Sentence Generation for Pattern-Based Machine Translation

Koichi Takeda
Tokyo Research Laboratory, IBM Research
1623-14 Shimotsuruma, Yamato, Kanagawa 242-8502, JAPAN
takeda@trl.ibm.co.jp

Abstract

In this paper we discuss sentence generation strategy for pattern-based machine translation and their computational properties. Even though sentence generation in general is known to be computationally expensive, there exists a polynomial-time algorithm for synchronized sentence analysis/generation for "fixed" pattern-based (and its variant) grammars. Experimental results of implemented algorithm showed that the K-best generation algorithm runs fast enough to be used for practical machine translation systems, and elapsed time ratio of analysis and generation was about 50:1.

1. Introduction

Grammar-based (sentence) generation has been one of the key issues in building a clean, compositional and declarative architecture for natural language processing (NLP) systems. Two types of generation algorithms -- top-down (Wedekind 1988) and head-driven (Shieber et al. 1990, Wilcock and Matsumoto 1998) algorithms -- have been known for more than a decade, and a wide range of computational aspects including efficiency (Martin 1992), termination, reversibility (Dymetman 1991), and generative power (Kaplan and Wedekind 2000) have been well studied. Grammar formalisms and lexical semantics can harness the generation grammar with a powerful descriptive power and a rich set of features, which allow us to give complex specifications for generating sentences.

Recent NLP-enabling technologies (in particular, for machine translation (MT) products), however, strongly suggest that efficient algorithms be exclusively employed since MT systems for translating Web pages are forming a huge market (e.g., SYSTRAN has been employed in the AltaVista WWW search engine) and Internet users may not be tolerant if translation of a Web page takes more than a few seconds. Grammar-based generation has yet to solve efficiency in addition to interlingua design problems for broad domains before being fully incorporated into the commercial MT.

Corpus-based approaches have steadily become the mainstream NLP research, and stochastic acquisition of lexicalized grammars (Charniak 1997), bilingual parsing (Wu 1997) and translation knowledge from bilingual corpora (Knight 1997) show promising results. Example-based MT (EBMT) (Sato and Nagao 1990, Somers 1999) and hybrid approach have been explored intensively to form a "data-driven" MT paradigm. In particular, synchronized grammars (Shieber and Schabes 1990) can be effectively combined with EBMT to capture "patterns" for efficiently accumulating translation knowledge (Takeda 1996, Watanabe and

Takeda 1998). Since patterns are based on a pair of CFG-style rules, they are easy to define even by end users for customizing the translation knowledge.

In this paper, we describe sentence generation for pattern-based MT systems. Even though it is known that generation in general is computationally intractable in the worst case, we have polynomial-time sentence analysis/generation algorithms for a fixed, synchronized context-free grammars. An experimental English-to-Japanese machine translation system has been implemented using the algorithms, and it achieved 0.63 second per sentence, or 19.6 words per second, for translating about 1700 sample sentences. One of the interesting observations was the 50:1 time ratio of analysis and generation, which is discussed in more details in Section 4.

2. Sentence Generation Complexity

Grammar-based sentence generation is not an easy task. Although unification grammars such as Lexical-Functional Grammar (LFG) (Kaplan and Bresnan 1982), PATR-II (Shieber 1986), and Head-Driven Phrase Structure Grammar (Pollard and Sag 1987) are powerful formalisms for describing natural languages, the computational complexity of generation clearly reflects the descriptive power of such grammar formalisms.

[NP-hardness of Sentence Generation with Unification Grammars]
Let G be a unification grammar[1], and F be a feature structure. Then, it is *NP-hard* to decide whether or not there exists a derivation of a sentence *s* such that G associates F with the input *s*, written G(F) = *s*.

Proof is given by transforming the Hamiltonian Circuit (HC) problem (Garey and Johnson 1979), a known NP-complete problem, into the above problem as follows:

1. Let the instance of the HC be a graph H = (V,E). Let N be the number of nodes in V. The graph H is transformed into a unification grammar G[2].
2. For every vertex $v_i \in V$, G has a grammar rule

$X_{i,i,1} \rightarrow v_i$
$<X_{i,i,1}\ v_i> = \text{done}$

3. For every edge $e_{i,j} = <v_i,v_j> \in E$ between two vertices v_i and $v_j \in V$, G has N(N-1) grammar rules (k = 1, …, N-1 and m = 1, …, N)

$X_{m,j,k+1} \rightarrow X_{m,i,k}$
$<X_{m,j,k+1}> = <X_{m,i,k}>$
$<X_{m,j,k+1}\ v_j> = \text{done}$

and

$S \rightarrow X_{j,i,N}$
$<S> = <X_{j,i,N}>$

[1] For simplicity, we assume a PATR-II-type grammar, and borrow its notation for describing grammar rules. Our results can be easily applied to other types of unification grammars as long as the notions of derivation and derivation tree are well defined.
[2] G has N^3+1 nonterminal symbols, S (start symbol) and $X_{1,1,1}, \ldots, X_{N,N,N}$, and N terminal symbols $v_1, \ldots, v_N$.

It immediately follows that every valid derivation of G is a chain of one pre-terminal rule and N phrasal rules. This chain can bind "done" with at most N features. Therefore, if we give a feature structure F such that $<F\ v_i> =$ done for every v_i ($i = 1, \ldots, N$), H has a Hamiltonian circuit $v_k \rightarrow \ldots \rightarrow v_i \rightarrow v_j$ ($\rightarrow v_k$) iff $G(F) = v_j$ for some j and a derivational sequence $S \rightarrow X_{j,k,N} \rightarrow \ldots \rightarrow X_{j,i,2} \rightarrow X_{j,j,1} \rightarrow v_j$.

Note that this theorem complements the known NP-hard parsing complexity of LFG (Kaplan and Bresnan 1982). It is interesting to note that parsing and generation complexity is somewhat unrelated. That is, the parsing complexity of the above class of unification grammar G is polynomial since G accepts only a one-word sentence v_i such that v_i is *reachable* from S.

The complexity of such sentence generation does not improve even if we give derivational information additionally for constraining possible generations as long as the unification grammar is used for generation.

[NP-completeness of Sentence Generation from Partially Constrained Feature Structure]
Let G be a unification grammar, and D be a partial derivation tree such that only the context-free portions of the grammar rules used in the derivation are specified. Then, it is *NP-complete* to decide whether or not there exists a sentence *s* which is derived from G by D.

Proof: It is in NP since we can guess an assignment of |D| (the size of D) actual grammar rules to D, and determine if it makes a valid generation. The 3-Satisfiability (3SAT) problem can then be transformed into this problem. For every conjunct c_i ($i = 1, \ldots, N$) with three variables $v_{i,1}$, $v_{i,2}$, and $v_{i,3}$ in the instance of the 3SAT, G has three grammar rules ($j = 1, 2, 3$),

$X_{i+1} \rightarrow X_i$
$<X_{i+1}> = <X_i>$
$<X_{i+1}\ v_j> = 0$

if $v_{i,j}$ is negated in c_i, or otherwise

$X_{i+1} \rightarrow X_i$
$<X_{i+1}> = <X_i>$
$<X_{i+1}\ v_j> = 1$

G also has two more rules $S \rightarrow X_{N+1}$ and $X_1 \rightarrow$ true. Let D be a sequence of $S \rightarrow X_{N+1} \rightarrow \ldots X_1 \rightarrow$true. Then, 3SAT is satisfiable iff G can generate "true" with D.

The complexity of sentence generation will not improve even if we employ synchronized version of context-free grammars (CFGs)[3], that is, by bypassing feature structures and directly describing syntactic relationship between source and target languages. The NP-hardness result of SDTS has been shown by Satta (1992). These properties imply that grammar-based sentence appears to be intractable in the worst case. It is therefore very important to identify a restricted subclass of grammars or to impose a set of constraints on feature structures or synchronized grammars so that an efficient generation algorithm can be obtained. Approximation algorithms allowing under- and over- generations to some extent can also be an alternative for implementing fast generation algorithms.

For a "fixed" grammar, SDTS is known to be solvable in polynomial time. Vijay-Shanker et al. (1987) showed the result holds for linear context-free rewriting systems (LCFRS). If an underspecified feature structure (Kuhn 1996) is allowed, however, we can show that a fixed unification grammar generation problem to be NP-complete.

[3] It is known as a "syntax-directed translation scheme" (SDTS) (Aho and Ullman 1972).

[NP-completeness of Sentence Generation from Underspecified Feature Structures]
Let G be a unification grammar with the following 65 rules (Figure 1),

```
[Rule 1] S → TERMS
          <S> = <TERMS>
[Rule 2] TERMS → TERM
          <TERMS> = <TERM>
[Rule3] TERMS → TERM TERMS
          <TERMS> = <TERM>
          <TERMS index next> = <TERMS index>
          <TERMS value> = <TERMS value>
[Rule 4-59 (56 possible combinations of VARs (PVAR or NVAR) and satisfiability)
          TERM → IND_1 VAR_1 IND_2 VAR_2 IND_3 VAR_3
          <TERM type> = 111     (one of 000, 001, 010, …, 111)
          <TERM index var_1> = <IND_1>
          <TERM index var_2> = <IND_2>
          <TERM index var_3> = <IND_3>
          <TERM value> = <VAR_1 value>
          <TERM value> = <VAR_2 value>
          <TERM value> = <VAR_3 value>
          <IND_1 index> = <VAR_1 index>
          <IND_2 index> = <VAR_2 index>
          <IND_3 index> = <VAR_3 index>
[Rule 60] PVAR → PVAR
          <PVAR index index> = <PVAR index>
          <PVAR value value> = <PVAR value>
[Rule 61] PVAR → 1
          <PVAR index> = 1
          <PVAR value> = 1
[Rule 62] NVAR → NVAR
          <NVAR index index> = <NVAR index>
          <NVAR value value> = <NVAR value>
[Rule 63] NVAR→ 0
          <PVAR index> = 1
          <PVAR value> = 0
[Rule 64] IND → i
          <IND index> = 1
[Rule 65] IND → IND
          <IND index> = <IND>
```

Figure 1. Unification Grammar for 3SAT

and f be a feature structure of the form[4]

((index (var$_{11}$ 1) (var$_{12}$ 1) (var$_{13}$ 1)
    (next (var$_{21}$ 1) (var$_{22}$ 1) (var$_{23}$ 1)

[4] where (var$_{ij}$ 1) is a shorthand notation of a path (var_i (index (index …(index 1)…))) with length j and its value 1, (i=1,2,3 and j=1,2,…,3k), and (var$_p$ (0 or 1)) is a shorthand notation of a path (index (index …(index (value (0 or 1)))…))) with length p and its value (0 or 1), (p=1,2,…,3k).

…
(next (var_{k1} 1) (var_{k2} 1) (var_{k3} 1))…)
(value (var_1 (0 or 1)) (var_2 (0 or 1)) … (var_n (0 or 1))…))

such that a conjunction of two atomic values "(0 or 1)" is allowed. Then, it is *NP-complete* to decide whether or not there exists a sentence *s* which is derived from G by f.

Proof: It is in NP since we can guess an assignment of at most (6*3k+2)*k+1 actual grammar rules to see if f can generate a sentence. Now, it is easy to see that f can represent the instance of 3SAT problem, and since Rules 4-59 makes only valid truth assignment to variables, G generates s from f iff the 3SAT instance is satisfiable.

In the next section, we will show that pattern-based CFG can be used to describe lexicalized translation patterns as well as efficient sentence generation.

3. Sentence Generation for Pattern-based Grammars

A pattern-based CFG (Takeda 1996) G consists of a set of pairs of grammar rules $<r_L, r_R>$ such that r_L (called a *source* rule) is a rule for analyzing source language sentences while r_R (called a *target* rule) is a rule for generating target language sentences using phrasal correspondences encoded in the pair of rules. Pattern-based CFG allows that predefined set of features can be associated with the grammar rules.

For example, the synchronized grammar rule

$$<S_1 \rightarrow NP_2{:}{+}3SG \text{ knows } NP_3,\ S_1 \rightarrow NP_2{:}{+}3SG \text{ connait } NP_3>$$

specifies that the English sentence "X knows Y" is translated into French counterpart "X' connait Y", where X' (Y') is a French translation of the English noun phrase X (Y), and such phrasal correspondences, called *links*, are expressed by the indices above.

Generation using synchronized grammars can be traced back to the idea of "transducers" in the formal language and automata theory (Aho and Ullman 1972) and the Synchronous Tree-Adjoining Grammar (STAG) (Shieber and Schabes 1990) among others, has shown that it is a very promising tool for practical natural language applications including machine translation (Abeille and Schabes 1990). Pattern-based CFG is especially useful for specifying idiosynctratic translation patterns in terms of "root" features. For example,

$$<VP_1 \rightarrow \text{know}{:}VP_2\ NP_3,\ VP_1 \rightarrow \text{savoir}{:}VP_2\ NP_3> \text{ and}$$
$$< VP_1 \rightarrow \text{know}{:}VP_2 \text{ a little},\ VP_1 \rightarrow \text{savoir}{:}VP_2 \text{ un peu}>$$

can specify that two translation pairs (know/savoir and a little/un peu) can be easily specified without describing the entire "VP NP adverb" context. As stated in the previous section, there is a polynomial time algorithm for parsing/generation from a fixed context-free rewriting system.

[Polynomial Sentence Generation for a fixed Pattern-Based CFG]
Let G = $<r_L, r_R>$ be the pattern-based CFG, and K be the maximum number of nonterminal symbols in the right hand side of either r_L or r_R. Then, there is an Earley-style (Earley 1970) parser/generator for G, which runs in time $O(|G|^2 n^{2K+1})$. (|G| is the size of the grammar G, and *n* is the length of the input string.)

One way to improve the algorithm is to separate generation from parsing. We can then work on optimizing the parser for obtaining the best parse tree of an input possibly by using stochastic or heuristic knowledge. Once the parse tree T is given, we can instantiate the corresponding target rules in O(KT) time and generate the translation if the instantiations are successful.[5] The parser could be made as efficient as $O(|G|n^3)$-time.

Other improvements of the algorithm include:[6]

1. restricting the number of completion procedure calls for each portion of the input string.
2. pre-compiling the grammar to bundle the rules with the same source rules.
3. defining the partial ordering of grammar rules, $r_i < r_j$ such that as long as r_i is initialized and instantiated, every r_j is ignored.

4. Experimental Result

We have implemented the parsing/generation algorithm of a pattern-based CFG for English-to-Japanese translation and obtained the result shown in Table 1 for a collection of 1,717 sample English sentences including an MT evaluation corpus (JEIDA 1995) developed by JEIDA (the Japan Electronic Industry Development Association). Our experimental synchronized grammar consists of about 2,000 hand-crafted phrasal rules and about 30,000 lexicalized rules collected by using online resources (with human intervention).

The maximum number of nonterminal symbols in a rule was 9. The sample sentences were collected from textbooks, newspaper articles, and Internet homepages. Less than 100 sentences of them resulted in either timeout or unsuccessful parsing/generation.
The sample program was written in C++, and was run on a UNIX workstation (RS/6000 model F50). We introduced *number* and *person* features into nouns and *infinite*, *finite*, *past participle*, and *present participle* features into verbs so that we can specify subject-verb agreement constraints and several verb inflectional constraints in the pure phrasal rules. This modification was necessary to obtain correct parsing/generation result without introducing too many nonterminal symbols, and just adds negligible overhead to the algorithm.

Max inactive charts	Sec/sentence	Words/sec	Difference (better/worse)	# of timeout (5 sec)
4	0.70	17.6	-	11
5	0.75	16.5	50 (10/4)	14
6	0.81	15.2	53 (13/11)	22

Table 1. Performance Evaluation of Synchronized CFG Parsing/Generation

The techniques incorporated in the algorithm were as follows:

1. Each rule is associated with a cost so that we can define a cost for each *inactive chart* (i.e., a fully instantiated source rule) as a sum of all rules constituting the chart. A rule is associated with a smaller cost if it has more terminal symbols.
2. The maximum number of inactive charts for each portion of input string is limited by a constant K. That is, the charts with up to K-th minimum cost are maintained.

[5] Otherwise, we get no translation at all. In this sense, this generation algorithm is not *sound* even if we allow the K-best parse trees as input for any constant K.
[6] They may not improve the worst-case complexity. Our experiments displayed, however, that average runtime of the parser/generator has been drastically improved by these techniques.

3. The maximum number of partially instantiated rules for each portion of input string is also limited by 200.

```
[(0 21) S -> *S1 PUNCT [S -> *S PUNCT]
 [(0 20) S1 -> *S1 SADJ [S -> S *S]
  [(0 11) S1 ->  NP *VP (agreement check)
                                [VP -> NP *VP]
   [(0 4) NP -> "the" *NP [NP -> *NP]
    [(1 4) NP -> *NOUN [NP -> *NOUN]
     [(1 4) NOUN -> "File" "Transfer" *"Protocol" [ファイル転送プロトコル]]]]
   [(4 11) VP -> *V1 [VP -> *VP]
    [(4 11) V1 -> *"be" NP [VP -> NP *COP]
     [(5 11) NP -> *NP "to" VP (+infinitive check) [NP -> VP *NP]
      [(5 8) NP -> "a" *NP [NP -> *NP]
       [(6 8) NP -> ADJP *NP [NP -> AP *NP]
        [(6 7) ADJP -> "standardized" [標準化された]]
        [(7 8) NP -> *NOUN [NP -> *NOUN]
         [(7 8) NOUN -> *"way"]]]]
      [(9 11) VP -> *VP NP [VP -> NP *VP]
       [(9 10) VP -> *V1 [VP -> *VP]
        [(9 10) V1 -> *VERB [VP -> *VERB]
         [(9 10) VERB -> *"connect" [結ぶ]]]]
       [(10 11) NP -> *NOUN [NP -> *NOUN]
        [(10 11) NOUN -> *"computer" [コンピュータ]]]]]]]
  [(11 20) SADJ -> CONJ *S1 [S -> *S CONJ]
   [(11 13) CONJ -> "so" *"that" [ように]]
   [(13 20) S1 ->  NP *VP (agreement check) [S -> NP *VP]
    [(13 14) NP -> *NOUN [NP -> *NOUN]
     [(13 14) NOUN -> *"file" [ファイル]]
    [(14 20) VP -> can *VP [VP -> *VP]
     [(15 20) VP -> *V1 [VP -> *VP]
      [(15 20) V1 -> "be" *VP (+past_participle check) [VP -> *VP]
       [(16 20) VP -> *VP ADVP [VP -> ADVP *VP]
        [(16 19) VP -> *VP PP [VP -> PP *VP]
         [(16 17) VP -> *V1 [VP -> *VP]
          [(16 17) V1 -> *VERB [VP -> *VERB]
           [(16 17) VERB -> *"share" [共有する]]]]
         [(17 19) PP -> between *NP [PP -> *NP]
          [(18 19) NP -> *"them" [彼(それ)ら]]]
        [(19 20) ADVP -> *ADV [ADVP -> *ADV]
         [(19 20) ADV -> *"easily" [容易に]]]]]]]]
 [(20 21) PUNCT -> *"."]]
```

The asterisk(*) preceding the terminal and nonterminal symbols shows that they are head daughters of a rule. Each line in the figure shows an inactive chart [(i j) *source rule*] [*target rule*]], where i and j are the positions of words in the input sentence. The to-infinitive “to connect ...” in the parse tree should have been analyzed as modifying the noun phrase “a standardized way”, but the Japanese translation is understandable with no semantic misleading (transliteration of Japanese words are omitted).

Figure 2. Sample Run of Synchronous Generation

The Table 1 mainly shows the effect of the constant K mentioned above. Translation (parsing and generation) of a sentence is obtained in 0.7 to 0.8 second average time, which is equivalently 15 to 17 words per second. The constant K (= 4,5,6) makes little difference[7] in the resulting translations, but it is evident that the algorithm sharply slows down for longer sentences as the number of timeout happens much more frequently when K increases. The ratio of parsing/generation elapsed time was about 50:1 in all cases. It is interesting to note that the pattern-based MT has no explicit “transfer” process. It is rather folded into the generation process, where conversion from source to target representations is immediately obtained from the patterns. It then follows that analysis process has been multiplied by average number of patterns with the same source rule and alternative target rules associated with it. It also explains why the algorithm had rather large number of timeout sentences when the constant K grows.

We can obtain two important observations from the experiments:

1. This algorithm can be incorporated into practical applications with serious speed requirements.
2. Reasonable quality of parsing/generation is preserved for amazingly small K. Human reviewers evaluated approximately 75% of translated sentences as either correct or easily understandable[8], which is comparable to the performance of the current commercial English-Japanese MT systems.

Figure 1 shows a synchronous translation of one of the sample sentences,

The File Transfer Protocol is a standardized way to connect computers so that files can be shared between them easily.

JAPANESE TRANSLATION:

ファイルが容易に彼(それ)らの間で共有することができるように，ファイル転送プロトコルは，コンピュータを結ぶ標準化された方法です.

(to make the files easily shared among them, file transfer protocol is a standardized method for connecting computers.)

5. Conclusion and Future Work

In this paper, we have shown computational complexity of sentence generation. Theoretical worst-case analysis suggests that grammar-based generation in general could be intractable, but a polynomial-time generation algorithm for a “fixed” pattern-based CFG exists, and experimental implementation of the algorithm, with additional acceleration techniques, showed an encouraging result of its efficiency and practical durability. Corpus-based approach should be easily integrated with the pattern-based generation algorithm, and make up a robust and scalable MT systems.

[7] Only about 50 sentences were resulted in different translations, and as many as 13 sentences actually made better/worse translations.
[8] possibly with incorrect use of function words or improper phrase ordering

Future work includes verification of similar techniques for generation with *synchronized unification grammars*, and comparison of generation algorithms using different CFG-equivalent grammar formalisms for practical situations.

For example, lexicalization plays a key role in polynomial-time *tree-to-forest* translation (Rambow and Satta 1996), infinite hierarchy of pushdown transducers (Aho and Ullman 1972) is characterized by a maximum number of nonterminal symbols allowed in target rules of *simple* synchronized CFG. These aspects might lead us to find a very practical class of synchronized grammars.

Since our experiments show that parsing takes more than 98% of the entire runtime for translation using synchronous grammars, powerful disambiguation techniques for parsing should be the most critical issues for successful translation algorithms. It is an open problem to find a class of instances where generation takes a significant amount of time.

References

A. Abeille Y. Schabes, and A. K. Joshi: 1990, “Using Lexicalized Tags for Machine Translation”, *in Proc. of the 13th International Conference on Computational Linguistics*, pp.1-6.

A. V. Aho and J. D. Ullman.: 1972, *The Theory of Parsing, Translation, and Compiling*, Volume 1: Parsing, Prentice-Hall, Englewood Cliffs, New Jersey.

R. C. Berwick.: 1982, “Computational Complexity and Lexical-Functional Grammar”, *American Journal of Computational Linguistics*, Vol.8, No.3, pp.97-109.

E. Charniak:1997, “Statistical Techniques for Naural Language Parsing”, *AI magazine*, Vol.18, No.4, pp.33-43.

M. Dymetman:1991, “Inherently Reversible Grammars, Logic Programming and Computability”, *in Proc. of the ACL Workshop on Reversible Grammar in Natural Language Processing*, pp.20-30, Berkeley, California.

J. Earley: 1970, “An Efficient Context-free Parsing Algorithm”, *Communications of the ACM*, Vol.6, No.8, pp.94-102.

M. Garey and D. Johnson: 1979, *Computers and Intractability*, W.H.Freeman and Co., San Francisco.

JEIDA: 1995, Evaluation Standards for Machine Translation Systems (in Japanese), Technical Report, 95-COMP-17, Japan Electronic Industry Development Association.

R. Kaplan and J. Bresnan: 1982, “Lexical-Functional Grammar: A Formal System for Generalized Grammatical Representation”, in J. Bresnan, editor, *Mental Representation of Grammatical Relations*, pp.173-281. MIT Press, Cambridge, Mass.

R. Kaplan and J. Wedekind: 2000, “LFG Generation Produces Context-free Languages”*, in Proc of the 18th International Conference on Computational Linguistics*, pp.425-431

L. Karttunen: 1994, “Constructing Lexical Transducers”, *in Proc. of the 15th International Conference on Computational Linguistics*, pp.406-411.

K. Knight: 1997, “Automatic Knowledge Acquisition for Machine Translation”, *AI magazine*, Vol.18, No.4, pp.81-96.

J. Kuhn: 1996, “An Underspecified HPSG Representation for Information Structure”, *in Proc. of the 15th International Conference on Computational Linguistics*, pp.670-675.

M. Martinovic and T. Strzalkowski:1992, “Comparing Two Grammar-Based Generation Algorithms: A Case Study*", in Proc. of the 30th Annual Meeting of ACL*, pp.81-88.

C. Pollard and I. A. Sag: 1987, *An Information-Based Syntax and Semantics*, Vol.1 Fundamentals, CSLI Lecture Notes, Number 13.

O. Rambow and G. Satta:1996, "Synchronous Models of Language". *in Proc. of the 34th Annual Meeting of the Association for Computational Linguistics*, pp.116-123.

S. Sato and M. Nagao: 1990, "Toward Memory-based Translation", *in Proc. of the 13th International Conference on Computational Linguistics*, pp.247-252.

G. Satta: 1992, “Recognition of Linear Context-Free Rewriting Systems”, *in Proc. of the 30th Annual Meeting of the Association for Computational Linguistics*, pp.89-95.

Y. Schabes and R. C. Waters: 1995, "Tree Insertion Grammar: A Cubic-Time, Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced", *Computational Linguistics*, Vol.21, No.4, pp.479-513

S. M. Shieber: 1986, *An Introduction to Unification-Based Approaches to Grammar*, CSLI Lecture Notes, Number 4, Stanford, CA.

S. M. Shieber, F. C. N. Pereira, G. van Noord, and R. C. Moore: 1990, “Semantic-Head-Driven Generation”, *Computational Linguistics*, Vol. 16, No.1, pp.30-42

S. M. Shieber and Y. Schabes, 1990, "Synchronous Tree-Adjoining Grammars". *in Proc. of the 13th International Conference on Computational Linguistics*, pp.253-258.

S. M. Shieber: 1994, Restricting the Weak Generative Capacity of Synchronous Tree Adjoining Grammar, *Computational Intelligence*, Vol.10, No.4, pp.371-385

H. Somers: 1999, “Review Article: Example-based Machine Translation”, *Machine Translation*, Vol.14, pp.113-157.

K. Takeda: 1996, "Pattern-Based Context-Free Grammars for Machine Translation", *in Proc. of the 34th Annual Meeting of ACL*, pp.144-151

K. Vijay-Shanker, D. J. Weir, and A. K. Joshi: 1987, “Characterizing Structural Descriptions Produced by Various Grammatical Formalisms”, *in 25th Annual Meeting of ACL*, pp.104-111

H. Watanabe and K. Takeda: 1998, “A Pattern-Based Machine Translation System Extended by Example-Based Processing”, in Proc. of the 17th International Conference on Computational Linguistics, pp.1369-1373.

J. Wedekind: 1988, "Generation as Structure Driven Derivation", *in Proc. of the 12th International Conference on Computational Linguistics*, pp.732-737.

G. Wilcock, and Y. Matsumoto: 1998, “Head-Driven Generation with HPSG”, *in Proc. of the 17th International Conference on Computational Linguistics*, pp.1393-1397

D. Wu: 1997, "Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora", *Computational Linguistics*, Vol.23, No.3, pp.377-403.