PAPER *Special Section on Natural Language Processing and its Applications*

Incremental Parsing with Adjoining Operation

Yoshihide KATO[†a)] ***and*** **Shigeki MATSUBARA**[†]**, *Members***

SUMMARY This paper describes an incremental parser based on an adjoining operation. By using the operation, we can avoid the problem of infinite local ambiguity. This paper further proposes a restricted version of the adjoining operation, which preserves lexical dependencies of partial parse trees. Our experimental results showed that the restriction enhances the accuracy of the incremental parsing.

key words: *incremental parsing, tree adjoining grammar, probabilistic parsing, Penn Treebank, real-time spoken language processing system*

1. Introduction

Incremental parser analyzes an input sentence from left to right and generates partial parse trees which connect all words in each initial fragment of the sentence. Incremental parsing is useful to realize real-time spoken language processing systems, such as a simultaneous machine interpretation system, an automatic subtitle generating system, or a spoken dialogue system [1]–[3].

Several incremental parsing methods have been proposed so far [4]–[6]. In these methods, the parsers can generate the candidates of partial parse trees on a word-by-word basis. However, they suffer from the problem of infinite local ambiguity, i.e., they may generate an infinite number of candidates of partial parse trees. This problem is caused by the fact that partial parse trees can have arbitrarily nested left-recursive structures and there is no information to determine the depth of nesting in advance.

To solve this problem, we introduce an adjoining operation to incremental parsing. An adjoining operation is used in Tree-Adjoining Grammar [7]. By using the operation, we can avoid the problem of infinite local ambiguity. This approach has been adopted by the previous methods [8], [9]. However, this raises another problem that their adjoining operation cannot preserve information which partial parse trees have. We formalize this problem from the viewpoint of lexical dependencies, and propose a restricted version of the adjoining operation, which preserves lexical dependencies. An experimental result showed that our method makes our incremental parser more accurate.

This paper is organized as follows: The next section describes the incremental parser proposed by Collins and Roark [4] and discusses its problem. Section 3 introduces an adjoining operation to incremental parsing and explains our incremental parser. In Sect. 4, we report an experimental evaluation of our incremental parser. Section 5 describes related works.

Manuscript received March 20, 2009.
Manuscript revised July 15, 2009.

[†]The authors are with Information Technology Center, Nagoya University, Nagoya-shi, 464–8601 Japan.
a) E-mail: yosihide@el.itc.nagoya-u.ac.jp

DOI: 10.1587/transinf.E92.D.2306

2. Incremental Parsing

This section gives a description of the Collins and Roark's incremental parser [4] and discusses its problem.

2.1 Collins and Roark's Incremental Parser

The Collins and Roark's incremental parser uses a grammar defined by a 6-tuple $G = (V, T, S, \#, C, B)$. V is a set of nonterminal symbols. T is a set of terminal symbols. S is called a start symbol and $S \in V$. # is a special symbol to mark the end of constituent. C is a set of *allowable chains*. An allowable chain is a sequence of nonterminal symbols followed by a terminal symbol. Each chain corresponds to a label sequence on a path from a node to its leftmost descendant leaf. B is a set of *allowable triples*. An allowable triple is a tuple $\langle X, Y, Z\rangle$ where $X, Y, Z \in V$. The triple specifies which nonterminal symbol Z is allowed to follow a nonterminal symbol Y under a parent X.

For each initial fragment of an input sentence, the Collins and Roark's incremental parser generates partial parse trees which connect all words in the fragment. The parsing process proceeds on a word-by-word basis from left to right.

As an example, let us consider the following initial fragment:

We describe ... (1)

For the first word "we", the parser generates the partial parse tree (a) shown in Fig. 1, if the allowable chain $\langle$S $\rightarrow$ NP $\rightarrow$ PRP $\rightarrow$ we$\rangle$ exists in C. For other chains which start with S and end with "we", the parser generates partial parse trees by using the chains. For the next word, the parser attaches the chain $\langle$VP $\rightarrow$ VBP $\rightarrow$ describe$\rangle$ to the partial parse tree (a), i.e., generates the partial parse trees (b) shown in Fig. 1 [*]. The attachment is possible when the allowable triple $\langle$S, NP, VP$\rangle$ exists in B.

[*]More precisely, the chain is attached after attaching the end-of-constituent # under the NP node.

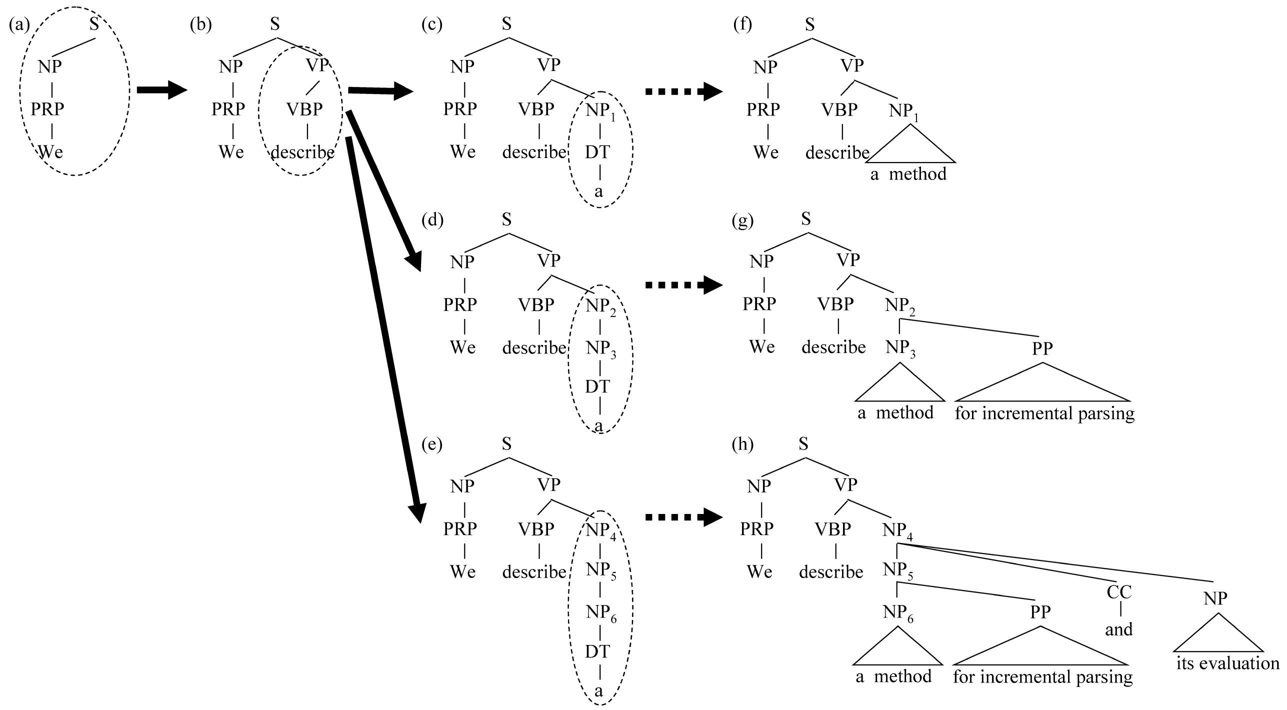

Fig. 1 A process in incremental parsing.

2.2 Infinite Local Ambiguity

Incremental parsing suffers from the problem of infinite local ambiguity. The ambiguity is caused by left-recursion. Incremental parser generates an infinite number of partial parse trees because it cannot know the depth of left-recursive embedding in advance.

As an example, let us consider the following initial fragment:

We describe a ... (2)

For the initial fragment, there exist several candidates of partial parse trees. Figure 1 shows candidates of partial parse trees. The partial parse tree (c) represents that the noun phrase NP_1 which starts with "a" has no adjunct (Parse trees such as (f) are derived from (c).). The partial parse tree (d) represents that the noun phrase NP_3 has an adjunct or is a conjunct of a coordinated noun phrase (See the parse tree (g).). The partial parse tree (e) represents that the noun phrase NP_6 has an adjunct and the noun phrase NP_5 with an adjunct is a conjunct of a coordinated noun phrase (See the parse tree (h).). The partial parse trees (d) and (e) are the instances of partial parse trees which have left-recursive structures. The problem is that there is no information to determine the depth of left-recursive nesting at this point. In principle, the depth can be arbitrary number, i.e., there exist an infinite number of candidates of partial parse trees [†].

3. Introduction of Adjoining Operation to Incremental Parsing

The previous section described the problem of infinite local ambiguity in incremental parsing. This section provides a solution to this problem.

To avoid the problem, previous works have adopted the following approaches:

1. A beam search strategy [4]–[6].
2. Limiting the allowable chains to those actually observed in the treebank [4].
3. Transforming the parse trees with a selective left-corner transformation [10] before inducing the allowable chains and allowable triples from the treebank [4].

The first and second approaches can prevent the parser from infinitely generating partial parse trees, but the parser has to generate partial parse trees as shown in Fig. 1, and the local ambiguity still remains. In the third approach, no left recursive structure exists in the transformed grammar, but the parse trees defined by the grammar are different from those defined by the original grammar. Therefore, it is not clear whether partial parse trees defined by the transformed grammar represent syntactic relations correctly.

[†]Regardless of whether the whole input sentence does or does not have a left-recursive structure, the incremental parser generates all partial parse trees such as (c), (d) and (e). Without the process of generating all partial parse trees, the parser cannot process some input sentences. For example, if the parser does not generate the partial parse tree (d), it cannot parse the sentence "We describe a method for incremental parsing."

In this paper, we adopt another approach which is the introduction of an adjoining operation to incremental parsing. Lombardo, et al. [8] and Kato, et al. [9] have already adopted this approach. However, their method causes another problem. Their adjoining operations cannot preserve the information which partial parse trees have. To solve the problem, this section proposes a restricted version of the adjoining operation.

3.1 Adjoining Operation

An adjoining operation is used in Tree-Adjoining Grammar [7]. The operation inserts a tree into another tree. The inserted tree is called an *auxiliary tree*. Each auxiliary tree has a leaf called a *foot* which has the same nonterminal symbol as its root. An adjoining operation is defined as follows:

adjoining An adjoining operation splits a parse tree σ at a nonterminal node η and inserts an auxiliary tree β having the same nonterminal symbol as η, i.e., combines the upper tree of σ with the root of β and the lower tree of σ with the foot of β (see Fig. 2).

We write $a_{\eta,\beta}(\sigma)$ for the partial parse tree obtained by adjoining β to σ at η.

We use simplest auxiliary trees, which consist of a root and a foot [†]. By using adjoining operation, we can avoid the problem of infinite local ambiguity.

As an example, let us consider the following sentence:

We describe a method for incremental parsing. (3)

As described in Sect. 2, the Collins and Roark's incremental parser generates partial parse trees such as (c), (d) and (e) for the initial fragment "We describe a". On the other hand, our parser generates only the partial parse tree (c). When a left-recursive structure is required during the process of parsing the sentence, our parser adjoins it. In the example above, the parser adjoins the auxiliary tree $\langle$NP $\rightarrow$ NP$\rangle$ to the partial parse tree (c) at the point when the word "for" is read, and attaches the allowable chain $\langle$PP $\rightarrow$ IN $\rightarrow$ for$\rangle$. The parsing process is shown in Fig. 3.

3.2 Adjoining Operation and Monotonicity

By using the adjoining operation, we avoid the problem of infinite local ambiguity. However, there arises another problem. The problem is that the adjoining operation cannot preserve the information which partial parse trees have. As an example, let us consider the following sentence:

We describe John 's method. (4)

The partial parse tree (a) shown in Fig. 4 is a candidate for the initial fragment "We describe John 's". The partial parse tree represents that the noun phrase "John 's" is the object of the verb "describe". When the parser reads the next word "method", it generates the partial parse tree (b) shown in Fig. 4 by adjoining the auxiliary tree $\langle$NP $\rightarrow$ NP$\rangle$ and attaches the allowable chain $\langle$NN $\rightarrow$ method$\rangle$ to generate the

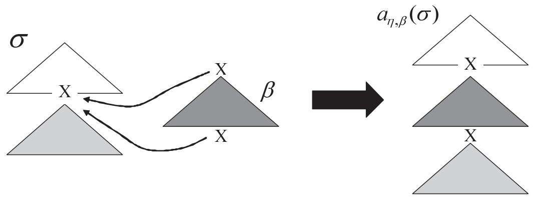

Fig. 2 Adjoining operation.

partial parse tree (c). The partial parse tree (c) represents that the noun phrase "John 's" is the determiner of the noun "method". The role of the noun phrase "John 's" is changed by applying the adjoining operation. This example demonstrates that the adjoining operation cannot preserve the information which partial parse trees have.

To formalize the problem described above, we focus on lexical dependencies of partial parse trees. A lexical dependency is a kind of relation between two words (a *dependent* and a *head*), for which we write $\langle$*dependent* $\rightarrow$ *head*$\rangle$. We can map parse trees to sets of lexical dependencies by identifying the head-child of each constituent in the parse tree [11].

First, we define the monotonicity of the adjoining operation. We say that adjoining an auxiliary tree β to a partial parse tree σ at a node η is *monotonic* when $dep(\sigma) \subseteq dep(a_{\eta,\beta}(\sigma))$ where *dep* is the mapping from a parse tree to a set of dependencies. An auxiliary tree β is monotonic if adjoining β to any partial parse tree is monotonic.

Let us go back to the example above. We mark each head-child with a special symbol $*$. We obtain three lexical dependencies $\langle$We $\rightarrow$ describe$\rangle$, $\langle$John $\rightarrow$'s$\rangle$ and $\langle$'s $\rightarrow$ describe$\rangle$ from the partial parse tree (a) shown in Fig. 4. However, the result of applying the adjoining operation to (a) does not have $\langle$'s $\rightarrow$ describe$\rangle$, i.e., the application is not monotonic.

To prevent the non-monotonic application of the adjoining operation, we restrict the form of auxiliary trees. All auxiliary trees must have the following forms:

adjunct or coordinated structure: The foot of an auxiliary tree is the head-child of its parent.

topicalized sentence: The foot of an auxiliary tree is a topicalized sentence.

The auxiliary tree $\langle$NP $\rightarrow$ NP$\rangle$ is in neither of the above forms. As shown in the example above, adjoining the auxiliary tree to the partial parse tree (a) shown in Fig. 4 is nonmonotonic. Therefore, our method processes sentence (4) without the adjoining operation as shown in Fig. 5. On the other hand, the auxiliary tree $\langle$NP $\rightarrow$ NP$^{*}\rangle$ is in the first form. As shown in Fig. 3, adjoining the auxiliary tree is monotonic.

Finally, we explain the second form of auxiliary tree.

[†]Since our auxiliary trees have no inner nodes, they cannot deal with indirect left-recursive structures such as `(NP (ADJP (NP (DT a) (NN lot)) (JJR more)) (NN space))`. However, there are few occurrences of indirect left-recursive structures in English. See Sect. 4.1.

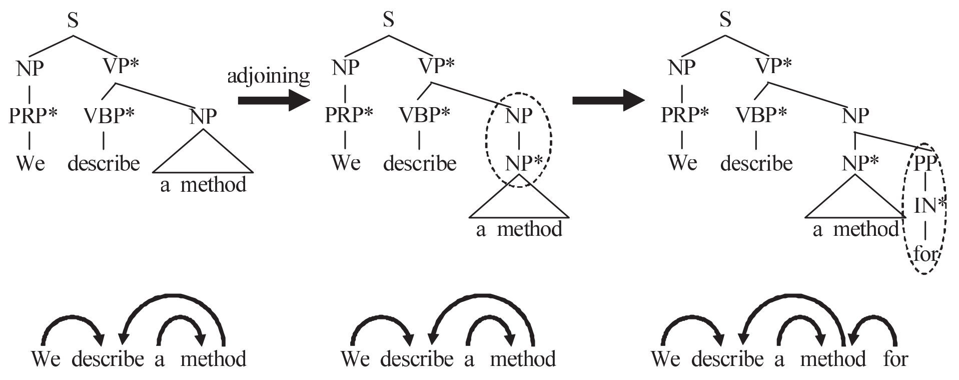

Fig. 3 Adjoining in incremental parsing.

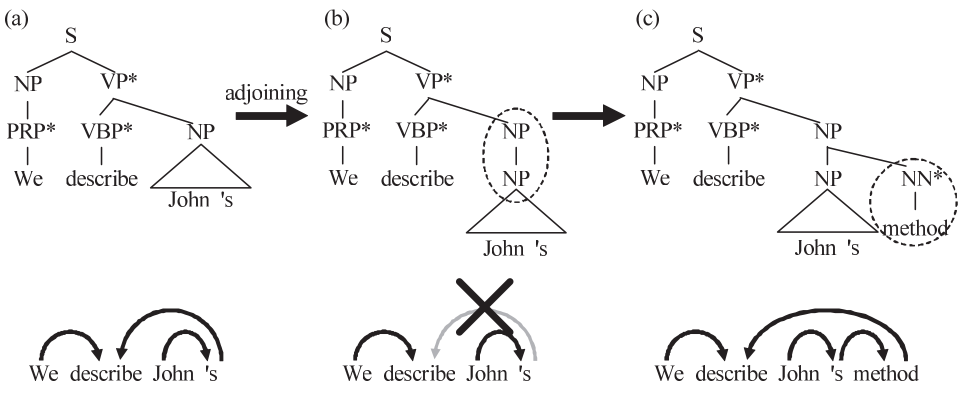

Fig. 4 An example of non-monotonic adjoining operation.

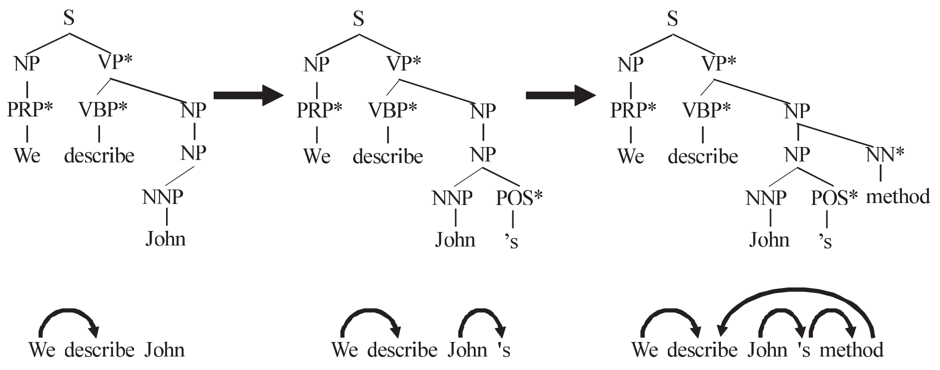

Fig. 5 Incremental parsing prosess for sentence (4).

The auxiliary tree in the second form is non-monotonic. However, topicalized sentences occur in the leftmost position in a parse tree and adjoining auxiliary trees at any leftmost node does not destroy the lexical dependencies as shown in Fig. 6, i.e., this type of auxiliary trees can be treated as monotonic auxiliary trees.

3.3 Our Incremental Parser

This section provides a description about our incremental parser. Our incremental parser is based on a probabilistic parsing model. First, we describe a grammar used by our parser. Next, we propose a probabilistic model which assigns a probability to each partial parse tree. Finally, we explain the parsing strategy.

3.3.1 Grammar Extraction

To build a wide coverage grammar, we extracted a grammar from a treebank. We decomposed parse trees in the treebank into allowable chains and auxiliary trees. The decomposition is a reverse process of incremental parsing. The decomposition procedure is shown in Fig. 7.

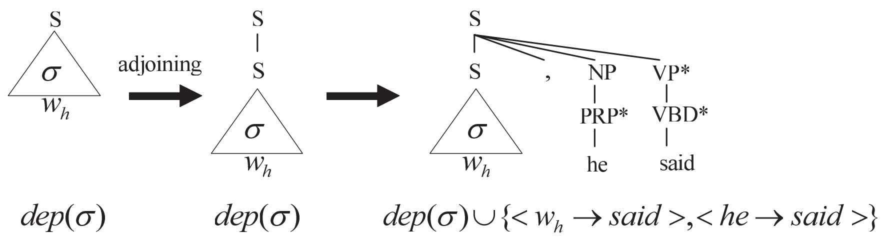

Fig. 6 Adjoining operation for processing topicalized sentences.

```
$decompose(node)$
    if $node$ has no siblings $\wedge$
      $node.label = node.parent.label$ $\wedge$
      $node$ is the head-child or a topicalized sentence then
          extract $node$ and $node.parent$ as an auxiliary tree
          $decompose(node)$
    else if $node$ is preterminal then
          $extractChain(node)$
    else
          extract end-of-constituent chain $\langle \# \rangle$
          $decompose(node.rightmostChild)$

$extractChain(node)$
    $ancestor = node$
    while $ancestor$ has no siblings do
      $ancestor = ancestor.parent$
    extract the chain from $ancestor$ to $node$
    $decompose(ancestor.leftSibling)$
```

Fig. 7 The procedure for extracting allowable chains and auxiliary trees.

3.3.2 Probabilistic Model

This section describes a probabilistic model that evaluates partial parse trees in incremental parsing. The model assigns a probability to each operation. The probability of a partial parse tree is defined by the product of the probabilities of the operations used in its construction.

We decompose the process of attaching an allowable chain into three steps to overcome a data sparseness problem. That is, the probability of attaching the allowable chain c to the partial parse tree σ is defined as follows:

$$\begin{aligned} & P_{attach}(c \mid \sigma) \\ & \quad = P_{root}(root(c) \mid \sigma) \\ & \qquad \times P_{template}(template(c) \mid \sigma, root(c)) \\ & \qquad \times P_{word}(word(c) \mid \sigma, template(c)) \end{aligned} \tag{5}$$

where $root(c)$ is the root label of c, $template(c)$ is the sequence which is obtained by omitting the last element from c and $word(c)$ is the last element of c. On the other hand, the process of adjoining is not decomposed since all auxiliary trees in our grammar are very simple.

We estimate the probability by using the following features:

- a current node label
- ancestor node labels
- left-sibling node labels
- a head node label
- a head word and its pos tag
- distance features

The features are similar to those of the previous methods.

It is worth noting that the non-monotonic adjoining operation may be harmful for building the probabilistic model. We illustrate such situations. As an example, let us consider sentence (4). The probability of the word "'s" is conditioned on its head word. However, the head word is identified by using the partial parse tree (a) shown in Fig. 4, i.e., the head word is "describe" and the probability may be estimated incorrectly. Moreover, this example shows that the model estimates the probability distributions without distinguishing the object NP and possessive NP. This fact also may cause a negative effect in building the model. As an example, let us consider the construction of the possessive NP "John 's." In the case where the non-monotonic adjoining operation is allowed, the probability of constructing the possessive NP is conditioned on the partial parse tree (b) shown in Fig. 1. The probability of constructing the object NP such as "a method" is also conditioned on the same partial parse tree. This means that the model estimates the probability of constructing the possessive NP and constructing the object NP according to the same distribution. However, the construction of possessive NPs must be distributed according to the different distribution. The reason is that possessive NPs always include possessive ending POS while object NPs rarely include it. On the other hand, when the non-monotonic adjoining operation is not allowed, the probability of constructing the possessive NP is conditioned on the partial parse trees shown in Fig. 5. At the point when the allowable chain ⟨NP→NP→NNP→John⟩ is attached, the possessive NP has the parent labelled with NP. This information enables the model to identify the NP as possessive. Therefore, the model can estimate the probability of constructing the possessive NP according to the different distribution. In Sect. 4, we assess such influence on probabilistic model.

3.3.3 Parsing Strategy

To achieve efficient parsing, we use a beam search strategy like the previous methods. For each word position i, our parser has a priority queue H_i. Each queue H_i stores the only N-best partial parse trees and discards the others. Moreover, the partial parse tree σ is discarded, if its probability $P(\sigma)$ is less than the $P^*\gamma$ where P^* is the highest probability on the queue H_i and γ is a beam factor.

4. Experimental Evaluation

To evaluate the performance of our incremental parser, we conducted a parsing experiment. We implemented the following three types of incremental parsers to assess the influence of the adjoining operation and its monotonicity:

- without the adjoining operation
- with the non-monotonic adjoining operation
- with the monotonic adjoining operation

The grammars were extracted from the parse trees in section 02-21 of the Wall Street Journal in Penn Treebank [12]. We discard the allowable chains which occured only once in these sections. We identified the head-child in each constituent by using the head rule of Collins [11]. Moreover, we added the information, such as gaps, arguments and coordinated phrases by using the rules in the literature [13]. The probabilistic models were built by using the maximum entropy method [14]. We set the beam-width N to 300 and the beam factor γ to 10^{-11}

4.1 Statistics of Grammars

Table 1 shows the number of different types of allowable chains and the coverage of the grammars. The coverage of a grammar is defined as follows:

$$\frac{\sum_{c \in C \cap C_{test}} freq(c)}{\sum_{c \in C_{test}} freq(c)} \tag{6}$$

where C is the set of allowable chains of the grammar, C_{test} is the set of allowable chains extracted from test corpus and $freq(c)$ is the number of occurrences of allowable chain c in the test corpus. We used section 24 for evaluation of the coverage. The coverage of each grammar is very high. This means that each grammar provides the allowable chains enough to parse sentences correctly.

We counted the frequency of occurrences of allowable chains which include indirect left-recursive structures. Our method cannot deal with indirect left-recursive structures by adjoining operation. We observed that only 57 occurrences of chains (0.01%) include indirect left-recursive structures. This means that the simplest auxiliary trees cover the majority of left-recursive structures.

4.2 Impacts of Adjoining and Its Monotonicity

We evaluated the parsing accuracy by using section 23. We measured labeled recall and labeled precision [15]. Table 2 shows the results. Our incremental parser is competitive with the previous methods[†]. The incremental parser with the monotonic adjoining operation outperforms the others. The incremental parser with non-monotonic adjoining does not outperform the one with no adjoining. The results mean that our proposed constraint of auxiliary trees improves parsing accuracy.

To investigate whether there exist any influences other than those of incorrect lexical dependencies, we estimated the models in which the lexical dependencies features are removed. The results are shown in Table 3. The incremental parser with non-monotonic adjoining yielded the worst result. This result demonstrates that the non-monotonic adjoining operation not only destroys lexical dependencies but also has a bad influence on the probabilistic model.

Table 1 Statistics of the grammars.

Grammar	type	coverage(%)
No adjoining	2792	99.5
Non-monotonic adjoining	1740	99.7
Monotonic adjoining	1839	99.7

Table 2 Parsing results.

	LR(%)	LP(%)	F(%)
Roark 2004	86.4	86.8	86.6
Collins et al. 2004 (w/o punctuation features)	86.5	86.8	86.7
Collins et al. 2004 (w/ punctuation features)	88.4	89.1	88.8
No adjoining	86.3	86.8	86.6
Non-monotonic adjoining	86.1	87.1	86.6
Monotonic adjoining	87.2	87.7	87.4

Table 3 Parsing results without lexical dependencies.

	LR(%)	LP(%)	F(%)
No adjoining	86.1	86.8	86.5
Non-monotonic adjoining	85.1	86.2	85.6
Monotonic adjoining	86.6	87.3	86.9

5. Psycholinguistic Models Using Adjoining Operation

Incremental parsing has been investigated in the area of psycholinguistics. This section describes the relation between our method and psycholinguistic parsing models.

In psycholinguistic literature, it is widely assumed that human language processing is incremental. Sturt, et al. [16] models a process of recognizing VP coordination by using an adjoining operation. Demberg, et al. [17] and Mazzei, et al. [18] propose an incremental version of the tree-adjoining grammar. In their grammars, the derivation process proceeds from left to right and each derived tree is a fully connected tree. These properties are suitable for incremental parsing.

These researches support the validity of introducing the adjoining operation to incremental parsing. The difference between our method and these psycholinguistic model is that these models lack mechanisms of resolving structural ambiguity of initial fragments, while our method has the probabilistic model to rank partial parse trees.

[†]The Collins and Roark's parser achieves the best performance, but we cannot compare it directly with our incremental parser. The reason is that the Collins and Roark's parser utilizes the final punctuation as a feature. Our incremental parser is strictly on a word-by-word basis, while Collins and Roark's parser is not. Especially, using final punctuation means that the parser cannot generate any partial parse trees until it reads the whole sentence. This property is unsuitable for real-time spoken language processing systems.

6. Conclusion

This paper introduced an adjoining operation to incremental parsing in order to solve the problem of infinite local ambiguity. The adjoining operation causes another problem that the parser cannot preserve lexical dependencies of partial parse trees. To tackle this problem, we defined the monotonicity of adjoining operation and restricted the form of auxiliary trees to satisfy the constraint of the monotonicity. Furthermore, our experimental result demonstrated that the monotonic adjoining operation makes incremental parsing more accurate.

In future work, we will investigate the incremental parser for head-final languages such as Japanese. Unlike English, head-final languages include many indirect left-recursive structures. In this paper, we dealt only with direct left-recursive structures. To process indirect left-recursive structures, we need to extend our method.

References

[1] G. Aist, J. Allen, E. Campana, C.G. Gallo, S. Stoness, M. Swift, and M.K. Tanenhaus, "Incremental understanding in human-computer dialogue and experimental evidence for advantages over nonincremental methods," Proc. 11th Workshop on the Semantics and Pragmatics of Dialogue, ed. R. Artstein and L. View, pp.149–154, June 2007.

[2] J. Allen, G. Ferguson, and A. Stent, "An architecture for more realistic conversational systems," Proc. International Conference of Intelligent User Interfaces, pp.1–8, Jan. 2001.

[3] Y. Inagaki and S. Matsubara, "Models for incremental interpretation of natural language," Proc. 2nd Symposium on Natural Language Processing, pp.51–60, Aug. 1995.

[4] M. Collins and B. Roark, "Incremental parsing with the perceptron algorithm," Proc. 42nd Meeting of the Association for Computational Linguistics (ACL'04), pp.111–118, Main Volume, Barcelona, Spain, July 2004.

[5] B. Roark, "Probabilistic top-down parsing and language modeling," Computational Linguistics, vol.27, no.2, pp.249–276, June 2001.

[6] B. Roark, "Robust garden path parsing," Natural language engineering, vol.10, no.1, pp.1–24, March 2004.

[7] A.K. Joshi, "Tree adjoining grammars: How much context sensitivity is required to provide a reasonable structural description?," in Natural Language Parsing, ed. D.R. Dowty, L. Karttunen, and A.M. Zwicky, pp.206–250, Cambridge University Press, 1985.

[8] V. Lombardo and P. Sturt, "Incremental processing and infinite local ambiguity," Proc. 19th Annual Conf. of the Cognitive Science Society, pp.448–453, Aug. 1997.

[9] Y. Kato, S. Matsubara, and Y. Inagaki, "Stochastically evaluating the validity of partial parse trees in incremental parsing," Proc. ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together, pp.9–15, July 2004.

[10] M. Johnson and B. Roark, "Compact non-left-recursive grammars using the selective left-corner transform and factoring," Proc. 18th International Conference on Computational Linguistics, pp.355–361, July 2000.

[11] M. Collins, Head-Driven Statistical Models for Natural Language Parsing, Ph.D. thesis, University of Pennsylvania, 1999.

[12] M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," Computational Linguistics, vol.19, no.2, pp.310–330, June 1993.

[13] D.M. Bikel, "Intricacies of Collins' parsing model," Computational Linguistics, vol.30, no.4, pp.478–511, Dec. 2004.

[14] A.L. Berger, S.A.D. Pietra, and V.J.D. Pietra, "A maximum entropy approach to natural language processing," Computational Linguistics, vol.22, no.1, pp.39–71, March 1996.

[15] E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski, "A procedure for quantitatively comparing the syntactic coverage of English grammars," Proc. 4th DARPA Speech and Natural Language Workshop, pp.306–311, Feb. 1991.

[16] P. Sturt and V. Lombardo, "Processing coordinated structures: Incrementality and connectedness," Cognitive Science, vol.2, no.29, pp.291–305, March 2005.

[17] V. Demberg and F. Keller, "A psycholinguistically motivated version of TAG," Proc. 9th International Workshop on Tree Adjoining Grammars and Related Formalisms, June 2008.

[18] A. Mazzei, V. Lombardo, and P. Sturt, "Dynamic TAG and lexical dependencies," Research on Language and Computation, vol.5, no.3, pp.309–332, Sept. 2007.

Yoshihide Kato received the B.E. degree, the M.E. degree, and the Dr. of Engineering degree in information engineering from Nagoya University, in 1997, 1999 and 2003, respectively. He was an Assistant Professor from 2003 to 2009 at Graduate School of International Development, Nagoya University. He is currently a Researcher at Information Technology Center, Nagoya University. His research interests include natural language processing and information retrieval. He is a member of the IPSJ, the NLP and the ACL.

Shigeki Matsubara received the B.E. degree in electrical and computer engineering from Nagoya Institute of Technology in 1993, and the M.E. degree, and the Dr. of Engineering from Nagoya University in 1995 and 1998, respectively. He was an Assistant Professor from 1998 to 2002 at the Faculty of Language and Culture, Nagoya University. He is currently an Associate Professor at Information Technology Center, Nagoya University. His research interests include natural language processing, information retrieval and digital library. He is a member of the IPSJ, the JSAI, the NLP, the IEEE and the ACM.