Pattern-Based Context-Free Grammars for Machine Translation

Koichi Takeda
Tokyo Research Laboratory, IBM Research
1623-14 Shimotsuruma, Yamato, Kanagawa 242, Japan
Phone: 81-462-73-4569, 81-462-73-7413 (FAX)
takeda@trl.vnet.ibm.com

Abstract

This paper proposes the use of "pattern-based" context-free grammars as a basis for building machine translation (MT) systems, which are now being adopted as personal tools by a broad range of users in the cyberspace society. We discuss major requirements for such tools, including easy customization for diverse domains, the efficiency of the translation algorithm, and scalability (incremental improvement in translation quality through user interaction), and describe how our approach meets these requirements.

1 Introduction

With the explosive growth of the World-Wide Web (WWW) as information source, it has become routine for Internet users to access textual data written in foreign languages. In Japan, for example, a dozen or so inexpensive MT tools have recently been put on the market to help PC users understand English text in WWW home pages. The MT techniques employed in the tools, however, are fairly conventional. For reasons of affordability, their designers appear to have made no attempt to tackle the well-known problems in MT, such as how to ensure the learnability of correct translations and facilitate customization. As a result, users are forced to see the same kinds of translation errors over and over again, except they in cases where they involve merely adding a missing word or compound to a user dictionary, or specifying one of several word-to-word translations as a correct choice.

There are several alternative approaches that might eventually liberate us from this limitation on the usability of MT systems:

Unification-based grammar formalisms and lexical-semantics formalisms (see LFG (Kaplan and Bresnan, 1982), HPSG (Pollard and Sag, 1987), and Generative Lexicon (Pustejovsky, 1991), for example) have been proposed to facilitate computationally precise description of natural-language syntax and semantics. It is possible that, with the descriptive power of these grammars and lexicons, individual usages of words and phrases may be defined specifically enough to give correct translations. Practical implementation of MT systems based on these formalisms, on the other hand, would not be possible without much more efficient parsing and disambiguation algorithms for these formalisms and a method for building a lexicon that is easy even for novices to use.

Corpus-based or example-based MT (Sato and Nagao, 1990; Sumita and Iida, 1991) and statistical MT (Brown et al., 1993) systems provide the easiest customizability, since users have only to supply a collection of source and target sentence pairs (a bilingual corpus). Two open questions, however, have yet to be satisfactorily answered before we can confidently build commercial MT systems based on these approaches:

- Can the system be used for various domains without showing severe degradation of translation accuracy?
- What is the minimum number of examples (or training data) required to achieve reasonable MT quality for a new domain?

TAG-based MT (Abeillé, Schabes, and Joshi, 1990)[1] and pattern-based translation (Maruyama, 1993) share many important properties for successful implementation in practical MT systems, namely:

- The existence of a polynomial-time parsing algorithm
- A capability for describing a larger *domain of locality* (Schabes, Abeillé, and Joshi, 1988)
- *Synchronization* (Shieber and Schabes, 1990) of the source and target language structures

Readers should note, however, that the pars-

[1] See LTAG (Schabes, Abeillé, and Joshi, 1988) (Lexicalized TAG) and STAG (Shieber and Schabes, 1990) (Synchronized TAG) for each member of the TAG (Tree Adjoining Grammar) family.

ing algorithm for TAGs has $O(|G|n^6)$[2] worst case time complexity (Vijay-Shanker, 1987), and that the "patterns" in Maruyama's approach are merely context-free grammar (CFG) rules. Thus, it has been a challenge to find a framework in which we can enjoy both a grammar formalism with better descriptive power than CFG and more efficient parsing/generation algorithms than those of TAGs.[3]

In this paper, we will show that there exists a class of "pattern-based" grammars that is weakly equivalent to CFG (thus allowing the CFG parsing algorithms to be used for our grammars), but that it facilitates description of the domain of locality. Furthermore, we will show that our framework can be extended to incorporate example-based MT and a powerful learning mechanism.

2 Pattern-Based Context-Free Grammars

Pattern-based context-free grammars (PCFG) consists of a set of translation *patterns*. A pattern is a pair of CFG rules, and zero or more *syntactic head* and *link* constraints for nonterminal symbols. For example, the English-French translation pattern[4]

NP:1 miss:V:2 NP:3 → S:2
S:2 ← NP:3 manquer:V:2 *à* NP:1

essentially describes a *synchronized*[5] pair consisting of a left-hand-side English CFG rule (called a *source* rule)

NP V NP → S

and a French CFG rule (called a *target* rule)

S ← NP V *à* NP

accompanied by the following constraints.

1. **Head constraints:** The nonterminal symbol V in the source rule must have the verb *miss* as a syntactic head. The symbol V in the target rule must have the verb *manquer* as a syntactic head. The head of symbol S in the source (target) rule is identical to the head of symbol V in the source (target) rule as they are co-indexed.
2. **Link constraints:** Nonterminal symbols in source and target CFG rules are *linked* if they are given the same index ":*i*". Linked nonterminal must be derived from a sequence of synchronized pairs. Thus, the first NP (NP:1) in the source rule corresponds to the second NP (NP:1) in the target rule, the Vs in both rules correspond to each other, and the second NP (NP:3) in the source rule corresponds to the first NP (NP:3) in the target rule.

The source and target rules are called *CFG skeleton* of the pattern. The notion of a syntactic head is similar to that used in unification grammars, although the heads in our patterns are simply encoded as character strings rather than as complex feature structures. A head is typically introduced[6] in preterminal rules such as

leave → V V ← partir

where two verbs, "leave" and "partir," are associated with the heads of the nonterminal symbol V. This is equivalently expressed as

leave:1 → V:1 V:1 ← partir:1

which is physically implemented as an entry of an English-French lexicon.

A set T of translation patterns is said to *accept* an input s iff there is a derivation sequence Q for s using the source CFG skeletons of T, and every head constraint associated with the CFG skeletons in Q is satisfied. Similarly, T is said to *translate* s iff there is a synchronized derivation sequence Q for s such that T accepts s, and every head and link constraint associated with the source and target CFG skeletons in Q is satisfied. The derivation Q then produces a translation t as the resulting sequence of terminal symbols included in the target CFG skeletons in Q. Translation of an input string s essentially consists of the following three steps:

1. Parsing s by using the source CFG skeletons
2. Propagating link constraints from source to target CFG skeletons to build a target CFG derivation sequence
3. Generating t from the target CFG derivation sequence

The third step is a trivial procedure when the target CFG derivation is obtained.

Theorem 1 *Let T be a PCFG. Then, there exists a CFG G_T such that for two languages $L(T)$ and $L(G_T)$ accepted by T and G_T, respectively, $L(T) = L(G_T)$ holds. That is, T accepts a sentence s iff G_T accepts s.*

Proof: We can construct a CFG G_T as follows:

1. G_T has the same set of terminal symbols as T.

[2] Where $|G|$ stands for the size of grammar G, and n is the length of an input string.

[3] Lexicalized CFG, or Tree Insertion Grammar (TIG) (Schabes and Waters, 1995), has been recently introduced to achieve such efficiency and lexicalization.

[4] and its inflectional variants — we will discuss inflections and agreement issues later.

[5] The meaning of the word "synchronized" here is exactly the same as in STAG (Shieber and Schabes, 1990). See also bilingual signs (Tsujii and Fujita, 1991) for a discussion of the importance of combining the appropriate domain of locality and synchronization.

[6] A nonterminal symbol X in a source or target CFG rule $X \rightarrow X_1 \cdots X_k$ can only be constrained to have one of the heads in the RHS $X_1 \cdots X_k$. Thus, *monotonicity* of head constraints holds throughout the parsing process.

2. For each nonterminal symbol X in T, G_T includes a set of nonterminal symbols $\{X_w | w$ is either a terminal symbol in T or a special symbol $\epsilon\}$.

3. For each preterminal rule

   X:i $\rightarrow w_1$:1 w_2:2 ... w_k:k $(1 \leq i \leq k)$,

   G_T includes[7]

   $Xw_i \rightarrow w_1\ w_2\ \ldots\ w_k\ (1 \leq i \leq k)$.

   If X is not co-indexed with any of w_i, G_T includes

   $X_\epsilon \rightarrow w_1\ w_2\ \ldots\ w_k$.

4. For each source CFG rule with head constraints $(h_1, h_2, \ldots, h_k)$ and indexes $(i_1, i_2, \ldots, i_k)$,

   $Y{:}i_j \rightarrow h_1{:}X_1{:}i_1 \ldots h_k{:}X_k{:}i_k\ (1 \leq j \leq k)$,

   G_T includes

   $Yh_j \rightarrow Xh_1\ Xh_2\ \ldots\ Xh_k$.

   If Y is not co-indexed with any of its children, we have

   $Y_\epsilon \rightarrow Xh_1\ Xh_2\ \ldots\ Xh_k$.

   If X_j has no head constraint in the above rule, G_T includes a set of $(N+1)$ rules, where Xh_j above is replaced with X_w for every terminal symbol w and X_ϵ (Yh_j will also be replaced if it is co-indexed with X_j).[8]

Now, $L(T) \subseteq L(G_T)$ is obvious, since G_T can simulate the derivation sequence in T with corresponding rules in G_T. $L(G_T) \subseteq L(T)$ can be proven, with mathematical induction, from the fact that every valid derivation sequence of G_T satisfies head constraints of corresponding rules in T.

□

Proposition 1 *Let a CFG G be a set of source CFG skeletons in T. Then, $L(T) \subseteq L(G)$.*

Since a valid derivation sequence in T is always a valid derivation sequence in G, the proof is immediate. Similarly, we have

Proposition 2 *Let a CFG H be a subset of source CFG skeletons in T such that a source CFG skeleton k is in H iff k has no head constraints associated with it. Then, $L(H) \subseteq L(T)$.*

[7]Head constraints are trivially satisfied or violated in preterminal rules. Hence, we assume, without loss of generality, that no head constraint is given in preterminal rules. We also assume that "$X \rightarrow w$" implies "X:1 $\rightarrow w$:1".

[8]Therefore, a single rule in T can be mapped to as many as $(N+1)^k$ rules in G_T, where N is the number of terminal symbols in T. G_T could be exponentially larger than T.

Two CFGs G and H define the range of CFL L(T). These two CFGs can be used to measure the "default" translation quality, since idioms and collocational phrases are typically translated by patterns with head constraints.

Theorem 2 *Let a CFG G be a set of source CFG skeletons in T. Then, $L(T) \subset L(G)$ is undecidable.*

Proof: The decision problem, $L(T) \subset L(G)$, of two CFLs such that $L(T) \subseteq L(G)$ is solvable iff $L(T) = L(G)$ is solvable. This includes a known undecidable problem, $L(T) = \Sigma^*$?, since we can choose a grammar U with $L(U) = \Sigma^*$, nullify the entire set of rules in U by defining T to be a vacuous set $\{S{:}1 \rightarrow a{:}S_b{:}1,\ S_b{:}1 \rightarrow b{:}S_U{:}1\} \cup$ U (S_U and S are start symbols in U and T, respectively), and, finally, let T further include an arbitrary CFG F. $L(G) = \Sigma^*$ is obvious, since G has $\{S \rightarrow S_b,\ S_b \rightarrow S_U\} \cup$ U. Now, we have $L(G) = L(T)$ iff $L(F) = \Sigma^*$.

□

Theorem 2 shows that the syntactic coverage of T is, in general, only computable by T itself, even though T is merely a CFL. This may pose a serious problem when a grammar writer wishes to know if there is a specific expression that is only acceptable by using at least one pattern with head constraints, for which the answer is "no" iff L(G) = L(T). One way to trivialize this problem is to let T include a pattern with a pair of pure CFG rules for every pattern with head constraints, which guarantees that L(H) = L(T) = L(G). In this case, we know that the coverage of "default" patterns is always identical to L(T).

Although our "patterns" have no more theoretical descriptive power than CFG, they can provide considerably better descriptions of the domain of locality than ordinary CFG rules. For example,

be:V:1 year:NP:2 old $\rightarrow$ VP:1
VP:1 $\leftarrow$ avoir:V:1 an:NP:2

can handle such NP pairs as "one year" and "un an," and "more than two years" and "plus que deux ans," which would have to be covered by a large number of plain CFG rules. TAGs, on the other hand, are known to be "mildly context-sensitive" grammars, and they can capture a broader range of syntactic dependencies, such as cross-serial dependencies. The computational complexity of parsing for TAGs, however, is $O(|G|n^6)$, which is far greater than that of CFG parsing. Moreover, defining a new STAG rule is not as easy for the users as just adding an entry into a dictionary, because each STAG rule has to be specified as a pair of tree structures. Our patterns, on the other hand, concentrate on specifying linear ordering of source and target constituents, and can be written by the users as easily as[9]

[9]By sacrificing linguistic accuracy for the description of syntactic structures.

to leave * = de quitter *
to be year:* old = d'avoir an:*

Here, the wildcard "*" stands for an NP by default. The preposition "to" and "de" are used to specify that the patterns are for VP pairs, and "to be" is used to show that the phrase is the BE-verb and its complement. A wildcard can be constrained with a head, as in "house:*" and "maison:*". The internal representations of these patterns are as follows:

leave:V:1 NP:2 → VP:1
VP:1 ← quitter:V:1 NP:2

be:V:1 year:NP:2 old → VP:1
VP:1 ← avoir:V:1 an:NP:2

These patterns can be associated with an explicit nonterminal symbol such as "V:*" or "ADJP:*" in addition to head constraints (e.g., "leave:V:*"). By defining a few such notations, these patterns can be successfully converted into the formal representations defined in this section. Many of the divergences (Dorr, 1993) in source and target language expressions are fairly collocational, and can be appropriately handled by using our patterns. Note the simplicity that results from using a notation in which users only have to specify the surface ordering of words and phrases. More powerful grammar formalisms would generally require either a structural description or complex feature structures.

3 The Translation Algorithm

The parsing algorithm for translation patterns can be any of known CFG parsing algorithms including CKY and Earley algorithms[10] At this stage, head and link constraints are ignored. It is easy to show that the number of target charts for a single source chart increases exponentially if we build target charts simultaneously with source charts. For example, the two patterns

A:1 B:2 → B:2 B:2 ← A:1 B:2, and
A:1 B:2 → B:2 A:1 ← B:2 A:1

will generate the following 2^n synchronized pairs of charts for the sequence of (n+1) nonterminal symbols $AAA \ldots AB$, for which no effective packing of the target charts is possible.

(A (A ... (A B))) with (A (A ... (A B)))
(A (A ... (A B))) with ((A ... (A B)) A)
...
(A (A ... (A B))) with (((B A) A) ... A)

Our strategy is thus to find a candidate set of source charts in polynomial time. We therefore apply heuristic measurements to identify the most promising patterns for generating translations. In this sense, the entire translation algorithm is not guaranteed to run in polynomial time. Practically, a timeout mechanism and a process for recovery from unsuccessful translation (e.g., applying the idea of fitted parse (Jensen and Heidorn, 1983) to target CFG rules) should be incorporated into the translation algorithm.

Some restrictions on patterns must be imposed to avoid infinitely many ambiguities and arbitrarily long translations. The following patterns are therefore not allowed:

1. $A \rightarrow X\ Y \leftarrow B$
2. $A \rightarrow X\ Y \leftarrow C_1 \ldots B \ldots C_k$

if there is a cycle of synchronized derivation such that

$A \rightarrow X \ldots \rightarrow A$ and
B (or $C_1 \ldots B \ldots C_k$) $\rightarrow Y \ldots \rightarrow B$,

where A, B, X, and Y are nonterminal symbols with or without head and link constraints, and C's are either terminal or nonterminal symbols.

The basic strategy for choosing a candidate derivation sequence from ambiguous parses is as follows.[11] A simplified view of the Earley algorithm (Earley, 1970) consists of three major components, *predict(i)*, *complete(i)*, and *scan(i)*, which are called at each position $i = 0, 1, \ldots, n$ in an input string I = $s_1 s_2 \ldots s_n$. Predict(i) returns a set of currently applicable CFG rules at position i. Complete(i) combines inactive charts ending at i with active charts that look for the inactive charts at position i to produce a new collection of active and inactive charts. Scan(i) tries to combine inactive charts with the symbol s_{i+1} at position i. Complete(n) gives the set of possible parses for the input I.

Now, for every inactive chart associated with a nonterminal symbol X for a span of ⟨i,j⟩ ($1 \leq i, j \leq n$), there exists a set P of patterns with the source CFG skeleton, $\ldots \rightarrow X$. We can define the following ordering of patterns in P; this gives patterns with which we can use head and link constraints for building target charts and translations. These candidate patterns can be arranged and associated with the chart in the complete() procedure.

1. Prefer a pattern p with a source CFG skeleton $X \rightarrow X_1 \cdots X_k$ over any other pattern q with the same source CFG skeleton $X \rightarrow X_1 \cdots X_k$, such that p has a head constraint $h{:}X_i$ if q has $h{:}X_i$ ($i = 1, \ldots, k$). The pattern p is said to be *more specific* than q. For example, p =

[10] Our prototype implementation was based on the Earley algorithm, since this does not require lexicalization of CFG rules.

[11] This strategy is similar to that of transfer-driven MT (TDMT) (Furuse and Iida, 1994). TDMT, however, is based on a combination of declarative/procedural knowledge sources for MT, and no clear computational properties have been investigated.

"leave:V:1 house:NP → VP:1" is preferred to q = "leave:V:1 NP → VP:1".

2. Prefer a pattern p with a source CFG skeleton to any pattern q that has fewer terminal symbols in the source CFG skeleton than p. For example, prefer "take:V:1 a walk" to "take:V:1 NP" if these patterns give the VP charts with the same span.
3. Prefer a pattern p which does not violate any head constraint over those which violate a head constraint.
4. Prefer the shortest derivation sequence for each input substring. A pattern for a larger domain of locality tends to give a shorter derivation sequence.

These preferences can be expressed as numeric values (*cost*) for patterns.[12] Thus, our strategy favors *lexicalized* (or head constrained) and *collocational* patterns, which is exactly what we are going to achieve with pattern-based MT. Selection of patterns in the derivation sequence accompanies the construction of a target chart. Link constraints are propagated from source to target derivation trees. This is basically a bottom-up procedure.

Since the number M of distinct pairs ⟨X,w⟩, for a nonterminal symbol X and a subsequence w of input string s, is bounded by Kn^2, we can compute the *m-best choice* of pattern candidates for every inactive chart in time $O(|T|Kn^3)$ as claimed by Maruyama (Maruyama, 1993), and Schabes and Waters (Schabes and Waters, 1995). Here, K is the number of distinct nonterminal symbols in T, and n is the size of the input string. Note that the head constraints associated with the source CFG rules can be incorporated in the parsing algorithm, since the number of triples ⟨X,w,h⟩, where h is a head of X, is bounded by Kn^3. We can modify the predict(), complete(), and scan() procedures to run in $O(|T|Kn^4)$ while checking the source head constraints. Construction of the target charts, if possible, on the basis of the m best candidate patterns for each source chart takes $O(Kn^2m)$ time. Here, m can be larger than 2^n if we generate every possible translation.

The reader should note critical differences between lexicalized grammar rules (in the sense of LTAG and TIG) and translation patterns when they are used for MT.

Firstly, a pattern is not necessarily lexicalized. An economical way of organizing translation patterns is to include non-lexicalized patterns as "default" translation rules.

[12] A similar preference can be defined for the target part of each pattern, but we found many counter-examples, where the number of nonterminal symbols shows no specificity of the patterns, in the target part of English-to-Japanese translation patterns. Therefore, only the head constraint violation in the target part is accounted for in our prototype.

Secondly, lexicalization might increase the size of STAG grammars (in particular, compositional grammar rules such as ADJP NP → NP) considerably when a large number of phrasal variations (adjectives, verbs in present participle form, various numeric expressions, and so on) multiplied by the number of their translations, are associated with the ADJP part. The notion of structure sharing (Vijay-Shanker and Schabes, 1992) may have to be extended from lexical to phrasal structures, as well as from monolingual to bilingual structures.

Thirdly, a translation pattern can omit the tree structure of a collocation, and leave it as just a sequence of terminal symbols. The simplicity of this helps users to add patterns easily, although precise description of syntactic dependencies is lost.

4 Features and Agreements

Translation patterns can be enhanced with unification and feature structures to give patterns additional power for describing gender, number, agreement, and so on. Since the descriptive power of unification-based grammars is considerably greater than that of CFG (Berwick, 1982), feature structures have to be restricted to maintain the efficiency of parsing and generation algorithms. Shieber and Schabes briefly discuss the issue (Shieber and Schabes, 1990). We can also extend translation patterns as follows:

> Each nonterminal node in a pattern can be associated with a fixed-length *vector* of *binary features.*

This will enable us to specify such syntactic dependencies as agreement and subcategorization in patterns. Unification of binary features, however, is much simpler: unification of a feature-value pair succeeds only when the pair is either ⟨0,0⟩ or ⟨1,1⟩. Since the feature vector has a fixed length, unification of two feature vectors is performed in a constant time. For example, the patterns[13]

V:1:+TRANS NP:2 → VP:1 VP:1 ← V:1:+TRANS NP:2
V:1:+INTRANS → VP:1 VP:1 ← V:1:+INTRANS

are unifiable with transitive and intransitive verbs, respectively. We can also distinguish *local* and *head* features, as postulated in HPSG. Simplified version of verb subcategorization is then encoded as

VP:1:+TRANS-OBJ NP:2 → VP:1:+OBJ
VP:1:+OBJ ← VP:1:+TRANS-OBJ NP:2

where "-OBJ" is a local feature for head VPs in LHSs, while "+OBJ" is a local feature for VPs in

[13] Again, these patterns can be mapped to a weakly equivalent set of CFG rules. See GPSG (Gazdar, Pullum, and Sag, 1985) for more details.

the RHSs. Unification of a local feature with +OBJ succeeds since it is not *bound*.

Agreement on subjects (nominative NPs) and finite-form verbs (VPs, excluding the BE verb) is disjunctively specified as

```
NP:1:+NOMI+3RD+SG VP:2:+FIN+3SG
NP:1:+NOMI+3RD+PL VP:2:+FIN-3SG
NP:1:+NOMI-3RD    VP:2:+FIN-3SG
NP:1:+NOMI        VP:2:+FIN+PAST
```

which is collectively expressed as

```
NP:1:*AGRS VP:2:*AGRV
```

Here, *AGRS and *AGRV are a pair of *aggregate* unification specifiers that succeeds only when one of the above combinations of the feature values is unifiable.

Another way to extend our grammar formalism is to associate weights with patterns. It is then possible to rank the matching patterns according to a linear ordering of the weights rather than the pairwise partial ordering of patterns described in the previous section. In our prototype system, each pattern has its original weight, and according to the preference measurement described in the previous section, a penalty is added to the weight to give the effective weight of the pattern in a particular context. Patterns with the least weight are to be chosen as the most preferred patterns.

Numeric weights for patterns are extremely useful as means of assigning higher priorities uniformly to user-defined patterns. Statistical training of patterns can also be incorporated to calculate such weights systematically (Fujisaki et al., 1989).

Figure 1 shows a sample translation of the input "He knows me well," using the following patterns.

NP:1:*AGRS VP:1:*AGRS → S:1
S:1 ← NP:1:*AGRS VP:1:*AGRS ... (a)

VP:1 ADVP:2 → VP:1
VP:1 ← VP:1 ADVP:2 ... (b)

know:VP:1:+OBJ well → VP:1
VP:1 ← connaitre:VP:1:+OBJ bien ... (c)

V:1 NP:2 → VP:1:+OBJ
VP:1:+OBJ ← V:1 NP:2:-PRO ... (d)

V:1 NP:2 → VP:1:+OBJ
VP:1:+OBJ ← NP:2:+PRO V:1 ... (e)

To simplify the example, let us assume that we have the following preterminal rules:

he → NP:+PRO+NOMI+3RD+SG
NP:+PRO+NOMI+3RD+SG ← il ... (f)

me → NP:+PRO+CAUS+SG-3RD
NP:+PRO+CAUS+SG-3RD ← me ... (g)

knows → V:+FIN+3SG
V:+FIN+3SG ← sait ... (h)
knows → V:+FIN+3SG
V:+FIN+3SG ← connait ... (i)

```
Input: He knows me well
Phase 1: Source Analysis
[0 1] He    ---> (f) NP
   (active arc [0 1] (a) NP.VP)
[1 2] knows ---> (h) V, (i) V
   (active arcs [1 2] (d) V.NP,
                [1 2] (e) V.NP)
[2 3] me    ---> (g) NP
   (inactive arcs [1 3] (d) V NP,
                  [1 3] (e) V NP)
[1 3] knows me ---> (d), (e) VP
   (inactive arc [0 3] (a) NP VP,
    active arcs [1 3] (b) VP.well,
                [1 3] (c) VP.ADVP)
[0 3] He knows me ---> (a) S
[3 4] well  ---> (j) ADVP, (k) ADVP
   (inactive arcs [1 4] (b) VP ADVP,
                  [1 4] (c) VP ADVP)
[1 4] knows me well ---> (b), (c) VP
   (inactive arc [0 4] (a) NP VP)
[0 4] He knows me well ---> (a) S
Phase 2: Constraint Checking
[0 1] He    ---> (f) NP
[1 2] knows ---> (i) V, (j) V
[2 3] me    ---> (g) NP
[1 3] knows me ---> (e) VP
   (pattern (d) fails)
[0 3] He knows me ---> (a) S
[3 4] well  ---> (i) ADVP, (j) ADVP
[1 4] knows me well ---> (b), (c) VP
   (preference ordering (c), (b))
[0 4] He knows me well ---> (a) S
Phase 3: Target Generation
[0 4] He knows me well ---> (a) S
[0 1] He    ---> il
[1 4] knows me well ---> (c) VP
      well  ---> bien
[1 3] knows me ---> (e) VP
[1 2] knows ---> connait
      (h) violates a head constraint
[2 3] me    ---> me
Translation: il me connait bien
```

Figure 1: Sample Translation

well → ADVP ADVP ← bien ... (j)
well → ADVP ADVP ← beaucoup ... (k)

In the above example, the Earley-based algorithm with source CFG rules is used in Phase 1. In Phase 2, head and link constraints are examined, and unification of feature structures is performed by using the charts obtained in Phase 1. Candidate patterns are ordered by their weights and preferences. Finally, in Phase 3, the target charts are built to generate translations based on the selected patterns.

5 Integration of Bilingual Corpora

Integration of translation patterns with translation examples, or *bilingual corpora*, is the most important extension of our framework. There is no dis-

crete line between patterns and bilingual corpora. Rather, we can view them together as a uniform set of translation pairs with varying degrees of lexicalization. Sentence pairs in the corpora, however, should not be just added as patterns, since they are often redundant, and such additions contribute to neither acquisition nor refinement of non-sentential patterns.

Therefore, we have been testing the integration method with the following steps. Let T be a set of translation patterns, B be a bilingual corpus, and ⟨s,t⟩ be a pair of source and target sentences.

1. [**Correct Translation**] If T can translate s into t, do nothing.
2. [**Competitive Situation**] If T can translate s into t' ($t \neq t'$), do the following:
   (a) [**Lexicalization**] If there is a paired derivation sequence Q of ⟨s,t⟩ in T, create a new pattern p' for a pattern p used in Q such that every nonterminal symbol X in p with no head constraint is associated with h:X in q, where the head h is instantiated in X of p. Add p' to T if it is not already there. Repeat the addition of such patterns, and assign low weights to them until the refined sequence Q becomes the most likely translation of s. For example, add
   leave:VP:1:+OBJ considerably:ADVP:2 → VP:1
   VP:1 ← laisser:VP:1:+OBJ considérablement:ADVP:2
   if the existing VP ADVP pattern does not give a correct translation.
   (b) [**Addition of New Patterns**] If there is no such paired derivation sequence, add specific patterns, if possible, for idioms and collocations that are missing in T, or add the pair ⟨s,t⟩ to T as a translation pattern. For example, add
   leave:VP:1:+OBJ behind → VP:1
   VP:1 ← laisser:VP:1:+OBJ
   if the phrase "leave it behind" is not correctly translated.
3. [**Translation Failure**] If T cannot translate s at all, add the pair ⟨s,t⟩ to T as a translation pattern.

The grammar acquisition scheme described above has not yet been automated, but has been manually simulated for a set of 770 English-Japanese simple sentence pairs designed for use in MT system evaluation, which is available from JEIDA (the Japan Electronic Industry Development Association) ((the Japan Electronic Industry Development Association), 1995), including:

#100: Any question will be welcomed.
#200: He kept calm in the face of great danger.
#300: He is what is called "the man in the news".
#400: Japan registered a trade deficit of $101 million, reflecting the country's economic sluggishness, according to government figures.
#500: I also went to the beach 2 weeks earlier.

At an early stage of grammar acquisition, [**Addition of New Patterns**] was primarily used to enrich the set T of patterns, and many sentences were unambiguously and correctly translated. At a later stage, however, JEIDA sentences usually gave several translations, and [**Lexicalization**] with careful assignment of weights was the most critical task. Although these sentences are intended to test a system's ability to translate one basic linguistic phenomenon in each simple sentence, the result was strong evidence for our claim. Over 90% of JEIDA sentences were correctly translated. Among the failures were:

#95: I see some stamps on the desk .
#171: He is for the suggestion, but I'm against it.
#244: She made him an excellent wife.
#660: He painted the walls and the floor white.

Some (prepositional and sentential) attachment ambiguities needs to be resolved on the basis of semantic information, and scoping of coordinated structures would have to be determined by using not only collocational patterns but also some measures of balance and similarities among constituents.

6 Conclusions and Future Work

Some assumptions about patterns should be reexamined when we extend the definition of patterns. The notion of head constraints may have to be extended into one of a set membership constraint if we need to handle coordinated structures (Kaplan and Maxwell III, 1988). Some light-verb phrases cannot be correctly translated without "exchanging" several feature values between the verb and its object. A similar problem has been found in be-verb phrases.

Grammar acquisition and corpus integration are fundamental issues, but automation of these processes (Watanabe, 1993) is still not complete. Development of an efficient translation algorithm, not just an efficient parsing algorithm, will make a significant contribution to research on synchronized grammars, including STAGs and our PCFGs.

Acknowledgments

Hideo Watanabe designed and implemented a prototype MT system for pattern-based CFGs, while Shiho Ogino developed a Japanese generator of the

prototype. Their technical discussions and suggestions greatly helped me shape the idea of pattern-based CFGs. I would also like to thank Taijiro Tsutsumi, Masayuki Morohashi, Hiroshi Nomiyama, Tetsuya Nasukawa, and Naohiko Uramoto for their valuable comments. Michael McDonald, as usual, helped me write the final version.

References

Abeillé, A., Y. Schabes, and A. K. Joshi. 1990. "Using Lexicalized Tags for Machine Translation". In *Proc. of the 13th International Conference on Computational Linguistics*, volume 3, pages 1–6, Aug.

Berwick, R. C. 1982. "Computational Complexity and Lexical-Functional Grammar". *American Journal of Computational Linguistics*, pages 97–109, July-Dec.

Brown, P. F., S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. "The Mathematics of Statistical Machine Translation: Parametric Estimation". *Computational Linguistics*, 19(2):263–311, June.

Dorr, B. J. 1993. *"Machine Translation: A View from the Lexicon"*. The MIT Press, Cambridge, Mass.

Earley, J. 1970. "An Efficient Context-free Parsing Algorithm". *Communications of the ACM*, 6(8):94–102, February.

Fujisaki, T., F. Jelinek, J. Cocke, E. Black, and T. Nishino. 1989. "A Probabilistic Parsing Method for Sentence Disambiguation". In *Proc. of the International Workshop on Parsing Technologies*, pages 85–94, Pittsburgh, Aug.

Furuse, O. and H. Iida. 1994. "Cooperation between Transfer and Analysis in Example-Based Framework". In *Proc. of the 15th International Conference on Computational Linguistics*, pages 645–651, Aug.

Gazdar, G., G. K. Pullum, and I. A. Sag. 1985. *"Generalized Phrase Structure Grammar"*. Harvard University Press, Cambridge, Mass.

Jensen, K. and G. E. Heidorn. 1983. "The Fitted Parse: 100% Parsing Capability in a Syntactic Grammar of English". In *Proc. of the 1st Conference on Applied NLP*, pages 93–98.

Kaplan, R. and J. Bresnan. 1982. "Lexical-Functional Grammar: A Formal System for Generalized Grammatical Representation". In J. Bresnan, editor, *"Mental Representation of Grammatical Relations"*. MIT Press, Cambridge, Mass., pages 173–281.

Kaplan, R. M. and J. T. Maxwell III. 1988. "Constituent Coordination in Lexical-Functional Grammar". In *Proc. of the 12th International Conference on Computational Linguistics*, pages 303–305, Aug.

Maruyama, H. 1993. "Pattern-Based Translation: Context-Free Transducer and Its Applications to Practical NLP". In *Proc. of Natural Language Pacific Rim Symposium (NLPRS' 93)*, pages 232–237, Dec.

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

Pustejovsky, J. 1991. "The Generative Lexicon". *Computational Linguistics*, 17(4):409–441, December.

Sato, S. and M. Nagao. 1990. "Toward Memory-based Translation". In *Proc. of the 13th International Conference on Computational Linguistics*, pages 247–252, Helsinki, Aug.

Schabes, Y., A. Abeillé, and A. K. Joshi. 1988. "Parsing Algorithm with 'lexicalized' grammars: Application to tree adjoining grammars". In *Proc. of the 12th International Conference on Computational Linguistics*, pages 578–583, Aug.

Schabes, Y. 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*, 21(4):479–513, Dec.

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

Sumita, E. and H. Iida. 1991. "Experiments and Prospects of Example-Based Machine Translation". In *Proc. of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 185–192, Berkeley, June.

JEIDA (the Japan Electronic Industry Development Association). 1995. *"Evaluation Standards for Machine Translation Systems (in Japanese)"*. 95-COMP-17, Tokyo.

Tsujii, J. and K. Fujita. 1991. "Lexical Transfer based on Bilingual Signs". In *Proc. of the 5th European ACL Conference.*

Vijay-Shanker, K. 1987. *"A Study of Tree Adjoining Grammars"*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Vijay-Shanker, K. and Y. Schabes. 1992. "Structure Sharing in Lexicalized Tree-Adjoining Grammars". In *Proc. of the 14th International Conference on Computational Linguistics*, pages 205–211, Aug.

Watanabe, H. 1993. "A Method for Extracting Translation Patterns from Translation Examples". In *Proc. of 5th Intl. Conf. on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, pages 292–301, July.