SOME KEY ISSUES IN DESIGNING MACHINE TRANSLATION SYSTEMS

Keh-Yih Su* and Jing-Shin Chang**

Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.
*E-Mail : *kys%bugs@twnctu01.bitnet*
***d797907%bugs@twnctu01.bitnet*

KEYWORDS : Machine Translation System, Generative Power, Grammar Splitting, Mixed Parsing Strategy, Parsing Efficiency, Scored Parsing, Statistical Inference, Control Strategy, English-Chinese MTS, ARCHTRAN.

ABSTRACT : During the development of a Machine Translation System (MTS), many *trade-offs* must be made for the wide variety of formalisms and controlling mechanisms available. Among these trade-offs are the general issues of the *generative power* of the grammar, the formal *linguistic power* and *efficiency* of the parser, the *manipulation flexibility* for the knowledge, the mechanism for *knowledge acquisition*, the degree of *expressiveness* and *uniformity* of the system, the *integration* of the knowledge sources, and so forth. In this paper, some basic decisions which must be evaluated carefully for constructing a large scale system are discussed; and our working experience with an operational *English-Chinese* Machine Translation System, **ArchTran**, are presented to help make such decisions on trade-offs.

1. MOTIVATION

Due to the complicated nature of natural languages, there is not yet any true *deterministic* mechanism for natural language analysis. As a result, a large number of ambiguous constructions, either due to legal (inherent) *ambiguity* or *overgeneration*, is common to almost every sentence. Moreover, many *ill-formed* construction, which are common in real text are withheld from regular formalisms. To overcome such tough situation, many mechanisms had been proposed in the context of machine translation. The ample variety of formalisms, knowledge sources and controlling mechanisms, however, make it vague on which formalism or technique to adopt in order to build a practical system. Hence, it is necessary to identify the issues which are important and to make appropriate trade-offs when contradictory factors are encountered.

In contrast to artificial languages, natural languages are *nondeterministic*, *ambiguous* and *unrestricted* to a great extent. Hence, to resolve the problem of *ambiguity*, *overgeneration* and *ill-formedness*, a huge amount of information including the information of individual lexical items, the well-formedness conditions for phrase construction, and even the contextual information should be consulted. In practice, none of the contemporary grammar formalisms can really provide a complete and consistent framework to integrate the complicated linguistic knowledge required to resolve the actual problems in a large system. Therefore, *engineering* problems become important when the system scales up and produces such dilemma as illustrated in the following. For a large system, on one hand, we might want to provide as much real world knowledge as possible to the system, and develop new formalism to resolve some specific problems. On the other hand, we must prevent the system from incorporating large but

inconsistent knowledge, and make certain that the operational system satisfies system constraints like the limit of allowable processing time and the allowable cost for system construction. Hence, to attain the balance it is usually not possible to adopt an idealized formalism without any modification in a large system.

From the engineering points of view, in addition to providing flexible mechanisms for supporting various formalisms we must also incorporate *systematic* approaches for *acquiring* and *maintaining* the required knowledge, and some *efficient* means to *manipulate* and to *integrate* various types of the knowledge. For these reasons, many trade-offs must be carefully evaluated when constructing a practical MTS. In order to make proper trade-offs in such a complicated mechanism, we must first identify the *basic characteristics* of the various formalisms or mechanisms. In this paper, recommendations are given for making decisions on such trade-offs.

These suggestions are based on our working experience with the *English-Chinese* Machine Translation System, **ArchTran**, which was originally developed at National Tsing Hua University since May 1985. Currently, continuous improvement of this system is being made by the MTS team at the BTC R&D Center. Some of the distinctive features of this transfer-based MTS is the adoption of *mixed* (top-down plus bottom-up) parsing strategy, the *scored* parsing mechanism, and the corpus-based statistics-oriented paradigm for linguistic knowledge acquisition. It is now a commercialized system, and serves as the kernel of a VAN-based translation service center. Even though the main target domain for **ArchTran** is computer manuals and related documents, other types of manuals from various customers are also engaged. Some interesting phenomena observed during the development of this MTS are reported and discussed here. However, the main topic is not about the implementation details but focuses on our experience working with a large MTS and on what should be evaluated carefully in developing such a system.

2. SOME ASPECTS OF INTEREST

As mentioned above, in the context of nearly unrestricted vocabulary, highly ambiguous syntax and semantics intensive process, we must take advantages of various knowledge sources and control mechanisms so as to construct a practical machine translation system. However, we have a great variety of choice over various mechanisms, some of which may turn out to have equivalent power in expressing or manipulating knowledge. Hence, we are in the position of identifying the key issues in the system design. Based on our working experience in constructing the ArchTran MTS, the following issues which might not be complete are essential :

- the *computational properties* of the *grammar formalism*
- the design of the *analysis tool*— the *parser*
- the strategy for selecting the *preferred* analysis from ambiguous ones
- the mechanism for knowledge *acquisition*
- the *interaction* between different types of knowledge.

These issues can be clustered into four closely related topics : knowledge *representation*, knowledge *manipulation*, knowledge *acquisition* and knowledge *integration*. In other words, we are concerned with the formal power of the mechanism to express and support linguistic

knowledge. We wish to have a simple, flexible and efficient control mechanism for applying linguistic knowledge to resolve problems. We need systematic approaches for acquiring consistent linguistic knowledge, so that a large practical system can be constructed in a systematic way. In order to take advantage of different types of knowledge, we must also provide a good strategy to integrate these knowledge.

The *grammar formalism* determines the *linguistic felicity*, *expressiveness* and *computational efficiency* of the system (Shieber 1985). The *formal power* of the system is closely related to the grammar, because it is the basis for augmentation to *semantic* and even *pragmatic* analyses. Hence, a good grammar formalism will have significant effects on the system. However, due to the complexity of natural languages, some linguistic phenomena may not be effectively disposed within the theoretical frameworks of a grammar formalism. Therefore, such *special* or *irregular* phenomena, which are mostly *ill-formed* in the real text must be taken care of by some other means with precaution of not taxing the formal power of the system or its efficiency too much.

To achieve high efficiency and better quality, the analysis tool, namely the *parser*, is undoubtly a very important component of the system. The parser must be flexible enough to accept a wide variety of linguistic knowledge representations to support various *grammar formalisms* and to provide different *driving mechanisms* in order to implement essential linguistic operations for a machine translation system. In addition, parsing efficiency is still a crucial factor for a practical MTS. If the translation process can not be done within a *reasonable short time*, the benefit claimed by machine translation systems will be significantly cutback. Hence, various time-saving techniques must be explored so as to obtain the most appropriate translation within a tolerable time limit. Furthermore, since information for disambiguation may come from *lexical*, *syntactic*, *semantic* or even *pragmatic* level, the parser should also be capable of being driven from different levels.

Even with a good grammar formalism, a powerful parser and special facilities, ambiguities and ill-formedness are still inevitable in real application. Therefore, a good strategy must be provided to select the most *preferred* analysis from all the ambiguous ones. In this regard, the *scoring mechanism* for objective preference assignment is thus an important issue in this regard.

Traditional approaches to ambiguity resolution rely heavily on semantic analysis. With the advent of AI techniques, the text *understanding* is further emphasized in many rule-based systems or knowledge-based MT's (KBMT's). However, the complication of knowledge *acquisition* thus introduced may not be affordable in a practical large-scale system. The trade-offs between the degree of understanding and the cost for knowledge acquisition and maintenance are of importance to a system designer.

The above list of issues might look quite "classical". However, these topics are still very critical for constructing a practical system. Hence, it is worth inspecting these issues from *engineering* rather than from academic or linguistic point of view. In particular, we will provide some supporting evidences, whenever possible, to examine these classical issues and their significance in system construction. In this manner, we are more concerned with the *formal power* of a grammar formalism and the way to *acquire* the required knowledge, rather than on the *formalism* itself. In addition,, most issues discussed here are concerned with the *analysis* phase and its *control* mechanism. As a result, some of the linguistics issues, such

as particular grammatical formalisms, complex-feature-based lexicon structures, "full blown" semantic analysis and pragmatic analysis, and general issues of text generation, may not receive as much emphasis as they were in other literatures.

The reasons for emphasizing the analysis phases are manyfold. First, according to our experience with the ArchTran MTS in translating computer manuals and many other types of documents, the *transfer* and *generation* phases can usually be well-done so long as the *analysis* phase returns a correct analysis. Hence, emphasis must be placed on the analysis phase. Second, most formalisms currently available, such as unification-based grammars, can only deal with the *well-formed* input and produce *all* possible grammatical interpretations to a sentence. Many inherent ambiguities remain unresolved after the unification is conducted. They provide practically no way to assign *preference* to competing interpretations or *ill-formed* constructions which are abundant in real text. Hence, grammatical formalisms must be augmented with a *scored parsing* mechanism and *error correction* capability to be of any use in an operational system. In other words, it is important to reject ungrammatical interpretation with a good grammar formalism; but, what is also important is to deal with the inherent ambiguity and the ill-formedness with some objective *preference measure*. Third, in a large MTS, *knowledge acquisition* is a bottleneck in terms of *cost* and *consistency*. In a commercialized MTS, for instance, the number of lexical entries can amount to several hundred thousands. Hence, it is very costly to adopt "full blown" semantic analysis and extensive pragmatic analysis, which requires extensive lexicon information. Similar arguments are also advocated in Bennett (1990). Although many attempts had been made to improve efficiency and analysis results with such mechanisms with various degree of success in restricted domain, successful large-scale applications based on such mechanisms are still not reported to date.

All these issues will be discussed in the following sections. Most of the arguments reported here are based on our actual experience in developing the ArchTran English-Chinese Machine Translation System (Su 1985, 87a, 87b, 87c, 88, Hsu 1986, Chang 1987, Chang 1988).

3. GRAMMAR

Grammar formalisms nowadays are abundant. The lexicon structures range from simple monolithic categories to complex feature-valued categories. The formulations for syntactic regularities and semantic interpretations range from simple augmented phrase structure rules to complicated unification-based grammars. And the number of new grammars proposed are ever increasing. However, from an enginering point of view many formalisms actually do not differ significantly from each other in terms of the *expressiveness*, the underlying *linguistic knowledge* and the *control principles*. In practice, it is possible to adapt a monolithic category system to a feature-valued category system. The incorporation of control principles of the most prevailing unification-based formalisms into a simple phrase structure based system is not a difficult task either. Therefore, the capacity of a grammar from a system designer's point of view should be evaluated in terms of its formal power to realize the linguistic knowledge and the number of ambiguities introduced by the grammar. In addition, in the case of wide coverage, it should be possible to tackle the over-generative part. Furthermore, whether the grammar can be easily implemented and efficiently controlled is also a matter of concern. Hence, we will not use any particular formalism in this paper but examine the underlying grammar in terms of its formal properties.

3.1. Generative Power and Expressive Power

The *linguistic* capacity of a grammar can be evaluated in terms of two factors. The first is its *generative power*, defined as the ability to generate the strings of the source language. Intuitively, strong generative power is desirable in order to cover all possible syntactic structures of the source language. But this intuition holds only when the grammar can also express the syntactic structures precisely without generating extra ambiguities. Therefore, a second factor, namely the *expressive* power of the grammar, must be evaluated. This factor is the ability to describe the same language with fewer or no extra ambiguities.

Figure 3-1 shows a simplified conceptual view of the generative power and the expressive power of a grammar G with respect to the source language L. The projection of G and L, namely G' and L', on the X-Y plane represents the set of the surface strings covered by the underlying grammar and the true language, respectively. The extension into the Z dimension is the degree of inherent ambiguities for a given surface string. In other words, it stands for the number of (lexical, syntactic or semantic) ambiguities for the corresponding surface strings. In terms of grammaticality, the region covered by L' stands for the set of acceptable surface strings in the source language. On the contrary, the G' region is the set of surface strings derivable from the underlying grammar. The intersection of G' and L' (region B, C and arc <l,g>) corresponds to the set of surface strings both covered by G and L. The strings in the B-region have more grammatical interpretations (or ambiguities) than that of the true language. The situation is

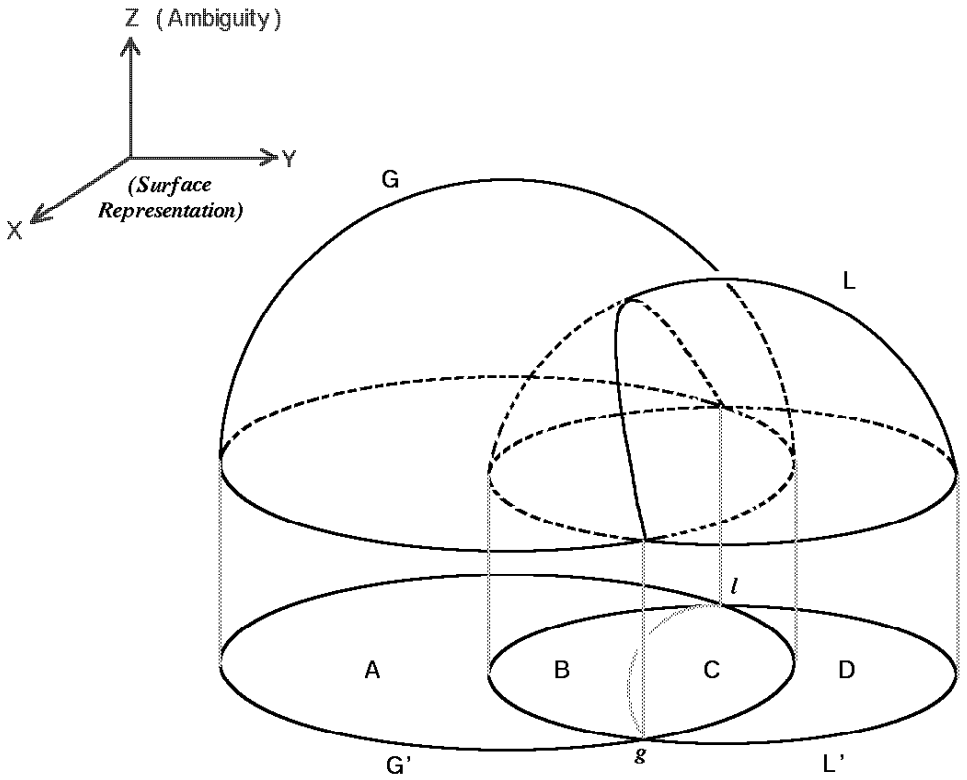


Figure 3-1. A Language Model for the True Language (L) and the Underlying Grammar (G) ofH the System

converse for the strings corresponding to region C. And the arc <l,g>, being the projection of the intersection of G and L, stands for the subset of the surface strings which have the same number of ambiguities both in G and L. In this figure, the generative power of the grammar can be expressed with the area of G', and the expressive power is measured by the the "closeness" or "distance" between G and L.

For a system designer, the regions of most concern are region A, region D and the volume above B and C. The A-region corresponds to the *over-generative part* of the grammar. The strings in this region rarely appear in the real text, and only consume the processing time with no other contribution. When the grammar is used for *top-down* parsing or *generation*, however, this part of state space will still be explored. Surface strings in the D-region are acceptable strings with respect to L. But unfortunately, they are beyond the generative power of the underlying grammar. They are usually so *special* or *irregular* that they can not be expressed *effectively* in terms of the grammar formalism. Hence, they are considered *extragrammatical* with some degree of *ill-formedness*. If we try to expressed these strings with the grammar by enlarging the coverage of G, excessive ambiguities would result. As for the strings in region B and C, they are covered by both G and L. Because most strings in these regions have more than one interpretations (i.e., $Z \geq 1$), ambiguity resolution and preference assignment are important for these regions.

For an MTS, it is reasonable to assume that most input sentences are well-formed. Hence, we might want to adopt an over-generative grammar with strong generative capacity to cover most syntactic structures. However, as the generative power of the grammar becomes stronger, G will in general become larger both in the X-Y plane (acceptable surface strings) and in the Z dimension (number of ambiguity). Although the coverage of grammatical input will increase substantially, the coverage of the over-generative part and the number of ambiguity for well-formed sentences would increase as well. As a result, the *expressive power* of the grammar might degenerate accordingly. The following rules give a good example of an over-generative grammar.

S –> X*
X –> a | b | ... | z | | A | B | ... | Z

These rules imply that an English sentence can be generated through the concatenation of an arbitrary number of letters and white spaces. They are generative but useless since they do not reveal any useful syntactic structure of the sentences. They also allow a large number of strings being introduced by the over-generative part of the grammar to be recognized as legal sentences. Hence, many strings which do not exist in real text will also be accepted. Therefore, one should carefully consider about the following possible disadvantages of an over-generative system.

1. It may not reveal the real syntactic structure of the input sentences.

2. It may introduce extra ambiguities for well-formed sentences and enlarge the state space of the over-generative part of the grammar.

In our opinion, to have a system with wide coverage, the grammar should be *a little over-generative* than is needed in the appropriate domain so as to cover most of the syntactic structures in the source language. But loosely designed grammar rules should be avoided because they

might increase the number of ambiguities for *well-formed sentences* and enlarge the state space of the grammar. The following reasons further support this assertion :

1. In a transfer-based MTS, if the success rate are equally likely for the *analysis*, *transfer* and *generation* phases, to acquire 80% accuracy in translation, on the average, at least 93% of the input sentences must be analyzed successfully ($0.8=(0.93)^3$). Therefore, the grammar should at least have such degree of *coverage* over the syntactic structures of concern.

2. The grammar of the source language is used mainly for *analysis*, not for *generation*. Furthermore, it is reasonable to assume that most input sentences are well-formed. Therefore, the effects of the over-generative part of the grammar over illegal input strings should not affect the system significantly if the grammar is properly restricted.

Therefore, a grammar with strong expressive power and to be a little over-generative than is needed in the target domain is recommended.

3.2. Grammar Type

The *computational* capacity of a grammar formalism is largely determined by its type. The type of the grammar also has substantial influence on the underlying parsing mechanism and the capability to support various grammar formalisms. According to the Chomsky Hierarchy (Aho 1972, Hopcroft 1979), a grammar or language can be classified as type 0 (*recursively enumerable*), type 1 (*context sensitive*), type 2 (*context free*) or type 3 (*regular*).

The type 0 language can be recognized by a *Turing Machine*. Hence, the automata for this language are essentially procedure driven. The procedural nature makes this type of grammar inappropriate as the formal linguistic representation for natural languages. On the other hand, the type 3 grammar is too restricted to be practical for natural language processing, because it has only the formal power of a finite automaton. Therefore, the appropriate base grammar falls into the categories of *Context-Sensitive Grammar* (CSG) and *Context-Free Grammar* (CFG).

Since a CFG is a proper subset of some CSG, the latter is mathematically more powerful. However, CFGs seem to be more suitable as the base grammar for natural language processing for the following reasons :

1. It is easier to describe a language with CFG than with CSG, because one merely describes the language by specifying the *hierarchical relationship* among constituents. One does not have to take care of the context constraints in the syntax. Alternatively, such constraints can be easily specified in an augmented CFG as semantic constraints. The CSG formalism is obviously more awkward as a natural way for specifying syntactic and semantic knowledge.

2. Most of the contemporary syntactic theories, for example, **GPSG** (Generalized Phrase Structure Grammar), **LFG** (Lexical Functional Grammar) (Sells 1985) and **FUG** (Functional Unification Grammar) (Kay 1985) were developed essentially within the limitation of context free grammar (Hutchins 1986). Therefore, a generalized phrase structure formalism (and hence an *augmented* CFG) can handle a wide variety of linguistic problems

of most concern through proper augmentation. As a result, none of the distinguished syntactic theories nowadays is based on CSG.

3. CFG is proved to be suitable for most programming languages. It is also very easy to tailor to MTs' need by augmenting the grammar with functional annotations for semantics as is the case in the *Augmented Transition Network* (**ATN**) paradigm (Woods 1970b); a grammar formalism augmented in this way is claimed to have the formal power of a *Turing Machine* (Bate 1978). In addition, efficient parsing algorithms are available for CFG parsers (Aho 1972). On the other hand, the parsers for CSG are more complicated and they still have some difficulties in handling infinitive analysis (Woods 1970a).

The above arguments suggest the adoption of a CFG as the base grammar for describing the source language. However, the formalism must be mildly augmented with semantic actions to account for the semantic aspects of natural languages. For instance, the augmentation should contains facilities for checking semantic constraints and facilities for changing the semantic roles of various constituents. The ATN paradigm provides such a semantic model. Contemporary unification-based grammars also use functional annotation for such tasks. In our system, an augmented CFG, which is augmented by the ATN-like *condition-action pairs* (Hsu 1986, Su 1987a) is adopted for the same reasons.

The above discussions are mainly concern with the degree of *context-freeness* of a language. However, in the contemporary grammar formalisms other aspects for dealing with semantics are also emphasized. Some formalisms lay stress on the importance of *lexicon*, and other formalisms highlight the simplicity to synthesize attribute with *unification-based* operations. These various formalisms may look very different at the first glance, but the implementation may turn out to be the same when they are transformed into computer manageable forms. Therefore, to take advantages of various formalisms, several points should be alerted when comparing the formal power of two different formalisms. A good formalism should have the following characteristics :

1. In *representation* layer, it should provide facilities which enable a linguist to express linguistic knowledge in an elegant way. In other words, the formalism must be flexible enough so that *declarative* knowledge, as well as *procedural* knowledge, can be easily described. Such flexibility is required for the system to support and take advantages of various new formalisms without changing the system drastically.

2. In *implementation* layer, it should be easy to map the semantic objects in a formalism to a good internal data structure for efficient processing of the objects. In algorithmic point of view, a good internal data structure is definitely a prerequisite for efficient processing. Therefore, a good formalism should have a good mapping to an appropriate data structure which can be efficiently processed.

In addition, some augmentation may be necessary to make the *control* strategy of the system more effective, or to patch up the deficiency which is not covered by the conventional linguistics. For example, mechanism for dealing with *inexact knowledge* should be included in the augmentation so as to handle decision problems, such as *preference* assignment which can not be dealt with exact rules. As another example, *lexicon-specific mechanism* is often required

to tackle linguistic problems which can not be effectively handled with general rules. Therefore, a good formalism also should be easily augmentable in these aspects.

3.3. Grammar Splitting

Sometimes, it is desirable to subdivide the grammar rules into several functionally independent groups according to their syntactic categories in order to provide more flexibility for semantic analysis. In particular, we may want to parse a *nonsentential* constituents such as an NP or VP independently. The necessity of splitting a grammar into *subgrammars* is thus evident.

Some well-known paradigms, such as the ATN model (Woods 1970b) and the PARSIFAL system (Marcus 1980), are good examples which adopt grammar splitting mechanism. In the ATN model, the nonterminals on the arcs of a transition network actually corresponds to a set of subgrammars modeled as individual subnetworks. In the PARSIFAL system, Marcus uses a number of *rule packets* to drive the parser. These rule packets contain a number of *pattern-action* pairs. A fixed-size buffer is used to match specific input patterns; when a match is found, the corresponding action is performed.

In the **ArchTran** MTS, the subgrammar concept is implemented by adding a special action, called *wake-up*, to the subparsing tables of a revised LR parser (Aho 1972, 1986). The grammar is logically divided into subgrammars for constructing *major constituents* like S (sentence), NP (noun phrase), and VP (verb phrase). For each subgrammar, a subparsing table is constructed just like a conventional LR parser. However, when a nonterminal **X** which is not defined in the current subgrammar is encountered in some state **s**, the set of lexical categories **C** which are the allowable first symbols of this nonterminal symbol is enumerated. Each wake-up action corresponding to the triple (**s**, **C**, **X**) is then kept in the subparsing table for the current subgrammar. In this manner, major constituents are essentially constructed *bottom-up*, but *top-down* predictions are made for activating other major constituents (**X**'s) in the appropriate states (**s**) when the set of admissible lexical categories (in **C**) appear in the input stream. Hence, it provides a convenient way to construct each major constituent *independently*. It also makes the implementation of a mixed parsing strategy called *Top-Down Filtering with Bottom-Up Parsing* (Pratt 1973, 1975, Su 1987a) or **TD+BU** parsing very easy. (More details will be discussed in Section 4.2.) In comparison with an earlier version of the **ArchTran** parser, which is purely bottom-up, the grammar splitting strategy shows some desirable features :

1. The *generative power* remains unchanged after the grammar splitting, but the capability for predicting more likely parsing paths is improved, for the *top-down filtering* capability is provided in addition to bottom-up data-driven parsing. In general, linguists tend to express their linguistic knowledge in a top-down manner. Therefore, we can usually take advantage of this fact with the *filtering routines* associated with each constituent to "filter out" impossible parses in advance. For example, we can use *syntactic attributes*, such as the *subcategorization* features, to prevent from constructing impossible syntactic structures associated with each *verb*.

2. Grammar splitting makes *partial parsing* possible because each constituent can be parsed independently of each other. Since the processing unit is not confined to a sentence

NUMBER OF ACTIONS PER STATE		1	2	more than 2	TOTAL
BU *	Number	40,005	12,829	3,204	56,038
PARSER	Percentage	71.4%	22.9 %	5.7 %	100 %
TD+BU **	Number	25,860	3,069	827	29,756
PARSER	Percentage	86.9 %	10.3 %	2.8 %	100 %

NOTE : The comparison is made in July 1987.
* BU refers to the Bottom-Up Parser (without subgrammars).
** TD+BU refers to Top-Down Filtering with Bottom-Up Parsing (with subgrammars).

Table 1. Comparison of conflict actions between BU and TD+BU parsers. This table shows the effect of grammar splitting on the number of conflict actions.

any more, we can conduct many special treatment more flexibly. For example, the identification of a *split idiom*, such as "turn NP on", is possible if the NP subgrammar can be activated independently. The treatment of *nonsentential* input (e.g., *titles* and *parenthesesized text*) also benefits from such feature. When combined with the *chart* mechanism which keeps the partial parses, it is also possible to realize *fail soft* strategy (Bennett 1985) for error recovery. Such error recovery strategy is very important for occasional *ungrammatical* (yet *acceptable*) input sentences.

3. The number of *conflicts* in the parsing table is reduced when subgrammars are used. The main reason for this reduction is that many actions taken in a conventional LR parser are substituted with the *wake-up* actions. Although the parsing state space is still the same, one can eliminate the whole state space of a subgrammar with filtering routines to improve the efficiency when waking up new subgrammars. Thus, the effective parsing paths have smaller branching factors. The number of paths to try and the number of backtracking are reduced accordingly. Table 1 shows this desirable behavior. It shows that the parsing table entries with two conflict actions (e.g., shift-reduce conflict) is reduced from 22.9% to 10.3% when subgrammars are used. The entries with more than two conflict actions are less than 3% in the case where subgrammars are used.

However, some adverse effects with subgrammars are also observed. Table 2 shows these effects :

1. Inter-subgrammar operations must be encoded, thus increases the complexity of the parser. The code of the system increases by about 53%, most of which is attributed to the extra effort to handle inter-subgrammar operations.

2. The size of the parsing table increases when subgrammar approach is adopted; because additional information for properly waking-up other subgrammars are required. Table 2 shows that the parsing table increased by about 1.33 times.

Since the disadvantages just mentioned are mainly technical, and they can be easily handled, the grammar splitting strategy is still recommended.

	Code Size (Bytes)	Parsing Table (Bytes)	TOTAL (Bytes)
* TD+BU PARSER	440 K	3.01 M	3.44 M
** BU PARSER	288 K	2.26 M	2.54 M
TD+BU : BU	1.53 x	1.33 x	1.35 x

NOTE : The comparison is made in July 1987.
* TD+BU refers to Top-Down Filtering with Bottom-Up Parsing (with subgrammars).
** BU refers to the Buttom-Up Parser (without subgrammars).

Table 2. Comparison of the size of the parsing tables for Bottom-Up and TD+BU parsers. This table shows the effect of grammar splitting on the size of code and parsing tables.

3.4. Extragrammatical Phenomena

As we can see, the power of a parser originates from its capability to handle *regular* linguistic phenomena in an efficient way. However, it is not always possible or effective to express all linguistic phenomena in terms of grammar rules. Some of them, though can be described in this way, tend to enlarge the state space of the grammar. For example, if we try to express *split idioms* (non-contiguous idioms, such as "turn ... on") in terms of grammar rules, many extra trials for regular syntactic patterns will be made. Hence, the state space extended by the grammar will become enormously large and the system will be over-loaded with ambiguities thus introduced.

Most of these special phenomena can be resolved by proper *normalization* of the input strings into a normalized token stream before they are passed to the parser. For this reason, we suggest that a *pre-processor*, as described in Su (1987a), be incorporated into the system to take care of such special structures like conjoined elements, formula in the text, split idioms, and so on. Other special phenomena are caused by the ill-formed input text. In a practical MTS, a substantial fraction of the real text contains ill-formed sentences. The degree of ill-formedness may range from simple typographical errors and violation of agreement to ellipsis in dialogue. To cope with such *extragrammatical* phenomena, it would be more practical to adopt some fail-soft *error recovery* strategies and *preference* assignment mechanisms in the parsing mechanism rather than express these constructions in the grammar.

4. PARSING STRATEGY

In Machine Translation Systems, to choose a proper parsing strategy is very crucial because it plays the central role in the analysis phase. Both *flexibility* for dealing with semantics and *efficiency* in parsing will be affected by the choice of the parsing strategy. The ultimate goal of the parser is to get the *correct* analysis *in a reasonable short time*. If we can not get a good translation within a reasonable time limit, the benefits claimed by MT would be lost. Such goal can be achieved in several ways. First, the strategy must provide flexibility for applying linguistic knowledge in different ways so that a *good* analysis is possible. To improve *efficiency*, one should traverse the state space of parsing in the direction of smaller branching factors. Further efficiency improvement can be achieved by eliminating duplicated or unlikely parsing process. One can also use *semantic* constraints to reject ungrammatical paths. However, because *ambiguity* and *ill-formedness* are abundant in the real text for a practical MTS, supporting a particular grammar

formalism (e.g., a unification-based grammar) to acquire a grammatical analysis does not really resolve the problems. Because they provide no way to select the *most probable* analysis. Hence, we must also have a *scoring mechanism* to select the most *preferred* interpretation. In the following sections, we shall examine these topics in more details.

4.1. Top-Down versus Bottom-Up

Conventional parsers can be roughly classified as either bottom-up (BU) or top-down (TD). The order of rule application of the grammar provides one degree of freedom for the system designers in several aspects. First, it provides two possible directions for applying syntactic rules. Therefore, we can choose a direction with smaller branching factor. Second, we have the freedom for applying semantic actions with two different points of view. The capability and flexibility for supporting linguistic knowledge thus differs significantly in these two directions. Hence, we can choose to apply semantic constraints to small constituents or large constituents depending on the strategy. In general, the top-down *hypothesis driven* mechanism (Marcus (1980)) differs from the bottom-up *data driven* mechanism in the following aspects :

First, semantic actions can be described more naturally in a top-down fashion. With a top-down parser, the hierarchical relationship and functional dependency between the constituents are explicit when traversing the parse tree. Therefore, it provides a way to apply the semantic constraints under a well-established relationship between the constituents. This characteristics is especially important when the semantic actions associated with the construction of one node needs to refer to the attributes of its contextual nodes.

On the other hand, it is more awkward for a bottom-up parser to deal with semantic actions which must refer to its *context*, since the relationship between the constituent to be formed and its context is still unknown while it is being constructed. The lack of such information may cause problems. For example, while forming an embedded relative clause, the characteristics of its antecedent may be consulted to determine whether the required *selectional restriction* are met. Since many noun phrases may precede the relative clause and the possible relationship and dependency between them are not yet established, the judgement must be deferred to a later time. However, *composition* of semantic features might be more easy for a bottom-up parser. For instance, in contemporary unification-based grammars, the *unification* operation is performed essentially in a bottom-up manner.

Second, it is usually *less efficient* to use a *top-down* parser in comparison with a bottom-up approach. For example, the English grammar for the "TD+BU" parser in Table 1 has 718 production rules, and there are 85 nonterminals for this grammar. On the average, each nonterminal is associated with 8 production rules. Hence, each call to parse a nonterminal may try as many as 8 alternative paths (or arcs in a transition-network) with a top-down parser. On the contrary, most state transitions (97.2%) can be accomplished within 2 trials by the bottom-up parser as shown in Table 1. Therefore, bottom-up parsing is more efficient for English, because the strategy is *data-driven* and it is *less branching* for English. When the size of the grammar becomes large, a great deal of backtracking is required for a top-down parser. The cost for top-down parsing is then very expensive, especially for long sentences (Liu 1989b). However, the

parsing efficiency really depends on the characteristics of the source language and the grammar (see Section 4.3. for further discussion.)

Third, the *error detection* capability of a top-down parser is poor. It may recursively apply the rules to match an *expected* constituent before it finds out a mismatch occurred between the terminal node and the expected one. On the other hand, a bottom-up parser can decide whether the next input is an illegal one immediately after the next token is available. This difference is significant when the input has multiple parts of speech. Early rejection of illegal combination of syntactic categories is possible for a bottom-up parser; while for a top-down parser, one must defer the rejection until all alternative paths have been tried.

From the above discussion, we conclude that the top-down approach is preferred as far as the semantic model is concerned. However, if we take efficiency into account, the bottom-up method is more desirable.

4.2. The Mixed Parsing Strategy

To take advantages of both TD and BU parsing strategies, a mixed parsing strategy is desirable. This approach is preferred since it provides the advantages inherent from both top-down and bottom-up parsers. In other words :

1. The parsing *efficiency* is retained by the bottom-up mechanism since it is *data driven* at the lowest level of parsing.

2. The *flexibility* to handle semantic actions is also preserved since a top-down view is provided.

For these reasons, the current ArchTran MTS adopts a mixed parsing strategy called *Top-Down Filtering with Bottom-Up Parsing* or *TD+BU* strategy (Su 1987a). By *bottom-up parsing*, we mean that all major constituents are constructed bottom-up. By *top-down filtering*, we mean that each nonterminal being constructed can be predicted and constrained with an associated *filtering routine* in a top-down manner. These filtering routines are used either to block an illegal parsing path or to redirect the parsing path to a preferred one. In either case, linguistic knowledge as well as probabilistic preference can be used to predict the most probable construction.

In our implementation, a conventional LR parser is revised to combine bottom-up parsing with top-down filtering (Su 1987a). An augmented CFG is partitioned into a number of *subgrammars*. Actions for bottom-up parsing ("shift" and "reduce") and top-down filtering ("wake-up") are then encoded in the subparsing tables for each subgrammar. (See Section 3.3. for details.) During a top-down session, the *filtering routine* for a constituent can be used to prevent the parser from trying further if some evidences suggest that the nonterminal is not a good candidate. The parser can also be *redirected* to a more preferred parsing path by checking the status and the *score* associated with the current path through the filtering routine. This strategy is very useful for blocking illegal or unlikely paths when a key word with strong prediction capacity on the syntactic structures is encountered. For example, when a *verb* is encountered, the possible syntactic structures following the verb are strongly implied by its *syntactic attributes* or *subcategorization features*. Hence, the parser can block inappropriate candidates with the filtering routines immediately. The efficiency of the parser is thus improved.

The TD+BU strategy is also adopted in other systems such as LINGOL by Pratt (1973, 1975). In Pratt's system, each node is built with Cocke-Younger-Kasami (CYK) bottom-up algorithm (Aho 1972) and Earley's top-down algorithm (Earley 1970). The efficiency is achieved by running the CYK algorithm, and the next "goal" (i.e., the expected constituent) is predicted by Earley's top-down algorithm. Since the modified LR TD+BU parser that ArchTran adopted employ the same control strategy to achieve bottom-up parsing and top-down filtering at the same time, it is easier to implement than the parser of LINGOL. However, both parsers have increased flexibility and parsing efficiency because the mixed strategy is adopted. Therefore, a mixed parsing strategy is highly recommanded.

4.3. The Efficiency of Parsing

Aside from the capacity to cope with semantics, *efficiency* is probably the most important consideration for a parser. The parsing efficiency can be improved in several ways. Our experience shows that the following factors are most important :

1. The efficiency is improved if it is (at least partially) *data driven* (Marcus 1980).

2. The efficiency is improved if *duplicated parsing processes* can be eliminated during backtracking in the course of nondeterministic parsing. This can be done with traditional *chart* mechanisms (Kay 1980) or state merging technique (Tomita 1987).

3. The efficiency is improved if the *degree of branching* of the parsing paths can be locally bounded to a more plausible region with well-designed *scoring mechanism* (Su 1987b, 1988, 1989a, 1990a).

Data-Driven Parsing

The simplest way to improve parsing efficiency is to adopt a data-driven mechanism which traverses the state space with less branching. Since bottom-up parsers are data-driven, it can prevent itself from trying impossible parsing paths in advance. Therefore, *bottom-up* parsing is inherently more efficient than top-down hypothesis-driven parsing. However, the situation may also depend on the characteristics of the source language and the underlying grammar. In general, if a constituent takes many different surface forms, that is, many grammar rules take the form of

$$X \rightarrow Y_i$$

where Y_i's stand for different syntactic patterns of X, then the branching factor in the top-down direction will be higher. Conversely, if a given syntactic pattern has many possible syntactic functions, that is, the underlying grammar contains a lot of rules of the form

$$X_i \rightarrow Y,$$

where X_i's are different syntactic categories which take the same form Y, then the degree of branching in the bottom-up direction will be higher. In our experience, top-down approaches are less favorable for English because the number of rules of the first type are far greater than rules of the second type. (Refer to Section 4.1.) Without any semantic constraints, the average speed-up factors of an LR parser over a top-down implementation of the ATN parser can amount

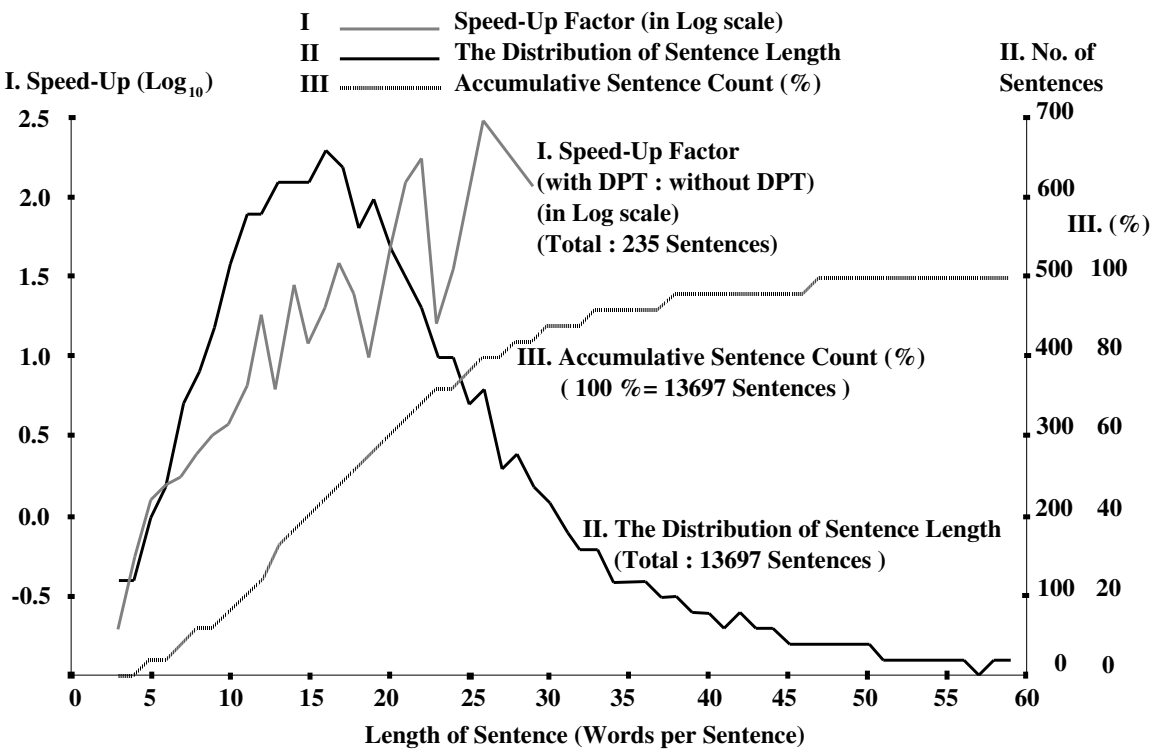


Figure 4-1. Speed–Up with DPT and the Distribution of Sentence Length

up to 20 even for a small grammar; and the time to get the first parse in the ATN parser is even much longer than with the bottom-up LR parser (Liu 1989b).

Chart Parsing and Structure Sharing

The second way to improve parsing efficiency, is to prevent *duplicated parsing*. Because most natural languages can not be parsed deterministically, backtracking is almost inevitable. During the backtracking process, the same constituents may be constructed repeatedly and a lot of time is wasted for the same actions. To avoid duplicated parsing, the *chart* parsing mechanism was proposed (Kay 1980, Winograd 1983). Any constituents will be constructed only once and be shared henceforth. The fact that physical data structure can be shared among various logical paths makes such mechanism very powerful in improving parsing efficiency.

The chart mechanism can be implemented in several ways. In **ArchTran**, a *Dynamic Phrase Table* (**DPT**) is designed to store the subtrees already constructed together with other information about the subtrees. It is similar to the triangular matrix (*"parse table"*) used in CYK parsing algorithm (Aho 1972). Each table entry DPT(i, j) contains the list of pointers to the subtrees which can be constructed by input words w_i ... w_j. The DPT is checked whenever the parser is driven to construct an expected subtree during the top-down session. If the desired subtree is already in the DPT, no additional parsing for this constituent is necessary. Otherwise, the constituent is constructed bottom-up. In this manner, the problem of duplicated parsing can be readily eliminated.

Figure 4–1 shows the improvement in parsing speed when DPT is used in comparison with a previous version of the parser where DPT is not implemented. (All possible parse trees are constructed for comparison. The speed-up factors are shown in logarithmic scale of base 10.) Note that the improvement is less obvious when the length of the sentence is small. In these cases, duplicated parsing is less significant and the operations for DPT become unnecessary. For sentences of moderate size (e.g., with 8 words or more), the speed-up factors range from 2.6 to 303.1. The distribution of sentence lengths of 13697 sentences is overlapped in the figure to show the effects of speed-up in the real context. In the sample text, only about 7.4% of the sentences are less than 8 words. About 77.4% of the sentences have 8 to 29 words, and sentences longer than 29 words are about 15.2%. On the average, the length is about 19.6 words per sentence. Figure 4-1 shows a tendency of having large speed-up with sentence length. In some extreme cases, the parser without the DPT mechanism may run for several CPU hours without any result while the parser with DPT obtains its output within only a few minutes. A speed-up factor of over 300 is observed in these situations. Since the efficiency of a practical system is dominated by *long* sentences, and the speed-up factor as a function of sentence length tends to be *exponential*, this study shows that the average behavior of speed-up with the DPT mechanism is quite satisfactory. The *expectation* of the speed-up factor is about 43 for sentences with the distribution of Figure 4-1. Therefore, a *structure sharing* mechanism, like DPT or general chart mechanism, is highly recommended for improving parsing efficiency.

State Merging

Another technique to improve parsing efficiency is to merge different paths which enter the same state when the next input symbols for these paths are the same. With such *state merge* technique, one can maintain multiple parses without duplicated parsing, because the subsequent operations for these paths will be the same. An extended LR parser based on this technique was first introduced by Tomita (Tomita 1987) for the CMU MT system (Carbonell 1987). By the adoption of a *graph-structured stack* to perform such state merge operation and the use of *subtree sharing* and *local ambiguity packing* techniques to store the parsed subtrees, Tomita's Algorithm provides an elegant way to eliminate duplicated parsing. It is claimed that the speed-up factor of Tomita's Algorithm over Earley's parsing algorithm (Earley 1970) is five to ten folds from an observation of 40 sample sentences under 220 context-free rules (Tomita 1987).

It is quite interesting to compare the speed-up factors of Tomita's Algorithm over Earley's algorithm with our experiments on DPT-based parsing over non-DPT-based parsing. The speed-up factors of both parsers over their non-charted counterparts are about the same order for the same sentence lengths. (The basis for comparison, of course, might not be identical in terms of grammar size, implementation techniques, and so on. However, the main difference in both pairs is the mechanism for structure sharing. Hence, their effects are comparable.) Since both parsers adopt chart-like mechanism to avoid duplicated parsing, it is reasonable to conclude that efficiency improvement in Tomita's parsing algorithm and the DPT parser are largely due to their use of the chart-like mechanism. The use of *state merging*, *subtree sharing* and *local ambiguity packing* is just another implementation of the chart mechanism. Hence, it has the same effect as the DPT in our system and other chart implementations.

The result can also be explained in another way. From an algorithmic point of view, the efficiency of a parsing algorithm depends mainly on the time for tree construction (including creation and manipulation of the stack and the tree nodes). Hence, how to redirect the state transition like the state merging technique is not the dominating factor for efficiency improvement. In other words, the result of time-saving in Tomita's algorithm and similar variations are largely due to the *sharing of physical storage* (i.e., the *chart mechanism*), not state merging. Hence, when one wishes to build an efficient parser, the *structure sharing* principle should be kept in mind.

One thing to note is the restriction of Tomita's Algorithm and most chart parsers. Most chart parsers are basically based on the *dynamic programming* principle (Horowitz 1978). However, the *dynamic programming* technique implicitly implies the *context-freeness* of the tree construction process. In other words, it assumes that the construction of a nonterminal is independent of its context so long as the constituents of the nonterminal are available. For this reason, the chart parsing, subtree sharing or DPT approaches must be modified when the effects of context are taken into account. For example, many semantic actions would assign different attributes or feature structures to a node based on the *contextual environment* of the node. Under such circumstance, the nodes shared in Tomita's Algorithm may no longer be sharable because different nodes with different attributes or feature structures should now be regarded as different entities.

Scored Parsing and Truncation

With a chart-like mechanism, we can avoid reparsing any constituent more than once and improve the parsing efficiency drastically. However, we also get *all* ambiguous constructions allowed by the underlying grammar. It is obvious that only the analysis that *best* fit human interpretation is useful for translation. With a chart parser, we still waste a lot of time in constructing thousands of useless constructions which will be discarded eventually. Hence, the fourth technique to improve parsing efficiency is to reduce the number of invalid trials by searching only the subspace where the best analysis is most likely to be found.

Traditional approaches to cut down the searching space is to first reorder the underlying grammar rules according to rule application frequency or to impose semantic constraints on the parses, then the first parse is taken as the desired analysis. In general, the first parse is usually not the best analysis unless a good scoring mechanism is provided. It is also not guaranteed that the path which shows the greatest preference in the first few steps of analysis will be the best one. Hence, such searching strategy (e.g., depth-first search) which traverse only single path with *ad hoc* scoring mechanism would be inadequate.

To avoid *full search* and *ad hoc preference measure*, we prefer a *scored truncation* strategy proposed by Su (1987b, 89a, 90a). The scored truncation strategy can be implemented in two ways. The first approach is the *parallel truncation* strategy (Su 1987b) (or beam search in some AI literature). With this strategy, each path being tried is assigned a *score* based on a *score function* (Su 1988, 89a, 90a, 90b, Chang 1990). The maximum number of parallel parsing paths N is bounded according to the scores. Any path with lower score is "truncated" (refrained from further parsing) if it is not among the N paths with the highest scores or if its score is below a threshold. If more than one grammatical analyses are found in the subspace of parsing, the

parse with the highest score is selected, otherwise the algorithm backtracks to the next possible subspace for continuation until the parse with the highest score is found.

An alternative implementation of the scored truncation strategy is to traverse only one path at a time sequentially. The searching process is directed by the score function and a set of dynamically adjustable thresholds. These thresholds serve to truncate unlikely parses by comparing the threshold of each stage and the current accumulative partial score at the end of each step. After the first parse is acquired, each new parse will also be compared with the analysis which has highest score among all. Hence, the best analysis (with respect to the score function) can be returned at the end of the parsing process. Such strategy is called *sequential truncation* (Su 1989a, 90a). With such *scored parsing* techniques, only a small fraction of the most promising parsing state space is traversed. Hence, the efficiency is improved. To summarize, the scored truncation strategy is preferred for the following reasons :

1. It can effectively constrain the parsing paths to a subspace without exploring the *full* state space exhaustively. With a well-designed score function and a proper control mechanism for setting and adjusting the thresholds, we can search only a small amount of the paths. The efficiency can thus be improved. When the subspace of searching is sufficiently large and a good scoring mechanism is used, it is very likely that the local optimum thus obtained is also the global optimal analysis.

2. The maximum number of parsing paths (N) in parallel truncation and the dynamic thresholds in sequential truncation can be *dynamically* adjusted according to the complexity of the sentence structure to avoid premature truncation before sufficient information is available. Thus, by adjusting the variable N or the dynamic thresholds, the *speed-quality* trade-off can be tuned to a more favorable status in a very simple way.

3. Most systems today adopt the *hard-rejection* approach to cut down the state space introduced by the over-generative part of the underlying grammar. An analysis is either *accepted or rejected* depending on whether some constraints are satisfied or not. Such mechanism makes the maintenance of the underlying rule bases of the system very difficult, and provides no practical solution for multiple analyses. (Refer to Section 4.4. for further discussion.) On the contrary, the scored truncation algorithm provides a *soft-rejection* strategy to cut down the searching space. It would not block a path that fails to meet some constraints. Instead, the analysis is evaluated in terms of a *preference* measure. A path with a low score will still be tried if better analyses can not be found in the most promising searching space. Hence, the above problems will be relieved with the scored parsing strategy.

In our experiments (Su 1990a), the speed of a BU *non-chart* parser with *sequential truncation* is compared with the TD+BU *chart* parser mentioned in the previous sections. The average speed-up factors of the *truncation* strategy over *chart* parsing can reach as high as 17 and 9 for sentences from the score database and sentences outside the database, respectively. (The average speed-up factors are derived from the individual speed-up factors for each sentence length weighted by the distribution of the sentence lengths from 13565 sentences.) These experiments show that scored truncation strategy can provide significant improvement in parsing efficiency

over chart parsing. However, a good *scoring mechanism* for preference assignment is essential to acquire better analysis. This topic will be discussed in more details in Section 4.4.

Scored Truncation versus Chart Parsing

As shown in previous sections, both chart parsing and scored truncation can improve parsing efficiency drastically. Can these two strategies be combined to further improve the parsing efficiency? Unfortunately, the different philosophy behind these two strategies make them hard to combine, because a chart parser tries to *retain* as many partial constructions as possible while a scored truncation strategy tries to *discard* as many unlikely constructions as possible in the course of parsing.

In scored truncation, one needs to take all nodes in a parse tree into consideration so as to assign a fair score to an interpretation. Hence, the score is essentially a function of the current constituent under construction and its context. In some cases, a subtree may appear less favorable because the associated score is low under some contextual environment. Under such circumstance, the subtree may be deleted from the chart. But the same subtree may have a high score under another contextual environment, and thus should be retained for later parsing. However, because this subtree has already been discarded, the parser will not find this subtree at a later stage when the parser backtracks to the state where the chart is consulted. Therefore, the subtree can not be referred. This problem shows how the context affects the efficient use of a chart mechanism. It also makes the combination of truncation and chart a hard task (Su 1990a). In fact, even if we can combine both strategies, the overhead might be too large to be worthwhile.

4.4. Preference Measure

The efficiency improving techniques mentioned above are useful in providing solutions for reducing the searching space in the state space defined by the underlying grammar. However, the remaining searching space are still very large for most practical systems. To overcome such problem, most contemporary systems apply semantic constraints to cut down the state space of the system further.

However, such mechanisms have some intrinsic problems. First, they can only cut down the state space introduced by the over-generative part of the grammar. Hence, they have no way to deal with the extragrammatical or *ill-formed* input in the real language. Second, the mechanisms are essentially *hard-rejection* approaches. Because these constraints are usually subject to exceptions (e.g., see Ingria (1990)), they may cut down the *grammatical* part of the real language as well. When it comes to such situation, the troubling constraints must be modified to deal with the exceptions. The state space of the grammar is therefore changed to-and-fro accordingly. Third, *inherent ambiguities* will not be resolved by such mechanisms, because they provide no way to assign preference for all grammatical interpretations. Unfortunately, in our experience, the documents from customers contain a substantial fraction of *ill-formedness* of various types. In addition, the *inherent ambiguity* for a large grammar is also significantly large. Therefore, a *scoring mechanism* is an essential component for a practical MTS.

By the scored truncation as mentioned previously, the state space of the grammar can be made relatively stable because we do not actually change the state space of the underlying

grammar but only restrict our searching space to the most probable subspaces. It also provides a method for ranking multiple grammatical interpretations with respect to a *preference* measure. Thus this strategy can avoid rejecting input text which are only *slightly semantically anomalous*, and select the most preferred interpretation when there are *multiple grammatical interpretations*. When the parser eventually fails due to *ill-formed* input, the score for the partial analysis can be used as cues for fail-soft *error recovery*. It also greatly relieves the linguists from the hard task of finding "exact" semantic rules (or constraints).

The central problem is : how to define a *good* scoring function for preference assignment? In general, the preference should be evaluated according to the information from lexicon, syntax, and semantics. The scoring mechanism should also satisfy certain *optimal criterion* to ensure the optimal status of the system in the course of system development. However, most scoring mechanisms in the literature are *ad hoc* in terms of the above criteria. The main approach among these is to count the frequency of the rule application (Fujisaki 1984), which may be normalized by the number of tree nodes (Caeyers 1990) or the number of *constraint satisfactions* (Wilks 1975a, 1975b, 1983). Such subjective preference measures either do not take lexical, syntactic and semantic knowledge into consideration at the same time or take part of them into consideration somewhat *heuristically*. Hence, it is doubtful whether they can be tuned to satisfy any optimal criterion. It is also questionable how to maintain such mechanisms.

To be more objective, the ArchTran MTS adopts a probabilistic preference measure called *score function* (Su 1988, 89a, 90a, 90b, Chang 1990). This *score function* integrates lexical, syntactic and semantic knowledge with a uniform formulation to assign a preference measure to each interpretation. It takes the form of :

$$\begin{aligned} \text{Score} &= P(\text{ Semantic_Feature, Syntax, Lexical_Feature} \mid \text{Input_String }). \\ &= \text{Semantic_Score} * \text{Syntactic_Score} * \text{Lexical_Score} \end{aligned}$$

where

$$\begin{aligned} \text{Semantic_Score} &= P(\text{Semantic_Feature} \mid \text{Syntax, Lexical_Feature, Input_String}), \\ \text{Syntactic_Score} &= P(\text{Syntax} \mid \text{Lexical_Feature, Input_Strings}), \\ \text{Lexical_Score} &= P(\text{Lexical_Feature} \mid \text{Input_Strings}), \end{aligned}$$

and `Lexical_Feature`, `Syntax` and `Semantic_Feature` are the sets of lexical, syntactic and semantic primitives used to encode the semantic interpretation of a sentence. The score function takes advantages of all knowledge sources from lexicon, syntax and semantics. The decision based on this score function can also be shown to have *minimal risk* in Bayesian sense. Hence, it is an objective measure rather than a heuristic one. The `Score` function can be decomposed into `Lexical_Score`, `Syntactic_Score` and `Semantic_Score` by applying the *multiplication theorem* of probability to the defining equation. In this manner, we can also apply different component of the score function independently to each analysis phase if it is desirable. (Refer to Su (1988, 89a, 90a, 90b) and Chang (1990) for technical details on how a semantic representation can be mapped into the primitives.)

Preliminary simulations on the power of the score function had been conducted with promising results (Su 1988). With just the *syntactic score*, the optimal parse in the training set can usually be ranked within the first two or three parses with the highest scores. And in many cases, the optimal parse is the one with the highest score. Further improvement is possible

by taking the *semantic score* into account. Specialized probabilistic semantic model (Liu 1989a, 1990) as well as generalized probabilistic semantic model (Chang 1990) for *semantic score* evaluation have been developed recently to improve preference assignment. The results are even more encouraging. Moreover, since the translation of the first few parses are quite similar or in some cases they are exactly the same, the first rank translation is usually close to the desired one from the post-editor's point of view. This means that the truncation strategy based on a carefully designed score function is appropriate for reducing the searching space while retaining good translation quality. Since the above score function is primary statistics-based, some problems concerning the statistics database might arise. These topics will be discussed in Section 5.3.

5. KNOWLEDGE ACQUISITION IN MTS

Although it is well-known that high quality MT (HQMT) is unlikely to be realized in a large unrestricted domain, HQMT in a restricted domain is still the major goal of most MT systems. To achieve such goal, many conventional MT systems place emphasis on "full blown" semantic analysis (Bennett 1990). In the recent years, many knowledge-based MT's (KBMT's) are also proposed with the emphasis of text understanding. The new generation MTS of CMU (Carbonell 1987) is one such example whose goal is to tackle the general problem of high quality MT.

The general philosophy of such MT systems is : high quality machine translation requires high degree of text *understanding*. To understand the text, complex-feature-based formalism and lexicon structure, complicated conceptual dependency, semantic nets as well as nontrivial control and inference mechanism are used. A lot of *world knowledge* and *discourse* cues are also introduced to resolve specific problems. It is believed that by incorporating more and more world knowledge, many translation problems can be well resolved, and high quality translation is possible.

However, these arguments must be carefully examined from a realistic point of view. In the next section, we shall examine the effects of increasing the complexity of language model, which is generally believed to be the most effective method for improving translation quality. Then we will discuss this problem in terms of its feasibility and cost-effectiveness in a large scale system. Through such discussion, some potential problems with the knowledge-based approaches will be revealed.

5.1 Model Complexity versus the Coverage in Unseen Text

A Machine Translation System is usually built by first giving a suitable language *model*, then the *parameters*, or so called *knowledge*, for the model is acquired *via* knowledge acquisition procedure either from the linguists or from a training corpus. Since the knowledge that a linguist gives is usually induced from the corpus he saw before, we might regard this kind of knowledge as if they are obtained from an implicit training corpus (we shall refer such corpus a "pseudo" corpus hereafter). Therefore, all the knowledge for the language model can be regarded as being obtained from a training corpus, no matter it is explicit or implicit. After acquiring the parameters from the training corpus, an MTS then uses these parameters to translate the text that is unseen before. Therefore, it implicitly assumes that the language model and parameters

acquired from the training corpus can properly characterize the sentences in the unseen text. Since all the useful work of translation is done on the unseen text, instead of the training corpus, what we really care is the performance for the unseen text, not for the training corpus. Hence, the above assumption must be carefully examined.

Although we hope that our model and parameters acquired from the training corpus can properly characterize the sentences in the unseen text, however, there are several factors which may prevent us from getting good result in the unseen text. First, the *model* may not be adequate to describe the real text. For example, a context-free grammar without augmentation may not be able to handle agreement problems. Therefore, with a simple model, we may introduce *modeling error*. Second, even if the model is appropriate, the knowledge required for the unseen text may not be covered in the training corpus. That is, we may encounter with linguistic phenomena which have not been considered before. In these cases, the problem of *poor coverage* will be introduced. Third, the unseen text may have different linguistic characteristics from what we have assumed due to domain mismatch or style mismatch. Such mismatch would prevent us from properly characterize the sentences in the unseen text with the knowledge acquired from the training corpus. Therefore, the performance of a system depends on the *modeling error*, the *coverage rate* in the unseen text and the *mismatch* between the training corpus and the real text. The translation quality can also be improved from these three dimensions.

These three factors interact closely with each other in real applications. For example, if we use a complicated language model with large dimensionality in the parameter space, we would need much more knowledge to cover all possible cases implied by the model. In this case, if the available knowledge in the training corpus are kept the same, we will get much lower coverage in the unseen text in comparison with a simple model. If the improvement due to more complicated model can not compensate for the loss due to the low coverage, the performance of the complicated model in the unseen text will be even worse than that of a simple one, even if some improvement are observed in the training corpus. In other words, if the training corpus used to tune the parameters (or knowledge) of a language model is not big enough, we might get a very good result in the training text, however, the performance in the unseen text, which is what we really care about, might be rather poor. Such phenomenon, called "over-tune", is well known in pattern recognition and speech recognition community. This concept can be further illustrated by the following simple example.

Suppose we have three points (the training corpus) which are the data points taken from a straight line (the real model) but have been perturbed a little bit, then we can use any polynominal with order higher than two to fit these three points perfectly. Most of the time, the high order polynominals (i.e., the complicated model) can fit better than the simple model of a straight line, because these three points might not lie in a straight line due to the perturbation. However, the better fitting for these polynominals are the results of *over-tuning* based on the training corpus. When the other data points (the unseen text) are observed, the polynominals with higher order may deviate more from the data points in comparison with the straight line model. This is the main reason why many laboratory systems which adopt very complicated model work well in a small training corpus but perform rather poor in the real application.

Since the cost in developing an MTS is directly related to the amount of knowledge to be acquired, the knowledge one can afford is limited by the fixed budget. So we must make a

compromise between the model complexity and the coverage rate in the unseen text. If the performance for the training corpus is much better than that for the unseen text, it might be already "over tuned". In this case, we should either increase the size of the training corpus (by spending more money to acquire more knowledge) to get higher coverage, or adopt a simpler model to raise the coverage rate for this model (as a simple model requires less amount of knowledge), if we must stay within the budget. In our experience, the coverage rate in the unseen text plays a dominant role in a practical MTS. Therefore, it is highly recommended not to adopt an over complicated model, because the low coverage rate in the unseen text may result in poor performance for the text from the customers.

From the above discussion, increasing modeling complexity may improve the performance of the system in the training corpus, because the system can be over-tuned to best fit what have been observed. However, unless the size of the training corpus is very large, its performance, in general, will degrade for the unseen text. Besides, it might not be cost-effective to adopt a complicated model as discussed in the next section.

5.2. Problems with Knowledge–Based Approaches

In general, it is agreed that *understanding* can be helpful in improving the quality of translation, and KBMT is attainable in a rather restricted domain. However, adopting a full blown semantic analysis in KBMT may not be *cost-effective* in a large-scale MT system. The reasons are manyfold.

First of all, the amount of knowledge required and the level of analysis needed for translation depend on the *domain* and *characteristics* of the linguistic problems to be handled. The purpose of text understanding is to retrieve the *meanings* of the sentences and answer some queries based on the understood meanings. To answer all queries implied in the sentences, extended understanding may be required. For *translation* problems, however, the basic problems to be attacked is the word senses of the source text and the word order of the target text. Hence, extended understanding might not be essential to get the correct translation, especially in a sublanguage domain, which most operational MT systems are restricted to. Some problems which seem to be important in language *understanding* may not be as important for language *translation*. For example, while it is important in text understanding to consider the discourse environment to understand the actual reference of a pronoun, it is not essential for translation in many cases, because the readers of the translated text can resolve the referential problem by themselves much better than any text understanding mechanism while reading the text.

In our research (Chang 1990), it is shown that some important tasks, such as ambiguity resolution, can be well characterized by a *finite* dimensional semantic features without resorting to full blown semantic analysis and text understanding. Although extra information can be helpful for some specific instances, it might turn out to be redundant for other instances. Therefore, the knowledge to be acquired should be confined to the most important dimensions of the problem space for cost-effectiveness.

Second, the main problems with machine translation at the current stage are still quite "classical". Many problems, such as lexical disambiguation, structural disambiguation, and so on, can be resolved (or at least explained) at the syntactic and semantic levels. Although real

world knowledge is required in some specific instances, they rarely occur in the domain of technical manuals. Hence, the demand for deep understanding of the text is rare. In fact, most operational MT systems today can resolve these linguistic problems at the surface structure level, and the translation quality can be improved with semantic features and lexicon-specific information without resorting to text understanding. The real problem, in our opinions, is the lack of an *effective* computational mechanism for *acquiring* and *controlling* the vast quantity of syntactic and semantic knowledge in a *systematic* and *consistent* way.

Third, the incorporation of understanding mechanism tends to make the system enormously large and complicated. The most optimistic (yet the least possible) result one can expect is to train a translation machine to have the ability of a human translator. But to achieve even a fraction of such goal, the knowledge required is still unimaginably large. In principle, if the knowledge of a human translator can be completely transformed into the knowledge base for an MTS and a good inference mechanism is provided, the MTS should be able to translate as good as a human translator. Even we can do so, the complexity of the inference process may prevent the MTS from providing a translation in a *reasonable short time*. We cannot expect to improve the translation quality with such complicated techniques while retaining the processing speed in a reasonable time when the knowledge base scales up. Furthermore, even with the current state-of-the-art AI techniques, it is still quite doubtful whether a translator's knowledge can be completely transformed into a knowledge base effectively. Therefore, with such approaches, the *computational cost*, *maintenance cost* and cost for *knowledge base construction* will be very high as the system becomes large.

Based on our observation, there are several reasons which make high quality translation not attainable in the current MTS's. First, the lexicon information (possibly including some specific world knowledge) is never sufficient, and it takes enormous time to acquire. Second, the available linguistic knowledge are usually incomplete and inconsistent even for a restricted domain. Fine grain linguistic knowledge can not be exhaustively enumerated, and the firing sequences of such complicated rule system can never be explicitly specified with no conflict. Third, technically, there is no systematic way to acquire the required lexicon information and linguistic knowledge while retaining the consistency.

Therefore, the *knowledge acquisition* problem is a major bottleneck in developing a large MT system. *Acquiring* the vast quantity of fine grain information, *verifying* the correctness of such information, and maintaining the *consistency* between all linguistic knowledge, which usually requires human intervention, take most of the time and the cost in system development. When more and more uncertain knowledge are included, the maintenance of consistency between all knowledge becomes even more difficult.

Since most problems in MT can be resolved or explained at syntactic and semantic levels as shown in our experience, failing to achieve HQMT at the current stage, might be not due to the lack of complicated model, like full blown semantic analysis or text understanding, but due to the lack of *effective* computational model to acquire essential knowledge for translation. Hence, it is not recommended to improve translation quality by increasing the complexity of language model. What is more important is to take advantages of available linguistic knowledge to construct more effective computational models, which can be used to acquire a large amount

of syntactic and semantic knowledge in a systematic and effective way. (The statistics-based methods to be discussed in the next section might be one such candidate.)

However, it is possible to develop a knowledge-based MT for a rather restricted domain. The reason is that in a restricted domain the linguistic phenomena is less complex. Therefore, a small number of parameters (or knowledge) is sufficient to characterize the text in that domain. Thus, knowledge acquisition is less of a problem, and consistency is easier to maintain. But this does not mean that this methodology should be adopted in this case. In fact, conventional MT paradigms can also get good translations under such circumstances. It turns out that the knowledge-based approaches might not take precedence over the conventional paradigms in this case either.

5.3. Statistical Inference as an Alternative

In contrast to the inference mechanisms in rule-based or knowledge-based MT, an old yet strongly supported inference mechanism is the statistical approach. Statistics-based approaches have been applied in many fields, such as statistical pattern recognition and speech recognition. The application of statistical approaches to natural language processing also increases rapidly. Emerging studies on statistics-based approaches have a wide coverage ranging from lexical analysis (Garside 1987, DeRose 1988, Church 1989), syntactic analysis (Baker 1979, Fujisaki 1984, Sampson 1986, Garside 1987, Church 1988, Su 1988, 89, 90a), specialized semantic analysis (Church 1989, Liu 1989a, 90) to generalized probabilistic semantic model (Chang 1990, Su 1990b). Experimental systems (Brown 1988) and miscellaneous topics were also studied in various aspects.

As mentioned previously, knowledge *acquisition* and *maintenance* are the major bottlenecks in constructing a large-scale system. We need a systematic approach to acquire knowledge which can be consistently controlled for judging the preference of various interpretations while keeping the knowledge base manageable. The rule-based approaches and KBMT's are somewhat awkward in this regard. In fact, the acquisition of *rules* and *concepts* is still a very hard task, and a lot of human intervention is inevitable. Moreover, the knowledge base tends to become inconsistent when the system scales up. Hence, KBMT's are very costly to scale up. On the other hand, the statistical approach has many desirable features. In terms of *cost* for *computation*, *maintenance* and knowledge base *construction*, the statistics-based approaches are superior in the following aspects.

First, the inference mechanism for a statistical approach is very *simple*. In such systems, the preference on any measurable event is determined simply by some statistics obtained from the training corpus. Hence, it is very easy to implement once the required probability entries are acquired through a large sample space. Because no complicated mechanism and knowledge base are required, the *computational cost* of a statistics-based approach is much lower than knowledge-based or rule-based approaches in run-time.

Second, statistic language models are usually *trainable*. The probabilistic parameters of the language model can be *automatically learned* and *tuned*. Hence, the cost for constructing the knowledge base for a statistics-based system is low. This is probably the most attractive feature of the statistical approaches, especially for a large-scale system. Many statistics-based

approaches, for example, the Hidden Markov Model (Rabiner 1986), have the capability of *self-learning*. This feature makes trainable language model possible (Baker 1979). In addition, by jointly considering all training data in the corpus, the results of parameter learning or tuning will be led to a *global* improvement in performance (in statistics sense). On the contrary, it is hard to learn a *rule* or to write an exact rule for fine grain linguistic phenomena in a rule-based system to guarantee to have global improvement. The verification of such rules is also very difficult.

Third, from our experience, the size of the *lexicon* for a statistic knowledge base is substantially smaller than a rule-based or knowledge-based system. Because most data entries required in a statistics-based model can be represented by a finite dimensional feature vector, and the lexicon entries only need to store simple feature values. Hence, the cost to construct the lexicon is substantially cheaper than the complex-feature-based systems.

Finally, the *maintenance cost* for a statistics-based system is also cheaper than for a knowledge-based system. Because we can provide more systematic way to update the statistic knowledge base when some changes are made. In knowledge-based or rule-based systems, since the rules are subject to exceptions from time to time, the rules and their associated preference measure, must be adjusted frequently to account for such situations. However, there is no systematic and nonheuristic approach to make the adjustment consistently. There is also no assurance that the local adjustment will lead to a better global performance in some *optimal* sense. Therefore, the rule-based system is very sensitive to the changes in the system.

In statistics-based approaches, the maintenance of the knowledge bases are much easier, because they can usually be updated in an *incremental* manner. In (Chang 1988), for example, an incremental approach is proposed to deal with such problem for the score database. The basic idea is to take the incremental change of the grammar as input, compute its side effects on the well-formed analyses which have been stored in the system database, and update the score database accordingly. Because the statistics-based score function is well-formulated, we can well *predict* the incremental changes of the database. Therefore, it is possible to maintain a large database and the scoring mechanism in an easy and *systematic* way. This may not be easy for a rule-based or knowledge-based MT, because the side effects of the change over the knowledge base or rule base is usually *unpredictable*. Furthermore, human power is still the most precious yet most *costly* resource in the development of an MTS, with statistics-based approaches, excessive intervention of the linguists in tuning the system can be avoided. The cost for *maintenance* is therefore much lower.

As long as *performance* is considered, the statistics-based approaches also have some desirable features over knowledge-based or rule-based approaches. First, the assignment of the *preference* measure for competing interpretations is more *objective* in the statistic approaches. Because all these measurements are acquired through the real distribution of the linguistic phenomena. On the contrary, the rule-based approach assigns the preference measure, e.g., certainty factors, in somewhat *heuristic* manner. Hence, we will get a more *consistent* knowledge base with a statistics-based approach. For example, in a statistics-based system, if a hypothesis is said to be probable to the degree of 0.8, then probably 8000 out of every 10000 instances will satisfy the hypothesis. Hence, we have a good interpretation for the preference measure in a *statistics sense*. Such statistics sense enables us to *combine* various hypotheses in nonheuristic manner. On the other hand, if a rule-based system assigns a certainty factor of 0.8 on the

same hypothesis, we can only interpret such preference by stating that a linguist *believes* that the hypothesis is 0.8 probable *in a heuristic sense*. Such preference is hard to interpret or to combine with other hypotheses. Furthermore, different linguists may assign different preference to the same hypothesis according to their expertises. Even the same linguist might assign different preference to the same hypothesis at different time. Therefore, preference assignment might never be measured on the same scale for different people at different time.

Second, most linguists only care about the linguistic phenomena which are theoretically interesting. However, phenomena which are not theoretically interesting may turn out to be practically important. Therefore, we can not and should not expect to get every details from the linguists. On the contrary, the statistical approaches will notice every fine grain "rules" observed from their sample space. Hence, they serve well to fill the knowledge-gaps not interesting to the linguists.

There are indeed some problems with a statistics-based system. The first problem is the requirement of a large database to guarantee the reliability of the system. This is especially important if the language model in the formulation is complicated. However, due to the increasing availability of large tagged or untagged text corpus, acquiring linguistic knowledge from large corpus is no longer a difficult task (Garside 1987). In addition, it takes a substantial time to add *tags* (either syntactic tags, semantic tags or tags representing the relationships between constituents) to an untagged corpus. However, with the aids of computer tools, the tagging process can be done in a quite systematic and effective way. In many cases, the tagging process can be done semi-automatically with minimal human intervention. In fact, adding a tag is much cheaper than building a complex lexicon entry. Furthermore, once the tagged corpus is constructed, various models can be trained base on the same corpus. The training process can also be done "off-line" without affecting the run-time performance of the model.

The second problem with statistics-based approaches is their poor capability in explaining the linguistic phenomena missing from the text corpus. In other words, the statistical approaches are poor in terms of their *generalization* capability, because all knowledge is acquired from the corpus only. Therefore, the system might not well predict other phenomena outside its sample space. Fortunately, the capability to unveil the knowledge of the underlying language model can be improved by enlarging the text corpus. Statistic approaches to reduce the estimation error for sparse data (Fienberg 1972, Efron 1979, Jelinek 1980, Nádas 1985, Su 1989b) are abundant in statistics and speech recognition communities. In addition, algorithms for adaptively adjusting the probabilistic parameters (Bahl 1988) are also available. With so many theoretical supports, it is much easier to acquire the *parameters* in a statistics-based approach than to acquire *rules* or abstract *concept* in a knowledge-based system.

5.4. Cooperative Approach

Nevertheless, it is not recommended to adopt a fully statistics-based MTS, as what is adopted in (Brown 1990), without taking advantages of readily available linguistics knowledge, because many useful information will be lost. The statistics-based models actually should be regarded as the systematic approaches for acquiring consistent *fine grain* linguistic knowledge. If there are indeed well-justified linguistic rules, the statistical approach can never learn better beyond such

rules. Hence, a better solution should be a *cooperative* approach in integrating general linguistics knowledge and statistic methods. To summarize, because of the attractive features of statistics-based approaches in acquiring consistent fine grain knowledge and the generalization capability of rule-based or knowledge-based MT, the following cooperative method is more desirable :

1. General rules which are well justified should be applied if they are proved to be significant for the linguistic problems.

2. For problems which involve complicated and implicit fine grain "rules", linguistic knowledge should be used to identify the most important factors involved so that appropriate probabilistic computational models can be developed.

3. For fast convergence of the training process, computational linguists can provide appropriate prior information as the initial value for the model.

4. The large number of fine grain "rules" or parameters for the models are then left to the statistics-based mechanisms.

The philosophy behind this cooperative approach is to benefit from both the *generalization* capability of the linguists and the *simple*, *objective*, *trainable* and *consistent* characteristics of the statistics-based approaches. In other words, we want to take advantages of the specialities of both human and computer. Generally speaking, human is competent in inducing abstract concepts, such as language model and linguistic rules from observations. But, it is not easy for a linguist to consistently handle massive data. In contrast, a computer can handle massive data and complicated formula in a very short time. But it is very awkward for a computer to learn a rule or to create a model. Hence, it would be wrong to force a computer to mimic the inference process of the human being, because the architecture of a computer is so different from the human brain. Nor should we force a human to do the work that can be done better by a computer. As an example, the early airplanes were designed with wings that flaps as a bird does. But it is well-known that this is not an effective way for a machine to fly. So we should take advantages of computers and the linguists differently according to their special competence. First, let the human design an effective and efficient computational language model, which are characterized by a set of parameters, and then let the computers acquire these parameters from the training corpus. Afterwards, these parameters are maintained and manipulated by the computer in a systematic and consistent way. Our experiences in a series of studies on the score function paradigm as mentioned previously have shown encouraging results by adopting such a cooperative approach.

6. INTEGRATION OF THE KNOWLEDGE SOURCES

After the individual design issues are inspected, we are in a position to consider the strategies for integrating various knowledge sources, including lexical, syntactic and semantic knowledge, in the system. In the course of integration, several trade-offs must be made.

First, the compromise between the *expressiveness* and the *uniformity* of the system must be taken into consideration. By expressiveness, we are concerned with the ability of the system to support various linguistic formalisms and various levels of analysis, and its ability to express

the required knowledge in an efficient and direct way (Shieber 1985). By uniformity, we mean whether the system has an uniform data structure and an uniform control to solve the problems.

In an MTS, it is desirable to have a high degree of expressiveness so that we can handle different problems at different levels without any difficulty. To achieve this goal, various driving mechanism should be provided. For regular phenomena, it is desirable to have a *rule-driven* facility so as to define and solve the problem in a formal and declarative way. For problems that are highly dependent on the individual characteristics of the words, a *lexicon-driven* facility is probably more favorable. As far as irregular phenomena are concerned, a *procedure-driven* approach may take precedence over others. But a system with so many driving mechanisms would loss its uniformity and make it hard to design, debug and maintain. Since most efforts in designing an MTS is devoted to incorporating the linguistic knowledge into the system, we would prefer to have a high degree of expressiveness by providing several different driving mechanisms at a higher level, and leave every implementation details in low level to computer engineers.

Second, we must consider the compromise between the *integrating capability* and *modularity* of the system. By integrating capability, we mean that each module (e.g., lexicon, syntax and semantics) can interact with each other in a direct and closely coupled way. By modularity, we mean that each module is separated from each other with minimal interference. In parsing an artificial language, the lexical, syntactic and semantic modules can be isolated into separate modules which can then be called without any interaction. This is also the operation mode of early MTS's. However, it is desirable to allow the lexical, syntactic and semantic modules of the system to take advantages of each other in the course of natural language processing. For example, the syntactic module (parser) may make use of the lexical information from the lexical module to handle some lexicon-specific events. The lexical module, on the other hand, may take advantages of the syntactic module to identify the existence or absence of some *nonsentential* constituents for special preprocessing tasks. In addition, a semantic module may identify some attributes, for example, the case role, from some fixed syntactic pattern of a sentence, and the syntactic modules may determine the grammatical relationship between two constituents by consulting the semantic module. Such interaction is necessary for a complicated system like an MTS, because human being tends to use the information from all these three types of analyses when parsing a sentence with a high degree of interaction and parallelism (Kay 1990). Thus a good working model with high degree of interaction might look like Figure 6-1. The functional modules communicate through the global data structure, called a *blackboard* (Erman 1980), and each modules can take advantages of all the others. (The blackboard may contains the input buffer, the annotated parse tree, and so on.) In addition, different driving mechanisms are provided to handle linguistic problems at different levels.

However, the high degree of interaction in functional modules may cause the modularity of the real programming modules to degrade substantially because each main module has to take care of the communication with others and side effects introduced by the other module. One way to relief this problem is to limit the access privilege of individual modules to the global data structure. The individual modules are allowed to access the whole global data, but are allowed to change only a fraction of it, so that the dependency between each module can be minimized. By providing such degree of flexibility, the system should be powerful enough to

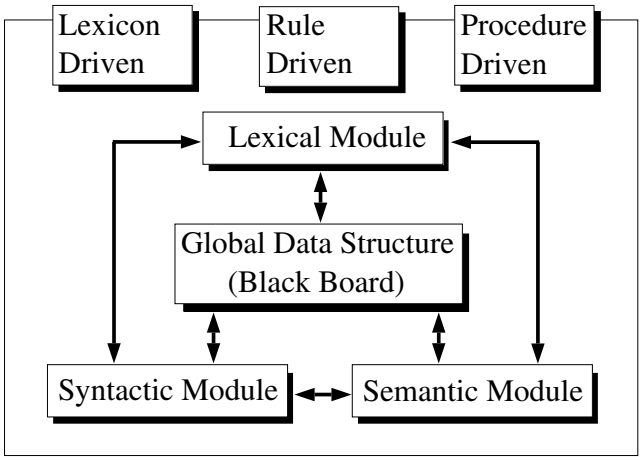


Figure 6-1. Operating Model of the Functional Modules

integrate a wide variety of knowledge sources encounted in MTS while relieving the potential maintenance problems of the system.

7. CONCLUSION

In this paper, some design issues about Machine Translation Systems are studied based on our experience in developing the **ArchTran** MTS. Recommendations are given to help make decisions on different trade-offs. Attempts are made to characterize an MTS in terms of several basic characteristics: the *representation*, *manipulation*, *acquisition* and *integration* of various knowledge sources. We summarize the main points of view as follows.

The formal power of the knowledge representation is evaluated in terms of two factors : the linguistic capacity is measured in terms of the *generative power* and *expressive power* of the grammar, and computational capacity is evaluated in terms of the grammar *type*. As far as generative power is concerned, a grammar with a little over-generative power than is needed for the target domain is recommanded so as to cover most of the syntactic structures of concern. However, the fraction of overgeneration should be well constrained to achieve the required expressive power.

Many contemporary grammar formalisms, including some unification-based grammars, are essentially based on context free grammar (CFG). Efficient parsing algorithms have been well developed for such grammar. It can also be easily augmented for semantic analysis to achieve the formal power of a *Turing Machine*. Therefore, it is appropriate as the base grammar in favor of other grammar types.

With grammar splitting, the generative power of the grammar is not changed. But the system is enforced with the capability of *partial parsing*, which makes some special treatment on *nonsentential* constituents more easy. It also makes top-down filtering easy to implement. Experimental results show that grammar splitting tends to reduce the number of conflict actions in the parsing table. Hence, it is recommended for a practical system.

It is not always easy or effective to deal with some special phenomena in terms of regular grammar rules. Two approaches can be adopted to relieve the parser from being over-loaded. First, a preprocessing unit can be provided to convert the input strings into *normalized* token stream for the parser. Second, fail-soft *error recovery* strategy should be provided to deal with extragrammatical input, which constitute a substantial fraction of the input in real text.

The manipulation and control of the knowledge is affected significantly by the *parsing strategy* of the system. The parsing strategy can be evaluated in terms of its *efficiency*, *flexibility* for conducting *semantic analysis*, and capability for *preference* assignment. On the one hand, the top-down approach provides a natural way to describe semantic actions, it also make filtering of large constituents possible. On the other hand, the bottom-up approach is inherently more efficient for many natural languages. The composition of semantic features of smaller constituents, which is a common operation in unification-based grammars, is also more easy to implement in bottom-up mannaer. Hence, a mixed parsing strategy, such as *bottom-up parsing with top-down filtering* (TD+BU), is considered to be superior in comparison with pure bottom-up or pure top-down approaches.

The parsing efficiency can be improved in three major ways, that is, *data-driven*, *structure sharing* and *scored truncation*. Data-driven can be easily implemented by adopting a bottom-up or mixed type parsing strategy. To prevent the parser from duplicated parsing, it is suggested that some structure sharing mechanism, such as conventional *chart* or the dynamic phrase table (DPT) adopted in ArchTran, be used to store the partial parse trees which have been constructed. State merging, with shared stacks and trees, is also a valid approach to improve parsing efficiency by preventing repeated traversal of the same state space. It is shown that the efficiency of the parser is greatly improved when a chart-like mechanism is used. To further improve the parsing efficiency, the *scored truncation* algorithm is highly recommanded for cutting down the searching space while retaining the preferred analysis. Such approach greatly reduces the searching space, and provides a tuning factor for speed-quality coordination.

Conventional approach to semantic analysis is to impose semantic constrains on each syntactic constructions. When any constraint is violated, the analysis will be rejected. Such *hard-rejection* approaches provide little help in resolving *multiple competing interpretations*. For *ill-formed* yet *acceptable* constructions, the hard-rejection approaches also fail to get reasonable analyses. Because *ill-formedness* and *inherent ambiguity* are common in a practical system, an objective scoring mechanism, such as the *score function*, is required to assign preference measure to each competing interpretations or ill-formed constructions. With such *soft-rejection* mechanism, the grammar rules and associated constraints can be maintained more easily by allowing uncertainty in the rules.

In the recent years, knowledge-based MT and full blown semantic analysis have been proposed in an attempt to achieve high quality translation. Although text understanding might be helpful to improve the translation quality in a restricted domain by increasing modeling complexity, many other factors, such as low coverage in unseen text, might have more significant influence for a practical system. In particular, our experience shows that most translation problems in technical manuals can be resolved or at least explained at the syntactic and semantic level in terms of general linguistic knowledge. This means that the *acquisition* of *consistent* linguistic knowledge which can be used to explain and predict linguistic phenomena in the

target domain is probably a more important issue. Although the knowledge-based approaches provide more complicated language models to improve the translation quality in a subdomain, they actually do not resolve the basic problems in translating unseen text. The high *cost* in building the knowledge bases and the difficulty to *maintain* a large and consistent knowledge base in KBMT will also make it impractical to be used in large scale applications in terms of *cost-effectiveness*.

To acquire the vast quantity of linguistic knowledge in a systematic way, a cooperative model is proposed. It is suggested that the computational linguists take advantages of their generalization capability in linguistic knowledge to construct probabilistic language models, and leave the parameters or fine grain knowledge of the model to a corpus-based statistics-oriented training mechanism.

Because various knowledge sources will be used to resolve the linguistic problems encountered in MTS, a policy for integrating such knowledge sources is necessary. Since a high degree of expressiveness is desirable to an MT system, we recommend that different driving mechanisms be provided so that different linguistic problems can be solved at appropriate levels. It is also suggested to allow the functional modules of the system to interact with each other so that each module can take advantages of one another. However, the access privileges across functional modules should be limited to maintain modularity at the same time.

Acknowledgements

We would like to thank the Behavior Tech. Corp. for its full financial support for this project and to the members of the MT research team at the BTC R&D center for their technical support.

REFERENCES

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

Aho, A.V., R. Sethi and J.D. Ullman. 1986. **Compilers : Principles, Techniques, and Tools.** Reading, MA : Addison-Wesley.

Bahl, L.R., P.F. Brown, P.V. de Souza, and R.L. Mercer. 1988. A New Algorithm for the Estimation of Hidden Markov Model Parameters. Proceedings of the 1988 IEEE International Conference on Acoustics, Speech, and Signal Processing. 493-496.

Baker, J.K. 1979. Trainable Grammars for Speech Recognition. Proceedings of the Spring Conference of the Acoustical Society of America.

Bate, M. 1978. The Theory and Practice of Augmented Transition Network Grammars. In : L. Bolc (ed.), **Natural Language Communication with Computer**. Springer Verlag, 191-259.

Bennett, W.S. and Jonathan Slocum. 1985. The LRC Machine Translation System. *Computational Linguistics*, **11**, No. 2-3 : 111-119.

Bennett, W.S. 1990. How Much Semantics is Necessary for MT Systems ? Proceedings of the Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language. University of Texas at Austin, USA.

Brown, P., J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. 1988. A Statistical Approach to Language Translation. Proceedings of COLING-88. Budapest, Hungary, **1** : 71 -76.

Brown, P., J. Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, F. Jelinek, J. D. Lafferty, R.L. Mercer, and P.S. Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, **16**, No. 2 : 79-85.

Caeyers, H. and G. Adriaens. 1990. Efficient Parsing Using Preferences. Proceedings of the Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language. University of Texas at Austin, USA.

Carbonell, J.G. and M. Tomita. 1987. Knowledge-based machine translation, the CMU approach. In S. Nirenburg (ed.), **Machine Translation - Theoretical and Methodological Issues**. New York, NY : Cambridge University Press, 68-89.

Chang, C.-L. and K.-Y. Su. 1987. A New Mechanism of Transfer for An English-Chinese MT System. Conference on Translation Today, Hong Kong.

Chang, J.-S. and K.-Y. Su. 1988. Incremental Environment for Scored Machine Translation Systems. Proceedings of ROCLING-1. Nantou, Taiwan, R.O.C., 63-73.

Chang, J.-S. 1990. GPSM : A Generalized Probabilistic Semantic Model for Ambiguity Resolution. Master Thesis, Institute of Computer Science and Information Engineering, National Chaio Tung University, Hsinchu, Taiwan, R.O.C.

Church, K. 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. ACL Proceedings of 2nd Conference on Applied Natural Language Processing. Austin, Texas, USA, 136-143.

Church, K. and P. Hanks 1989. Word Association Norms, Mutual Information, and Lexicography. Proceedings of 27th Annual Meeting of the ACL University of British Columbia, Vancouver, British Columbia, Canada, 76-83.

DeRose, S. J. 1988. Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics*, **14**, No. 1 : 31-39.

Earley, J. 1970. An Efficient Context-Free Parsing Algorithm. *CACM*, **13**, No. 2 : 94-102.

Efron, B. 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, **7**, No. 1 : 1–26.

Erman, L.D. and V.R. Lesser. 1980. The HEARSAY-II Speech Understanding System : A Tutorial. In W.A. Lea (ed.), **Trends in Speech Recognition**. Englewood, NJ : Prentice-Hall.

Fienberg, S.E. and P.W. Holland. 1972. On the Choice of Flattening Constants for Estimating Multinominal Probabilities. *Journal of Multivariate Analysis*, **2** : 127–134.

Fujisaki, T. 1984. A Stochastic Approach to Sentence Parsing. Proceedings of COLING-84. Stanford Univ., CA, 16-19.

Garside, R., G. Leech and G. Sampson, 1987. **The Computational Analysis of English : a Corpus-Based Approach.** London : Longman.

Hopcroft, J.E. and J.D. Ullman. 1979. **Introduction to Automata Theory, Language, and Computation.** Reading, MA : Addison-Wesley.

Horowitz, E. and S. Sahni. 1978. **Fundamentals of Computer Algorithms.** Computer Science Press Inc..

Hsu, H.-H. and K.-Y. Su. 1986. A Bottom-Up Parser in The Machine Translation System with The Essence of ATN. Proceedings of International Computer Symposium (ICS) 1986, Tainan, Taiwan, R.O.C., **1 of 3** : 166-173.

Hutchins, W.J. 1986. **Machine Translation : Past, Present, Future.** West Sussex, England : Ellis Horwood Limited.

Ingria, Robert J.P. 1990. The Limits of Unification. Proceedings of 28th Annual Meetings of ACL, Pittsburgh, PA, USA. 194–204.

Jelinek, F. and R.L. Mercer. 1980. Interpolated Estimation of Markov Source Parameters from Sparse Data. In: E.S. Gelsema and L.N. Kanal (eds.), **Pattern Recognition in Practice.** Amsterdam, The Netherlands: North-Holland Publishing Company, 381–397

Kay, M. 1980. Algorithm Schemata and Data Structures in Syntactic Processing. CSL-80–12, Xerox Corporation.

Kay, M. 1985. Parsing in Functional Unification Grammar, In : D.R. Dowty, L. Kartunnen and A. Zwicky (eds.), **Natural Language Parsing : Psychological, Computational and Theoretical Perspectives.** Cambridge, England : Cambridge University Press, 251–278.

Kay, M. 1990. On the Architecture of a Machine Translation System. Proceedings of ROCLING-III, Taipei, R.O.C.

Liu, C.-L. 1989a. On the Resolution of English PP Attachment Problem with a Probabilistic Semantic Model. Master Thesis, Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan, R.O.C.

Liu, C.-L. and K.–Y. Su. 1989b. A Quantitative Comparison Between an LR Parser and an ATN Interpreter. Proceedings of ROCLING-II, Nantou, Taiwan, R.O.C., 277–288.

Liu, C.-L, J.-S. Chang and K.-Y. Su. 1990. The Semantic Score Approach to the Disambiguation of PP Attachment Problem. Proceedings of ROCLING-III, Taipei, R.O.C., 253–270.

Marcus, M.P. 1980. **A Theory of Syntactic Recognition for Natural Language.** London, England : MIT Press.

Nádas, A. 1985. On Turing's Formula for Word Probabilities. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, **ASSP-33**, No. 6 : 1414-1416.

Pratt, V.R. 1973. A linguistics oriented programming language. Proceedings of 3rd IJCAI, Stanford Univ., CA, 372-381.

Pratt, V.R. 1975. LINGOL—A progress report. Proceedings of 4th IJCAI, Tbilisi, Georgia, USSR, 422-428.

Rabiner, L.R. and B.H. Juang. 1986. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, Jan. 1986 : 4-16.

Rich, E. 1983. **Artificial Intelligence**, New York, NY : McGraw-Hill.

Sampson, G. 1986. A Stochastic Approach to Parsing. Proceedings of COLING-86, Univ. Bonn, 151-155.

Sells, P. 1985. Lectures on Contemporary Syntactic Theories. CSLI Lecture Notes, Stanford University.

Shieber, S.M. 1985. Criteria for designing computer facilities for linguistic analysis. *Linguistics*, **23**, 189-211.

Slocum, J. 1986. Machine Translation : An American Perspective. *Proceedings of the IEEE : Special Issue on Natural Language Processing*, **74**, No. 7 : 969 - 978.

Su, K.-Y., H.-H. Hsu and M.-L. Hsiao. 1985. Development of an English-To-Chinese Machine Translation System. Proceedings of Natl. Computer Symposium (NCS) 1985, Kaohsiung, Taiwan, R.O.C., **2**, 997-1001.

Su, K.-Y., J.-S. Chang, and H.-H. Hsu. 1987a. A Powerful Language Processing System for English-Chinese Machine Translation. 1987 Int. Conf. on Chinese and Oriental Language Computing, Chicago, Illinois, 260-264.

Su, K.-Y., J.-N. Wang, W.-H. Li and J.-S. Chang. 1987b. A New Parsing Strategy in Natural Language Processing Based on the Truncation Algorithm. Proceedings of Natl. Computer Symposium (NCS) 1987, National Taiwan University, Taipei, R.O.C., **2** : 580-586.

Su, K.-Y., C.-L. Chang and S.-C. Chen. 1987c. Hands-On Experiences in the Development of An English-Chinese Machine Translation System. Conf. on Translation Today, Hong Kong.

Su, K.-Y. and J.-S. Chang. 1988. Semantic and Syntactic Aspects of Score Function. Proceedings of COLING-88, Budapest, Hungary, **2** : 642-644.

Su, K.-Y, J.-N. Wang, M.-H. Su and J.-S. Chang. 1989a. A Sequential Truncation Parsing Algorithm Based on the Score Function. Proceedings of International Workshop on Parsing Technologies, CMU, Pittsburgh, PA, U.S.A., 95-104.

Su, K.-Y., M.-H. Su and L.-M. Kuan. 1989b. Smoothing Statistic Databases in a Machine Translation System. Proceedings of ROCLING-II, Nantou, Taiwan, R.O.C., 333–347.

Su, K.-Y., J.-N. Wang, M.-H. Su and J.-S. Chang. 1990a. GLR Parsing with Scoring. to appear In : M. Tomita (ed.), **Generalized LR Parsing.**

Su, K.-Y., J.-S. Chang and Y.-C. Lin. 1990b. A Unified Approach to Disambiguation Using A Uniform Formulation of Probabilistic Score Functions. in preparation.

Tomita, M. 1987. An Efficient Augmented-Context-Free Parsing Algorithm. *Computational Linguistics*, **13**, No. 1-2 : 31-46.

Wilks, Y. A. 1975a. Preference Semantics. In Keenan, Edward Louis, III (ed.), **Formal Semantics of Natural Language.**Cambridge University Press.

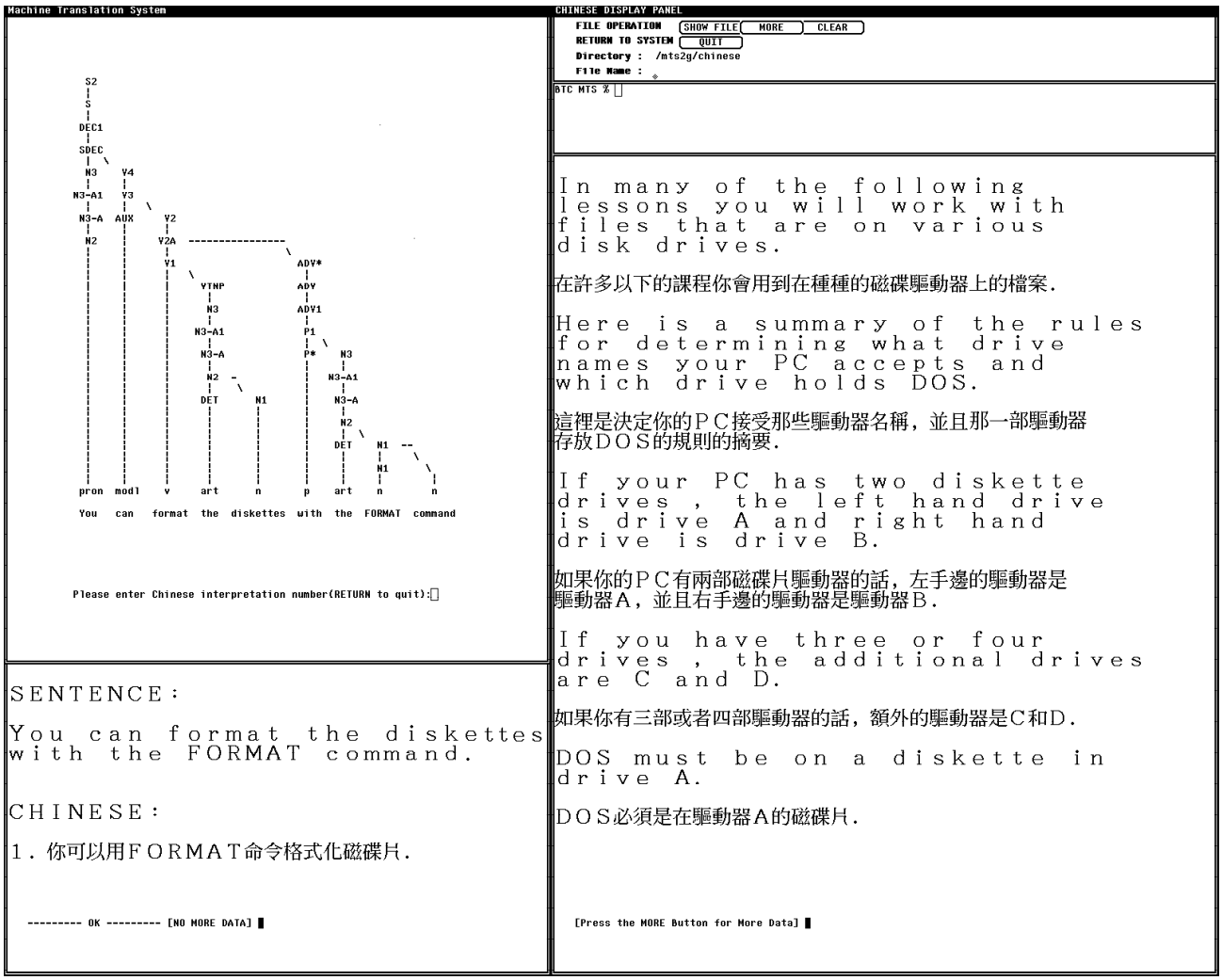
Wilks, Y. A. 1975b. An Intelligent Analyzer and Understander of English. *CACM*, **18**, No. 5 : 264-274.

Wilks, Y.A. 1983. Preference Semantics, Ill-Formedness, and Metaphor. *AJCL*, **9**, No. 3-4 : 178-187.

Winograd, T. 1983. **Language as a Cognitive Process.** Vol. 1. Reading, MA : Addison-Wesley.

Woods, W.A. 1970a. Context-Sensitive Parsing. *CACM*, **13**, No. 7 : 437-445.

Woods, W.A. 1970b. Transition Network Grammars for Natural Language Analysis. *CACM*, **13**, No. 10 : 591–606.



APPENDIX : The ArchTran English-Chinese Machine Translation System