by applying the standard bottom-up chart parsing to $\mathbf{w}$.

For the word sequence $\mathbf{w}$, the set $C(\mathbf{w})$ of connecting terms which is generated by the proposed method is defined as follows.

[Definition 10] (set of connecting terms) For the word sequence, the function C_n that returns the set of connecting terms is defined as follows.

(1) For any word sequence $\mathbf{w}$, let $C_1(\mathbf{w}) = T(\mathbf{w})$.

(2) For any word sequence $\mathbf{w}$, let

$C_{i+1}(\mathbf{w})$
$= \{\tau |$ there exists $\sigma_1 \in T(\mathbf{w}_1)$ and $\tau_2 \in C_i(\mathbf{w}_2)(\mathbf{w} = \mathbf{w}_1\mathbf{w}_2)$ such that τ is the connecting term $\sigma_1 \&_h \tau_2$ or $\sigma_1 \&_d \tau_2$.

$C(\mathbf{w})$ is defined as follows.

$$C(\mathbf{w}) = \bigcup_{n=1}^{\infty} C_n(\mathbf{w})$$

□

$C_n(\mathbf{w})$ is the set of connecting terms that are obtained by connecting n terms.

The lemmas needed in the proof of Theorem 8 are as follows.

[Lemma 11] For an arbitrary n, if $\sigma \in I_n(\mathbf{w})$, there exist $\sigma' \in I_{op5,j}(\mathbf{w}) \cup T(\mathbf{w})$ $(j \le n)$, and grammatical rules r_p $(p = 1, \ldots, q = n - j)$ such that $\sigma = apply(\sigma', r_1 \cdots r_q)$.

(Proof) This is obvious from the definition of $I_n(\mathbf{w})$.

□

[Lemma 12] Let $\sigma \in I(\mathbf{w})$ and let $r_1 r_2 \cdots r_q$ be a sequence of q grammatical rules. Then, $dep(apply(\sigma, r_1 \cdots r_q)) = dep(\sigma)$.

(Proof) This property is shown by the induction in regard to q. When $q = 0$, there applies $apply(\sigma, \varepsilon) = \sigma$, and the lemma is obviously true. Assume that the lemma is true for $q = n$. Let $q = n + 1$. Let $r_1 = A \rightarrow XY \cdots Z$ and $\mathbf{r} = r_2 \cdots r_{n+1}$. Then,

$$apply(\sigma, r_1 r_2 \cdots r_{n+1}) = apply([\sigma[?]_Y \cdots [?]_Z]_A, \mathbf{r}) \quad \text{(A.1)}$$

and

$$\begin{aligned} &dep(apply(\sigma, r_1 r_2 \cdots r_{n+1})) \\ &= dep(apply([\sigma[?]_Y \cdots [?]_Z]_A, \mathbf{r})) \quad ((\text{A.1})) \\ &= dep([\sigma[?]_Y \cdots [?]_Z]_A) \quad \text{(assumption of conduction)} \\ &= dep(\sigma) \quad \text{(Definition 2)} \end{aligned}$$

Thus, Lemma 12 is shown by induction. □

[Lemma 13] Let $\sigma \in I(\mathbf{w})$, and let $r_p (p = 1, \cdots, q)$ be a grammatical rule such that $hc(r_p) = 1$. Then, $head(apply(\sigma, r_1 \cdots r_q)) = head(\sigma)$.

(Proof) The property is shown by induction in regard to q. It is obvious that the lemma is true when $q = 0$. Assume that the lemma is true for $q = n$. Let $q = n + 1$. Let $r_1 = A \rightarrow XY \cdots Z$ and $\mathbf{r} = r_2 \cdots r_{n+1}$. Then, Eq. (A.1) applies. On the other hand, since $hc(r_1) = 1$, consequently, by Definition 1,

$$head([\sigma[?]_Y \cdots [?]_Z]_A) = head(\sigma) \qquad (A.2)$$

and

$$\begin{aligned} &head(apply(\sigma, r_1 r_2 \cdots r_{n+1})) \\ &\quad = head(apply([\sigma[?]_Y \cdots [?]_Z]_A, \mathbf{r})) \quad ((A.1)) \\ &\quad = head([\sigma[?]_Y \cdots [?]_Z]_A) \quad \text{(assumption of induction)} \\ &\quad = head(\sigma) \quad ((A.2)) \end{aligned}$$

Thus, Lemma 13 is shown by induction. □

[Lemma 14] Let $\sigma \in I(\mathbf{w})$ and let $head(\sigma) = ?$. Let $r_p (p = 1, \ldots, q)$ be a grammatical rule. Then, $head(apply(\sigma, r_1 \cdots r_q)) = ?$.

(Proof) The property is shown by induction in regard to q. The lemma is obviously true for $q = 0$. Assume that the lemma is true for $q = n$. Let $q = n + 1$. Let $r_1 = A \rightarrow XY$ and $\mathbf{r} = r_2 \cdots r_{n+1}$. As the first step, it is shown for $apply(\sigma, r_1) = [\sigma[?]_Y \cdots [?]_Z]_A$ that

$$head([\sigma[?]_Y \cdots [?]_Z]_A) = ? \qquad (A.3)$$

The case is divided for the case of $hc(r_1) = 1$ and the case otherwise.

When $hc(r_1) = 1$:

$$\begin{aligned} head([\sigma[?]_Y \cdots [?]_Z]_A) &= head(\sigma) \quad \text{(Definition 1)} \\ &= ? \quad \text{(assumption of lemma)} \end{aligned}$$

When $hc(r_1) \neq 1$: Let H be the head child of r_1. Then,

$$\begin{aligned} &head([\sigma[?]_Y \cdots [?]_H \cdots [?]_Z]_A) \\ &\quad = head([?]_H) \quad \text{(Definition 1)} \\ &\quad = ? \quad \text{(Definition 1)} \end{aligned}$$

Thus, Eq. (A.3) is valid.

For $r_1\mathbf{r} = r_1 r_2 \cdots r_{n+1}$, the length of $\mathbf{r}$ is n and Eq. (A.3) is valid. Consequently, by the assumption of induction,

$$head(apply([\sigma[?]_Y \cdots [?]_Z]_A, \mathbf{r})) = ? \qquad (A.4)$$

Consequently,

$$\begin{aligned} &head(apply(\sigma, r_1 r_2 \cdots r_{n+1})) \\ &\quad = head(apply([\sigma[?]_Y \cdots [?]_Z]_A, \mathbf{r})) \quad ((A.1)) \\ &\quad = ? \quad ((A.4)) \end{aligned}$$

Thus, by induction, Lemma 14 is shown. □

[Lemma 15] Let $\sigma \in I(\mathbf{w})$, and let $r_p (p = 1, \ldots, q)$ be a grammatical rule. Furthermore, assume that there exists $r_i (1 \leq i \leq q)$ such that $hc(r_i) \neq 1$. Then, $head(apply(\sigma, r_1 \cdots r_q)) = ?$.

(Proof) Let $\sigma' = apply(\sigma, r_1 \cdots r_{i-1})$, and let $r_i = A \rightarrow XY \cdots Z$. Since $hc(r_i) \neq 1$, letting the head child of r_i be H,

$$\begin{aligned} &head(apply(\sigma', r_i)) \\ &\quad = head([\sigma'[?]_Y \cdots [?]_H \cdots [?]_Z]_A) \quad \text{(Definition of apply)} \\ &\quad = head([?]_H) \quad \text{(Definition 1)} \\ &\quad = ? \quad \text{(Definition 1)} \end{aligned}$$

Consequently, by Lemma 14, $head(apply(apply(\sigma', r_i), r_{i+1} \ldots r_q)) = ?$. In other words, $head(apply(\sigma, r_1), \ldots r_{i-1} r_i r_{i+1} \ldots r_q)) = ?$ and Lemma 15 is shown. □

[Lemma 16] Let $\sigma_1 \in T(\mathbf{w}_1)$, and let the leftmost undecided term of σ_1 be $[?]_{X_k}$. Also, let $\sigma_2 \in I(\mathbf{w}_2)$, $cat(\sigma_2) = X_k$. Let $\sigma = rep(\sigma_1, \sigma_2)$. Also, let $\tau_2 \in C(\mathbf{w}_2)$ and let $\tau = \sigma_1 \&_h \tau_2$ be the connecting term. Furthermore, let

$$head(\sigma_2) = head_C(\tau_2), \quad dep(\sigma_2) = dep_C(\tau_2)$$

Then,

$$head(\sigma) = head_C(\tau) \text{ and } dep(\sigma) = dep_C(\tau)$$

(Proof) Let $\sigma_1 = [[\cdots]_{X_1} \cdots [\cdots]_{X_{k-1}} [?]_{X_k} \cdots [?]_{X_m}]_A$ and let $h = hc(A \rightarrow X_1 \cdots X_{k-1} X_k \cdots X_m)$. As the first step, $head(\sigma) = head_C(\tau)$ is shown by dividing the cases into the case of $h = k$ and the case of $h \neq k$. When $h = k$,

$$\begin{aligned} head(\sigma) &= head(\sigma_2) \quad \text{(Definition 1)} \\ &= head_C(\tau_2) \quad \text{(by assumption of lemma)} \\ &= head_C(\tau) \quad \text{(Definition 6)} \end{aligned}$$

When $h \neq k$,

$$\begin{aligned} head(\sigma) &= head([\cdots]_{X_h}) \quad \text{(Definition 1)} \\ &= head_C(\tau) \quad \text{(Definition 6)} \end{aligned}$$

Thus, $head(\sigma) = head_C(\tau)$.

As the next step, $dep(\sigma) = dep_C(\tau)$ is shown. For this, it suffices to show that $d(\sigma) = d_C(\tau)$, using Definitions 2 and 7, as well as the assumption of lemma $dep(\sigma_2) = dep_C(\tau_2)$. The case is divided into the cases of $h = k$ and $h \neq k$. When $h = k$:

$$\begin{aligned} d(\sigma) &= \{\langle w_d \rightarrow head(\sigma_2)\rangle \mid w_d = head([\cdots]_{X_i}) \\ &\qquad (1 \leq i \leq m, i \neq h)\} \quad \text{(Definition 2)} \\ &= \{\langle w_d \rightarrow head_C(\tau_2)\rangle \mid w_d = head([\cdots]_{X_i}) \\ &\qquad (1 \leq i \leq m, i \neq h)\} \quad \text{(by assumption of lemma)} \\ &= d_C(\tau) \quad \text{(Definition 7)} \end{aligned}$$

When $h \neq k$:

$$
\begin{aligned}
&d(\sigma) \\
&= \{\langle w_d \rightarrow head([\cdots]_{X_h})\rangle \mid \\
&\quad w_d = head([\cdots]_{X_i})(1 \leq i \leq m, i \neq h, i \neq k) \\
&\quad \text{or} \quad w_d = head(\sigma_2)\} \quad \text{(Definition 2)} \\
&= \{\langle w_d \rightarrow head([\cdots]_{X_h})\rangle \mid \\
&\quad w_d = head([\cdots]_{X_i})(1 \leq i \leq m, i \neq h, i \neq k) \\
&\quad \text{or} \quad w_d = head_C(\tau_2)\} \quad \text{(by assumption of lemma)} \\
&= d_C(\tau) \quad \text{(Definition 7)}
\end{aligned}
$$

Thus, there holds $d(\sigma) = d_C(\tau)$. It follows from Definitions 2 and 7, as well as the assumption of the lemma that $dep(\sigma) = dep_C(\tau)$. Thus, Lemma 16 is shown. □

[Lemma 17] Let $\sigma_1 \in T(\mathbf{w}_1)$, and let the leftmost undecided term of σ_1 be $[?]_{X_k}$. Also let $\sigma_2 \in I(\mathbf{w}_2)$, $cat(\sigma_2) = X_k$ and let $\sigma = rep(\sigma_1, \sigma_2)$. Let $\tau_2 \in C(\mathbf{w}_2)$, and let $\tau = \sigma_1 \&_d \tau_2$ be the connecting term. Furthermore, let $head(\sigma_2) = ?$, $dep(\sigma_2) = dep_C(\tau_2)$. Then,

$$head(\sigma) = head_C(\tau) \text{ and } dep(\sigma) = dep_C(\tau)$$

(Proof) The proof of this property almost parallels that of Lemma 16. Let $\sigma_1 = [[\cdots]_{X_1} \cdots [\cdots]_{X_{k-1}}[?]_{X_k} \cdots [?]_{X_m}]_A$ and let $h = hc(A \rightarrow X_1 \cdots X_{k-1} X_k \cdots X_m)$. As the first step,

$$head(\sigma) = head_C(\tau)$$

is shown by dividing the case into the case of $h = k$ and the case of $h \neq k$. When $h = k$, there holds $head(\sigma) = head(\sigma_2) = ?$ and $head_C(\tau) = head([?]_{X_k}) = ?$. When $h \neq k$, there holds $head(\sigma) = head([\ \ldots\ ]_{X_k}) = head_C(\tau)$. Thus, $head(\sigma) = head_C(\tau)$.

The next step is to show that $dep(\sigma) = dep_C(\tau)$. For this, it suffices to show that $d(\sigma) = d_C(\tau)$. When $h = k$, both $d(\sigma)$ and $d_C(\tau)$ are ϕ. When $h \neq k$,

$$
\begin{aligned}
d(\sigma) \;&=\; \{\langle w_d \rightarrow head([\cdots]_{X_h})\rangle \mid \\
&\qquad w_d = head([\cdots]_{X_i})(1 \leq i \leq m, i \neq h)\} \\
&=\; d(\sigma_1) \\
&=\; d_C(\tau)
\end{aligned}
$$

Consequently, $d(\sigma) = d_C(\tau)$ and $dep(\sigma) = dep_C(\tau)$. Thus, Lemma 17 is shown. □

[Lemma 18] For arbitrary n and a word sequence $\mathbf{w}$, if $\sigma \in I_n(\mathbf{w})$, there exists $\tau \in C(\mathbf{w})$ satisfying the following (1) and (2):

(1) $dep(\sigma) = dep_C(\tau)$.

(2) If $\sigma \in I_{op5,n}(\mathbf{w}) \cup T(\mathbf{w})$, $cat(\sigma) = cat_C(\tau)$ and $head(\sigma) = head_C(\tau)$.

(Proof) The property is shown by induction in regard to n. When $n = 0$, there holds $\sigma \in I_0(\mathbf{w}) = T(\mathbf{w}) \subseteq C(\mathbf{w})$ and it is obvious that the lemma is true by letting $\tau = \sigma$. Assume that the lemma is true when $n = l$. Letting $\sigma \in I_{l+1}(\mathbf{w})$, there holds $\sigma \in I_{op4,l+1}(\mathbf{w})$ or $\sigma \in I_{op5,l+1}(\mathbf{w})$.

When $\sigma \in I_{op4,l+1}(\mathbf{w})$, it suffices to show that there exists $\tau \in C(\mathbf{w})$ satisfying condition (1) $dep(\sigma) = dep_C(\tau)$. Since $\sigma \in I_{op4,l+1}(\mathbf{w})$, there exists a grammatical rule r and $\sigma' \in I_l(\mathbf{w})$, such that $\sigma = apply(\sigma', r)$. By Lemma 12, $dep(\sigma) = dep(\sigma')$. Since $\sigma' \in I_l(\mathbf{w})$, there exists, by assumption of induction, $\tau \in C(\mathbf{w})$ such that $dep(\sigma') = dep_C(\tau)$, and there holds $dep(\sigma) = dep(\sigma') = dep_C(\tau)$.

When $\sigma \in I_{op5,l+1}(\mathbf{w})$, there exist σ_1 and σ_2 such that $\sigma_1 \in T(\mathbf{w}_1)$ and $\sigma_2 \in I_l(\mathbf{w}_2)$, $\mathbf{w} = \mathbf{w}_1\mathbf{w}_2$, and there holds $\sigma = rep(\sigma_1, \sigma_2)$. Since $\sigma_2 \in I_l(\mathbf{w}_2)$, it follows from Lemma 11 that there exist $\sigma_2' \in I_{op5,j}(\mathbf{w}_2) \cup T(\mathbf{w}_2)(j \leq l)$ and grammatical rules $r_p(p = 1, \ldots, q = l - j)$, such that $\sigma_2 = apply(\sigma_2', r_1 \cdots r_q)$. Then, by the assumption of induction, there exists $\tau_2 \in C(\mathbf{w}_2)$, such that

$$dep(\sigma_2') = dep_C(\tau_2) \tag{A.5}$$

$$cat(\sigma_2') = cat_C(\tau_2) \tag{A.6}$$

$$head(\sigma_2') = head_C(\tau_2) \tag{A.7}$$

Let the category of the leftmost undecided term of σ_1 be X. If $hc(r_p) = 1$ for any $r_p(1 \leq p \leq q)$, there holds $cat(\sigma_2') \overset{h}{\rightsquigarrow}{}^* X$, that is, $cat_C(\tau_2) \overset{h}{\rightsquigarrow}{}^* X$. Consequently, by Definition 10, $\sigma_1 \&_h \tau_2 \in C(\mathbf{w})$. For $\sigma_1 \&_h \tau_2$, there hold

$$
\begin{aligned}
dep(\sigma_2) &= dep(\sigma_2') \quad \text{(Lemma 12)} \\
&= dep(\tau_2) \quad \text{((A.5))}
\end{aligned}
$$

$$
\begin{aligned}
head(\sigma_2) &= head(\sigma_2') \quad \text{(Lemma 13)} \\
&= head(\tau_2) \quad \text{((A.7))}
\end{aligned}
$$

It follows from Lemma 16 that $dep(\sigma) = dep_C(\sigma_1 \&_h \tau_2)$ and $head(\sigma) = head_C(\sigma_1 \&_h \tau_2)$. There also holds $cat(\sigma) = cat(\sigma_1) = cat_C(\sigma_1 \&_h \tau_2)$.

When $hc(r_p) \neq 1$ for a certain $r_p(1 \leq p \leq q)$, there holds $cat(\sigma_2') \rightsquigarrow^* \overset{d}{\rightsquigarrow}\rightsquigarrow^* X$, that is, $cat_C(\tau_2) \rightsquigarrow^* \overset{d}{\rightsquigarrow}\rightsquigarrow^* X$. It follows then from Definition 10 that $\sigma_1 \&_d \tau_2 \in C(\mathbf{w})$. By Lemma 12, $dep(\sigma_2) = dep(\sigma_2')$ and $dep(\sigma_2) = dep_C(\tau_2)$. On the other hand, by Lemma 15, $head(\sigma_2) = ?$. Consequently, by Lemma 17, $dep(\sigma) = dep_C(\sigma_1 \&_d \tau_2)$ and $head(\sigma) = head_C(\sigma_1 \&_d \tau_2)$. Furthermore, there holds $cat(s) = cat(\sigma_1) = cat_C(\sigma_1 \&_d \tau_2)$.

Thus, the lemma is valid for $n = l + 1$. Thus, Lemma 18 is shown by induction. □

[Lemma 19] For arbitrary n and a word sequence $\mathbf{w}$, if $\tau \in C_n(\mathbf{w})$, there exists $\sigma \in I(\mathbf{w})$ satisfying $dep(\sigma) = dep_C(\tau)$, $head(\sigma) = head_C(\tau)$, and $cat(\sigma) = cat_C(\tau)$.

(Proof) The property is shown by induction in regard to n. When $n = 1$, the lemma is obviously valid, as is seen by setting $\sigma = \tau$ since $\tau \in C_1(\mathbf{w}) = T(\mathbf{w}) \subseteq I(\mathbf{w})$. Assume that the lemma is valid for $n = l$. Letting $\tau \in C_{l+1}(\mathbf{w})$, there exist certain $\sigma_1 \in T(\mathbf{w}_1)$ and $\tau_2 \in C_l(\mathbf{w}_2)$ ($\mathbf{w} = \mathbf{w}_1\mathbf{w}_2$), $R \in \{\&_h, \&_d\}$, such that $\tau = \sigma_1 R \tau_2$. By the assumption of induction, there exists $\sigma_2 \in I(\mathbf{w}_2)$ satisfying $dep(\sigma_2) = dep_C(\tau_2) head(\sigma_2) = head_C(\tau_2)$ and $cat(\sigma_2) = cat_C(\tau_2)$.

Let the category of the leftmost undecided term of σ_1 be X. When $R = \&_h$, there holds $cat_C(\tau_2) \overset{h}{\rightsquigarrow}{}^* X$. Then, there exist grammatical rules $r_p = A_p \rightarrow Y_p\alpha_p (p = 1, \ldots, q)$ such that $Y_1 = cat_C(\tau_2)$, $Y_p = A_{p-1} (1 < p \leq q)$, $A_q = X$, and $hc(r_p) = 1 (p = 1, \ldots, q)$ (where α_p is the sequence of categories). For σ_2 in the assumption of induction, let $\sigma_2' = apply(\sigma_2, r_1 \cdots r_q)$. Then, by Definition 9, there holds $\sigma_2' \in I(\mathbf{w}_2)$. Since $cat(\sigma_2') = X$, there exists $rep(\sigma_1, \sigma_2')$, which is an element of $I(\mathbf{w})$. Let it be σ. Then,

$$\begin{aligned} dep(\sigma_2') &= dep(\sigma_2) \quad \text{(Lemma 12)} \\ &= dep(\tau_2) \quad \text{(assumption of induction)} \end{aligned}$$

$$\begin{aligned} head(\sigma_2') &= head(\sigma_2) \quad \text{(Lemma 13)} \\ &= head(\tau_2) \quad \text{(assumption of induction)} \end{aligned}$$

By Lemma 16, $dep(\sigma) = dep_C(\tau)$ and $head(\sigma) = head_C(\tau)$. Furthermore, $cat(\sigma) = cat(\sigma_1) = cat_C(\tau)$.

When $R = \&_d$, $cat_C(\tau_2) \rightsquigarrow^* \overset{d}{\rightsquigarrow}{}^* X$. Consequently, there exist grammatical rules $r_p = A_p \rightarrow Y_p\alpha_p (1 \leq p \leq q)$ such that $Y_1 = cat_C(\tau')$, $Y_p = A_{p-1} (1 < p \leq q)$, and $A_q = X$. For a certain $r_p (1 \leq p \leq q)$, there holds $hc(r_p) \neq 1$. Letting $\sigma_2' = apply(\sigma_2, r_1 \cdots r_q)$, there holds $\sigma_2' \in I(\mathbf{w}_2)$. Since $cat(\sigma_2') = X$, there exists $rep(\sigma_1, \sigma_2')$, which is an element of $I(\mathbf{w})$. Let it be σ. By Lemma 12 and the assumption of induction, $dep(\sigma_2') = dep_C(\tau_2)$. By Lemma 15, $head(\sigma_2') =$?. Then it follows from Lemma 17 that $dep(\sigma) = dep_C(\tau)$ and $head(\sigma) = head_C(\tau)$. Furthermore, $cat(\sigma) = cat(\sigma_1) = cat_C(\tau)$. Thus, the lemma is valid for $n = k + 1$. In other words, Lemma 19 is shown by induction. □

It is obvious from Lemma 18 that $dep(I(\mathbf{w})) \subseteq dep_C(C(\mathbf{w}))$. It is also obvious from Lemma 19 that $dep(I(\mathbf{w})) \supseteq dep_C(C(\mathbf{w}))$. It follows from those that $dep(I(\mathbf{w})) = dep_C(C(\mathbf{w}))$. In other words, Theorem 8 is shown.

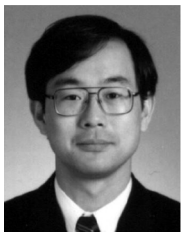
(end of proof for Theorem 8)

AUTHORS (from left to right)

Yoshihide Kato (member) graduated from Nagoya University in 1997, completed the doctoral program in 2003, and became a research associate at the Graduate School of International Development. He is engaged in research on natural language processing. He holds a D.Eng. degree, and is a member of Inf. Process. Soc. and Assoc. Natural Language Process.

Shigeki Matsubara (member) graduated from Nagoya Institute of Technology in 1993 and completed the doctoral program in 1998. He has been an associate professor at the Information Technology Center of Nagoya University since 2002. His research interests are natural language processing, spoken language processing, and machine translation. He holds a D.Eng. degree, and is a member of Inf. Process. Soc., Soc. Artif. Intell., Assoc. Natural Language Process., Japan Translation Soc., and ACL.

AUTHORS (continued) (from left to right)



Katsuhiko Toyama (member) graduated from Nagoya University in 1984 and completed the doctoral program in 1989. He is now an associate professor in the Graduate School of Information Science at Nagoya University. His research interests are logical knowledge representation and inference, as well as natural language understanding. He holds a D.Eng. degree, and is a member of Inf. Process. Soc., Assoc. Natural Language Process., Soc. Artif. Intell., and Japan Soc. Cogn. Sci.

Yasuyoshi Inagaki (member) graduated from Nagoya University in 1962 and completed the doctoral program in 1967. After serving professor at Mie University and at Nagoya University, he has been a professor at the Aichi Prefectural University since 2003. His research interests are theory of switching circuit, automaton and language theory, computation theory, basic theory of software, theory of parallel processing, algebraic specification description, basic theory of artificial intelligence, and natural language processing. He is a Vice-President and Fellow of IEICE. He holds a D.Eng. degree, and is a member of Inf. Process. Soc., Assoc. Natural Language Process., Soc. Artif. Intell., IEEJ, Japan Soc. Software Sci., Japan OR Soc., IEEE, ACM, and EATCS.