INCREMENTAL ENVIRONMENT FOR SCORED MACHINE TRANSLATION SYSTEMS

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

[*]BTC R&D Center
2F, No. 28, R&D Road II
Science-Based Industrial Park
Hsinchu, Taiwan, R.O.C.

[**]Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.

ABSTRACT

In a Machine Translation System a wide variety of data and rule bases are involved. Some of these data bases must be constructed, trained or learned from the *well-formed* output of the system. Such data bases are sensitive to the changes in the other data bases in the system, and the reconstruction of these data bases will be time-consuming if everything is reconstructed from scratch.

In this paper, we propose an *incremental* strategy to this problem. The *basic operations* in constructing an incremental interface are outlined and the associated topics are discussed. An illustrative example will be given to show the basic methodology behind the philosophy of incrementality. Some general consideration and extension about incremental approach will be discussed as well. The same strategy can be extended to handle large data base which is subject to change in a much more efficient way.

With an incremental approach, the data bases which are sensitive to changes can be reconstructed by making only *local change* to the original ones. Thus, the *time and cost* for reconstruction will be reduced significantly.

Introduction

In a Machine Translation System (MTS), a wide variety of data and rule bases ("data bases" for short) are involved. Typical data bases are the *underlying grammar* of the system (possibly augmented), *transformation rules*, *generation rules*, knowledge-based *heuristic rules* and the *lexicon information* for the lexicon items. They serve as the basis for driving the parsing mechanism and other analysis modules. All these data bases are constructed by linguists to solve most of the linguistic problems that can be generalized.

However, the linguists can not supply every *subtle* and *detail* information for the system. For instance, the detailed co-occurrence restrictions on all combinations of the syntactic categories may not be completely or explicitly described in the rules proposed by the linguists. Furthermore, in many cases the preference for different interpretations in language processing involves *uncertainty* such that even a distinguished linguists may have difficulty in providing a precise decision rule. Determination of the preference for these cases may requires *statistic* measurement instead of *subjective* judgement from linguists. Therefore, tuning the linguist-constructed data bases is usually essential to the system.

One way to tune the linguist-constructed data bases is to introduce other components capable of providing default information induced from the *previous behavior* of the system. This will create another type of data bases which are based on the *well-formed output* of the system. Data bases of this type can be used as the *feedback* to the system for *self-learning* or as the *statistic* data bases for decision making. Supplementary information for tuning linguists' rules can also be acquired from these data bases.

The **scored parsing mechanism** proposed by [Su 87], for example, uses the well-formed syntax trees observed to take part in the **score function** [Su 88]. The score function is then used for truncating potentially inappropriate parsing paths *during* scored parsing, or for computing relative preference to the ambiguous parse trees *after* parsing. Under the scoring mechanism proposed, it can be shown that the preference measurement (the score) for ambiguous parse trees, which are semantically annotated, can be divided into *syntactic* and *semantic* components in a *statistic* sense. While the **semantic score** is reflected in the lexicon information, in the augmented part of the underlying grammar, in the heuristic rules and in the associated *subjective* score assigned to them, the **syntactic score** corresponding to the syntactic component of the score function can be extracted from the well-formed syntax trees observed. A typical system involving such data base can be modeled as in Figure 1.

This model closely resembles a closed-loop control system with the **Learner** and the **Expert** modules as the feedback components. The Expert Module corresponds to the linguists (*Linguistic Experts*), who specify the grammar, linguistic knowledge, lexicon information and heuristic rules based on their expertise. Because the Expert is likely to give *inexact* and *incomplete* rules to the system, *scored parsing*, involving the management of *uncertainty*, is required for evaluating the preference to different parsing paths or the annotated trees associated with the paths.

In such a system the Learner plays an important role in tuning the expertise of the linguists. It learns the preferred syntactic construction, extracts the information required for scored parsing from the well-formed output syntax trees and constructs a special data base which is used for tuning. We shall call this data base a **Self-Adaptive Data Base (SADB)**. Probability is associated with the extracted information in this data base as a measurement

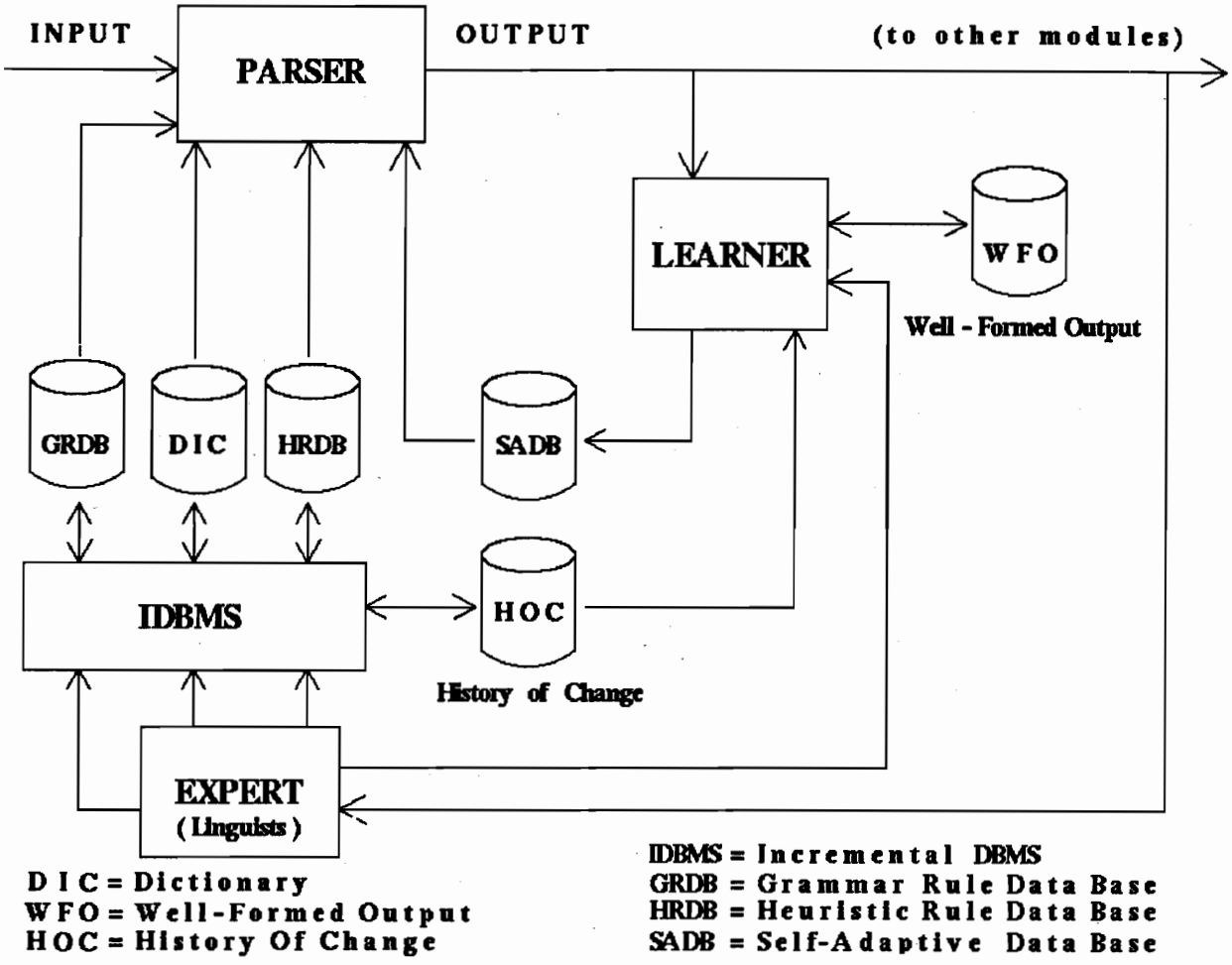


Figure 1 The Data Bases in a Scored MTS with Incremental Interface

of preference. The preference not explicitly specified over different syntactic structures is then embedded in this data base in a *statistic* sense. In general a linguist would not specify such data base entries explicitly in their rules, but they are required for the system to perform **default reasoning** when expert-proposed rules are not available. They are also required when competing interpretations are encountered such that these interpretations can be distinguished only in statistic sense. Note that we have used three typical data bases to model expert-constructed data bases just for simplicity. In a practical system, the number of data bases should be much larger.

Motivation for an Incremental Interface

The special type of data base just mentioned may introduce a problem when the other related data bases are changed. This problem results from the ever-changing characteristics of the data bases in an MTS. For a Machine Translation System, since the source language to be handled is invariant as well as the parsing mechanism, the feedback components must be changed gradually so as to obtain a stable system. However, the changes in the *expert-constructed* data bases, especially that of the *underlying grammar* will change the output for

the original input which is used to construct the SADB. Because the expert-constructed data bases are subject to changes from time to time, any data base, such as the SADB, that is output-dependent must be reconstructed each time the other related data bases are changed. (The data bases which cause the output for the same input to be changed between two data base versions will be called the *change-inducing data bases* hereafter.)

To reconstruct the SADB when the other data bases are changed, two approaches can be adopted. The first approach is to reconstruct the data base *from scratch*; that is, the associated input of the outdated output is *re-parsed*, the correct syntax trees are *picked out* (by linguistic experts), and the required information is *extracted* from the new well-formed output. After the tedious procedures, the new data base is *reconstructed.* This process is quite *time-consuming* because *human intervention* is required. In addition, a lot of *computing power* is wasted merely for re-computation. Moreover, the old data base becomes useless once the other related data bases are changed, no matter how trivial the change may be. Therefore this approach is impractical for a large-scaled system with large data bases.

An alternative solution to this process is to adopt an *incremental interface* between the system output and the Learner module. The Learner can be switched between two modes, the normal mode and the incremental mode. In the *normal* mode, the Learner takes the set of output which is generated for the *first* time as its input, and constructs the required data base under the directions of the linguists. In the *incremental* mode, existing well-formed output which is previously used to construct the SADB is *patched* according to the *history of change* (**HOC**) of the grammar. Under such circumstance, the reconstruction of the data base will be much more efficient because only *local* patch is performed. This methodology is termed *incremental* because all one has to do is to take the *incremental change* of the change-inducing data base into account and reshape *part of* the outdated output and data base without restarting from scratch.

Basic Operations of an Incremental Interface

To construct an incremental interface, one must define a set of *primitive operations* to represent any possible change in the change-inducing data base before the incremental interface is set up. The incremental approach then involves the execution of the following basic operations when a local change is made in the change-inducing data base.

1. Compute the *incremental change* between the two versions of the change-inducing data base.
2. Identify the sequence of the *primative operations* involved in the change.
3. Compute *the effects of the change* on the *output.*
4. Find *the range to be patched* in the original output.
5. *Patch* the original output according to the predicted effects of the change and *reconstruct* the required data base.

For example, when the underlying grammar is changed, the syntax trees previously recognized as well-formed may change their syntax structures under the new grammar. To reflect the change and patch the old output, we must first compute the difference between the two versions of the grammar. The difference may show that the change is due to *addition, deletion or renaming of symbols.* In this case, the *substitution* of a new symbol for the old one is sufficient to patch up the original output. The change may also result from *addition or*

deletion of the **Phrase Structure (PS) Rules**. Such change may cause part of the **state space** (or **transition network**) of the parsing mechanism to be changed. Under this circumstance, the incremental change in the state space is predicted, the original output is inspected to see *whether it falls within the altered states*, the range to be patched is then searched if necessary, and the structure in the range is patched accordingly.

In general, the effect of the grammar change will only be *local*. They can be precomputed, under certain type of **closure operation**, from the change in the grammar and a set of associated productions. Therefore, the effort to reconstruct the data base is no more than patching up the portion which has been changed. Time-saving is thus expected if the incremental approach is adopted.

An Example

To show the above idea more explicitly, consider two different versions of the underlying grammars named G and G′, respectively.

G :	**G' :**	**Difference (Delta-G)**
S -> NP VP	S -> NP VP	G : NP -> DET n
NP -> DET n	NP -> det n	DET -> det
DET -> det	VP -> v NP	
VP -> v NP		G' : NP -> det n

In this example, the change in the grammar is simply the reformulation of the NP (Noun Phrase) part of a sentence (S). This change can be viewed as a deletion of the two productions in G followed by an insertion of a new production in G′ which also appear in Delta-G, the difference between G and G′. (Note, however, this expression of the change in terms of the sequence of insert/delete primitives is not unique.)

The **goto-graphs** [Aho 85] for G and G′ are shown in Figure 2. The graphs consist of a number of **set of items**, each corresponding to an internal state of the parser, and a set of arcs which show the direction of transition for a specific input. The productions in a set of items and the "dots" for the productions indicate the potential position of the input pointer if the parser enter the corresponding state. In some sense, these graphs corresponds to the transition networks in **ATN** Formalism [Woods 70].

These graphs do not show the effect of the individual insertion or deletion explicitly, but the net effect of the grammar change is apparent. In this example, the incremental change in the state space is the substitution of the portion labeled as **NP sub-Grammar** in the goto-graph for G with its counterpart in G′. By examining these two graphs carefully, one can also find that the portion labeled as **NP sub-Grammar** in either graph is closely related to the **closure** of the set of items corresponding to the productions in *Delta-G*. More specifically, the portion deleted and the portion patched (after deletion) can be generated from the corresponding productions in *Delta-G*. This observation confirms our previous intuition that *local change in the underlying state space of the parser can be precomputed from the local change in the underlying grammar incrementally*.

After the incremental change in the state space is patched, we will now examine how incremental approach can be adopted to patch the old syntax tree for reuse. The syntax trees

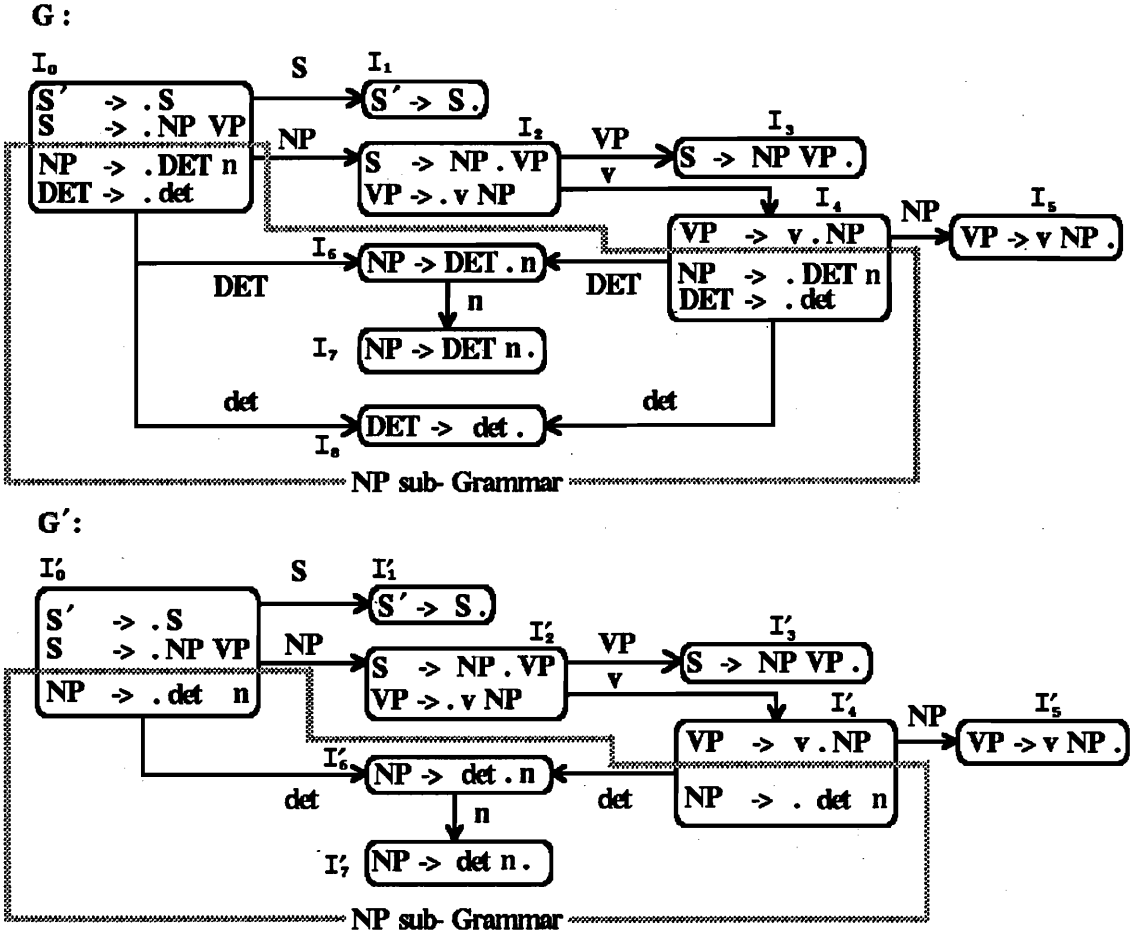


Figure 2 The GOTO- Graphs (Transition Diagrams) for G and G′

in Figure 3 show an instance of transformation from an old syntax tree to a new one, where syntax tree T is for the original grammar G and T′ for G′, respectively.

In Figure 3, the subtrees enclosed by the dashed lines are the parts to be or been patched. To see how the scopes of patching are identified, let's scan the terminal nodes from left to right. At t_0, the arrow in the figure for T is at the position preceding the nonterminals **S**, **NP**, **DET** and the terminal node **det**. If we view the arrow in T as the "dot" in the set of **LR(0) items** in the goto-graph for G, one can find without difficulty that this position corresponds to the set of items I_0. Since this set of items no longer exists in the new goto-graph, we know that this point is the *starting point* of the scope to be patched. At t_1, the state corresponding to the arrow position is also at an old state, namely I_6, not found in the new state space. Hence, it is still in the range to be patched. When the arrow is moved further to the right at t_2, the state is exactly the same as that of I_2. This set of items also appears at the goto-graph for G′ as I'_2. Hence the *end* of patching is found. It is also obvious, from the tree T, that the target to be patched is an **NP**. Scanning further in the same manner, one can find another scope to be patched in T, whose target is also an **NP**.

Keeping this information in mind, the remaining task is simply to reconstruct the **NP** subtrees instead of the whole sentence. This task can be done by traversing the new state space of the new grammar from an appropriate entry, namely I'_0, at the goto-graph for G′. Any parser capable of *partial parsing* should serve well for this purpose.

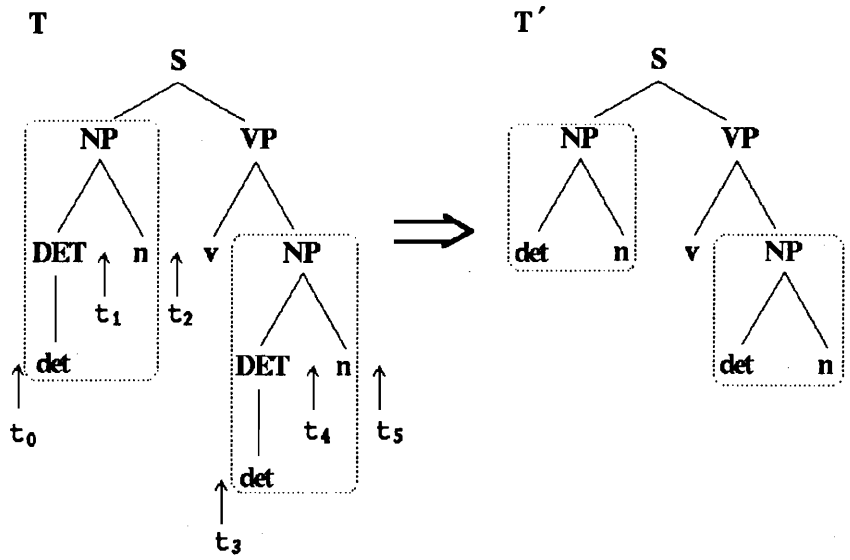


Figure 3 Syntax Trees before and after patching with incremental method

Although we have demonstrate this example in terms of some observations at the state space level, it is by no mean the sole and optimal way to address the methodology behind incremental approaches. Trade-offs for implementation will be left to a later paragraph.

It is important not to get confused with the concept of *incremental parser* [Ghezzi 80] when discussing the *incremental interface* just mentioned although both of them serve as tools to patch syntax trees. An incremental parser is used to patch a parse tree when its *input* (i.e., the *program*) is altered. The underlying *grammar* (or the *state space* of parsing) is kept unchanged. The incremental interface, on the contrary, is used to patch a well-formed syntax tree so that no re-parsing is required when the underlying *grammar* is changed. Therefore, the incremental interface is much more complicated than an incremental parser.

Basic Components in the Incremental Interface

To implement the incremental interface in the previous example, two basic components are required. The first is a **grammar editor** which records the incremental change in the symbols, grammar rules, and so forth. The second component is a **mini-parser** with *partial parsing* capability. It is used to predict the incremental change in the underlying state space and the effect of the grammar change to the parse trees. It is also used to patch up the original syntax trees if required. It is not necessary for the mini-parser to have the full power of the parser of the MTS because it only performs *local* patch. Its function can be a subset of the MTS parser, thus the name *mini-parser*. When possible, it can also be incorporated into the MTS parser and toggled in different modes.

Although we have use the dependency between the GRDB (Grammar Rule Data Base) and SADB as an example to show the idea of incrementality. This idea can be further extend to other data bases. The roles of the GRDB and SADB in the model in Figure 1 can, in fact, be played by any data bases in the model. For example, the deletion of a syntactic category from the dictionary (DIC) may cause some of the productions of the underlying grammar to become *useless*. In this case, an incremental approach can be adopted to modify

the underlying grammar in a similar manner as long as it is cost-effective. The incremental change in the grammar is then handled as mentioned previously.

Hence, for a more general and integrated scheme, the grammar editor is involved in an *incremental data base management system (IDBMS)* which handles the access to the data bases in the system and maintains the history of change of the whole data base system. The components to patch up specific data bases are then incorporated into an incremental interface in a module similar to the Learner module of the more general learning system.

General Consideration

From the previous discussion we can summarize the basic philosophy behind incrementality for maintaining a large data base. The first point of greatest concern for incrementality is to make old data base entries *reusable*, and the second is to reduce the *cost of computation* for the reconstruction of the data base when local change arises.

To implement an efficient incremental interface, a few factors should be pointed out. The first important factor to consider is the selection of the set of *primitive operations* of the grammar change. It is obviously that these primitives must be able to represent or synthesize any sequence of modification to the underlying grammar. In addition, since the selection of the primitives and the representation of the change are not unique, *optimization* on the representation of the change in terms of the primitive operations would be significant when the requirement for efficiency is crucial. For example, we can *permute* the sequence of insertions and deletions in the demonstrative example without changing the result. But the *order* of the sequence of primitives may be a matter of great concern when the incremental change is patched on a *production-by-production* basis. Instead of using the deletion and insertion of a single production as the primitives, one can also take the deletion and insertion of *a group of related productions* as the primitives for incremental change. Closure operation for these productions can be performed to compute the net effect of the total change. The algorithm for patching in this way may be more complicated then the previous one, but the efficiency may be more rewarding. In either case, the IDBMS should provide a user-friendly interface for the linguists to specify the change in a linguistics-oriented manner.

Another important factor to consider is the efficiency of the mini-parser. Many strategies can be adopted to implement the mini-parser. One can *re-parse* the required subtrees for the original data base by making transition between the internal states of the mini-parser each time a syntax tree is read in. An alternative way is to *precompute* the transformation for the subtrees which are subject to change due to the local change of the grammar. The job of patching old syntax trees is then simply a task of *pattern matching*. While the first alternate requires more computation, the second one requires larger storage if the number of transformation is large. The choice is a trade-off between computing time and storage requirement. Thus a third way by mixing these two strategies is also possible. With the mixed strategy one can record the transformation when a new instance of transformation is found while re-parsing a required subtree. On the other hand, the required subtree to be patched can be replaced by the transformed pattern without re-parsing if the transformation has been recored.

Conclusion

In this paper, *incremental* strategy is proposed to resolve the problem of reconstructing the data base used in scored parsing. The algorithm for incremental patching of the data base

is outlined. Fundamental components in the incremental interface are also discussed. We have pointed out a few important factors to considered about the efficiency of the incremental interface. With minor variation, the strategy can be easily extended to any self-adaptive data base which uses the output of the system as the source of the data bases. The *cost of computation* for reconstruction of the original data base can be reduced significantly with incremental approach, and the original data base will be made *reusable* without much effort for recomputation.

REFERENCES

[Aho 85] A. V. Aho, R. Sethi and J. D. Ullman, *Compilers : Principles, Techniques and Tools,* Addison-Wesley Publishing Company, Reading, MA, 1985.

[Ghezzi 80] Carlo Ghezzi and Dino Mandrioli, "Augmenting Parsers to Support Incrementality," *J. ACM*, vol. 27, no. 3, pp. 564–579, July 1980.

[Su 87] K.-Y. Su, J.-N. Wang, W.-H. Li and J.-S. Chang,"A New Parsing Strategy in Natural Language Processing Based on the Truncation Algorithm," *Proc. of Natl. Computer Symposium (NCS) 1987*, vol. 2, pp. 580-586, National Taiwan University, Taipei, R.O.C., Dec 17-18, 1987.

[Su 88] K.-Y. Su and J.-S. Chang, "Semantic and Syntactic Aspects of Score Function," *Proc. of the 12th Int. Conf. on Comput. Linguistics, COLING-88*, vol. 2, pp. 642–644, Budapest, Hungary, 22-27 Aug. 1988.

[Woods 70] W.A. Woods, "Transition Network Grammars for Natural Language Analysis," *CACM*, vol. 13., no. 10, pp. 591-606, ACM, Oct. 1970.

漢語的時間詞組和語言剖析

張莉萍

中央研究院計算中心，輔仁大學語言所

論文摘要

時間詞組在漢語中所扮演的角色及出現的位置，很容易讓電腦在剖析(parsing)語言時，誤以爲是動詞的必要論元(argument)，而誤導剖析的方向或使得剖析的過程複雜化。因此，本文整理出漢語時間成分的各種表達形式並提出判定時間成分的模式，使電腦能辨認時間詞組並解決含混情形。此外，本文試圖爲漢語的時間概念分類，希望從它們的形式特徵以及出現在句中的位置幫助電腦進一步的了解句意。

1.前言

剖析漢語時，動詞是整句的中心，適當的動詞分類可以決定一個句子的語法結構及相關的語意訊息[Chen et al. 88]。因此，規定一個動詞帶幾個論元(argument)，對於電腦剖析(parsing)自然語言而言，是非常重要的訊息。但是，有時句子的某些成分(constituents)並非動詞支配的語意論元，如：副詞、連接詞、地方詞、時間詞。尤其我們在分析天下雜誌上的文章時，發現幾乎每三個句子中，就出現一次表示時間的成分，出現的頻率相當高，而且時間成分往往是名詞性的詞(組)；不像副詞、連接詞這些成分，可以由詞類得知它們不是動詞的必要論元；也不像地方詞，往往由介詞引介出現；而時間詞組常常因爲它的詞性及出現的位置，容易被誤以爲是動詞的必要論元，如果電腦能辨認出它們是"額外"的論元，那麼不僅可以簡化剖句的過程，而且可以節省剖句的時間。例如"他寫了三個月，才寫了一篇短文"，這裡的"三個月"不是動詞的必要論元，而是表示"寫一篇短文"這個動作所持續的時間。如果電腦剖析時，就知道"三個月"是時間詞，而且放在動詞後面 [註1]，並不是動詞的必要論元，就不會把它和整個句子做匹配(matching)的工作。又例如 "昨天他沒來" 這個句子中，動詞前有兩個名詞組，如果電腦知道"昨天"是時間詞，就可以只管"他"是不是符合動詞的格框(case frame)及語意限制(semantic restriction)，而不用管"昨天"了。可見任何表示事件的語句，都可以加上時間概念來描述補充一個事件，而且在句中出現的頻率又高。本文就從時間成分著手，以天下雜誌中的語句爲範例，

針對中文的剖析步驟，整理出漢語中時間的表現方式及其成分，放入辭典(lexicon)中，讓電腦在斷詞時能立即判斷出句子中的時間成分，以簡化下一步的句子分析過程，進而幫助電腦理解句意。

2．時間成分的種類及判定方式：

首先，我們在整理漢語語法時，將時間成分依詞性分爲三大類：一是時間名詞(組)，二是表時間的介詞片語，三是時間副詞。以下就這三類的內容與語法特徵，分類詳加討論電腦如何判定它們是表示時間的成分。

2.1 時間名詞(組)

依構詞法再分爲三小類：

a.簡單式時間詞:

這一類由單純的名詞(沒有任何標記)形成的時間詞，在辭典中是列得完的一類，包括：

朝代名，如：夏、商、周、...

年號名，如：乾隆、雍正、...

干支名，如：甲子、丙寅、...

節氣(或節日)名，如：冬至、雨水、端午、中秋、...

副詞性時間詞，如：現在、過去、未來、...

一般時間詞，如：早上、中午、晚上、時代、朝代、日子、時候、時期、月、星期、...

b.定量式時間詞(組)

定詞加時間量詞構成時間詞(組)，例如："每天"、"上禮拜"、"前三天"[註2]。我們不可能在辭典中窮舉這一類的詞，不過，我們可以窮舉時間量詞以及能和時間量詞構成時間詞(組)的定詞。如下所示:

定詞：數詞定詞、這、那、哪、前、後、上、下、初、周、今、明、昨、去[註3]、第、半、多[註4]、每、某、本、此、該、當、全、整、幾、多少、許多、很多、好多、好些、 好幾。

時間量詞：世、紀、世紀、年、年度、載、歲、月、旬、星期 、禮拜、週、日、天、夜、時、更、點(鐘)、刻(鐘)、分(鐘)、秒(鐘)、小時、期、學期、季、朝、回、次、陣(子)、會(兒)、生、輩子等[註5]。

另外，我們建議把週(或禮拜或星期)一、二、三、四、五、六、日(或天)及初一... 初十，這些數量有限，構詞法又很特別的時間名詞當做漢語的時間專有名詞放入辭典中。

c. 和方位詞構成時間詞(組)

這一類可以是名詞組、動詞組或句子加上方位詞形成表示時間的成分，例如："春天裡"、"消息傳出後"、"他進門之前"。由於方位詞所標誌的可能是表示地方的概念，也可能是表示時間的概念，如："桌子上"的"上"一定標示地方，"十八世紀時"的"時"一定是標示時間；因此，我們需要將含有時間特徵的方位詞標記出來，以便和具有空間特徵的方位詞識別開來。時間方位詞是可以窮舉的一類，如右：前、以前、之前、後、以後、之後、內、以內、之內、中、之中、間、之間、中間、當中、來、以來、頭、裡、底、末、終、初、起、時、開始、左右。又因爲時間方位詞中，有部分也兼空間方位詞 [註6]；就我們歸納的結果發現，這些方位詞前的成分如果是時間詞組或是帶事件特徵(event feature)的詞組(以下簡稱Event)，才能說和方位詞構成的是一個表示時間的成分。所謂帶事 件特徵的詞組 [註7]是指名詞組、動詞組或句子蘊含事件的語意概念，如："車禍"。而和 Event共存的時間方位詞有：時、中、中間、當中、間、前、後、以前、以後、之前、之後、以來。例子有："他當財政部長時"、"休息中"、"談話中間 "、"演奏當中"、"慌亂間"、"股市尚未崩潰前"、"消息傳出後"、" 他就任以來"。

知道了以上三種構成時間詞(組)的方式，要讓電腦在句子中辨識出時間詞(組)就不是件難事了。我們的作法是在時間名詞、時間量詞的詞項記載(lexical entry)之下給予詞類以及時間特徵(time feature)。詞類是幫助電腦剖析句子時，知道什麼樣的詞類可以結合成一個成分 (constituent)，時間特徵是讓電腦能判別它們是表示時間的成分，如(1) "上"和"星期"構成一個時間成分，(2) "下"、"個"和"月"構成時間成分。

(1) 他 上 星期 去 美國。

N 定 量詞 V N

[+time]

(2) 他 下 個 月 考試。

N 定 量 N V

[+time]

這些詞組都可以由時間量詞或時間名詞判斷出是表示時間的成分。

如果是 *Event* 和時間方位詞構成一個時間成分，我們的作法是規定它們有特徵共存限制*(feature coocurrence restriction)[Gazdar et al. 85]*，隱含時間特徵，如下所示：

[+Event] & [+TLOC] ⊃ *[+Time]*。

例如*(3)*"進去時"是一個帶事件特徵詞組和方位詞構成的時間成分，"戴"才是這個句子的主要動詞。再如*(4)*，第一個子句是表示時間的附屬子句，第二個子句才是句子的主要子句。

(3) 他　進去　時　戴　了　一頂紅帽子。

N　*V*　方位詞　*V*　*Asp*　*NP*

[+Event]　*[+TLOC]*

(4) 消息傳出後，大家都很擔心。

2.2 表時間的介詞片語

關於表時間的介詞片語這一類，由於介詞所引介的對象不只一種，可能引介時間、地點、工具、情況等語意角色，爲了讓電腦能辨別介詞所引介的是一個表示時間的成分，我們整理分析了所有表達時間觀念的介詞片語，得到了"介詞+時間詞*(*組*)*+後續成分"這個句式 *(pattern)*，並窮舉各個介詞和後續成分的共存限制。茲分爲下列七類，每類下面各舉兩個例子：

a. { 在 / 於 } { 時間名詞*(*組*)* / *Event* } { 前、之前、以前、之後、以後、後、時、中、間、中間、內、裡、初、末、底、終、左右、*(*的*)*時候。 }

(5) 在這問題未解決之前...

(6) 部分結果終於在十月下旬出來了...

這一類如果省略後續成分，介詞後只能接時間名詞*(*組*)*。

b. { 到、直到、等到 } { 時間名詞*(*組*)* / *Event* } { 時、*(*的*)*時候、後、之後、以後、*(*爲*)*止 }

(7) 直到他認錯爲止...
(8) 等到我死了以後...

a 和 b 類介詞還有一個特殊現象，即介詞後接地方詞，再接一個"時"，仍是表示時間的成分而非地方成分，如(9)

(9) 他在學校時，是個好學生。

c. { 趁 } { 時間名詞(組) / Event } { 前、以前、之前、時、 / (的)時候 }

(10) 趁天黑以前...
(11) 趁我吃飯時...

d. { 自、從、自從 / 打、打從 } { 時間名詞(組) / Event } { 到 } { 時間名詞(組) / Event }

(12) 從你進門到現在...
(13) 自你起跑到抵達終點...

e. { 自、打、從 / 打從、自從 } { 時間名詞(組) / Event } { 來、以來、後、以後 / 之後、起、開始、至今 }

(14) 從訪問團抵達機場開始...
(15) 打從1982年底鄧小平的經濟改革政策宣布以來...

f. { 當 } { Event } { 後、之後、以後、時、(的)時候 }

(16) 每當價格管制在某一部門放寬和「自負盈虧」後...
(17) 正當美國股票市場在十月中旬遭受到歷史上最慘重的跌落時...

g. { 離、距 } { 時間名詞(組) / Event }

(18) 離暑假還很久
(19) 距他回國還有一段時間

我們所以把表時間的介詞片語句式整理詳列出來，主要是因爲漢語中時間概念的表達方式實在是精細而繁多，目前我們只要能做到讓電腦辨識出句子裡面的時間成分，不致於和動詞的必要論元搞混，以及能辨別那些是時間成分的合法表示方式，就可以解決句子中附屬成分 (Adjunct)一大部分的問題。也就是說電腦在剖析句子時，除了動詞的格框及語意限制可以幫助電腦剖析及決定語意論元

外，附屬成分或功能詞所引介的語意角色，則要靠進一步的語法整理來幫助電腦連結或理解語意。然而漢語中的功能詞(即介詞)，通常不只引介一個語意角色，因此，我們整理出來的表示時間的介詞片語句式，可以幫助電腦判斷在什麼樣的情況下，介詞所引介的是一個時間成分。如(20)

(20) 大家趁天黑趕快行動。

"天黑"是一個帶事件特徵的詞組，"趁天黑"則是一個表示時間的成分。又如(21)

(21) 他趁這個機會打他一頓。

"這個機會"只是一個普通名詞，這裡的介詞片語是表示動作所利用的條件或機會。因此，針對電腦剖析語言的需求，如果介詞所引介的是時間名詞(組)，這個名詞所帶的時間特徵也會向上傳給介詞；如果介詞所引介的是 Event，我們則規定它們有特徵共存限制，隱含時間特徵，如下所示：

$$[-N\ -V\ \ [+TEMP]]\ \&\ \ [+EVENT\ ] \supset [\ +TIME\ ]$$

這麼一來，介詞組就帶有時間特徵。 至於介詞和後續成分的共存限制，是爲了讓電腦能辨識時間成分的合法表示方式，如(22)，"自從"這個介詞本身就帶有從一個特定時間到以後的語意，所以它不能和"以前"之類的方位詞共同出現。

(22) a. 自從他出國以後，就沒有消息了。
b. *自從他出國以前，就沒有消息了。

2.3 時間副詞

時間副詞在我們的日常言談中常常出現，但是它們多不表示"時"，而是 表示"態" [陸儉明等 85]。例如：

表示過去時態：曾、曾經、已經、一度、早已、老早。
表示未來時態：早晚、終歸、即將、就要、快。
表示進行時態：正(在)。

這些訊息對於機器翻譯而言，也是很有幫助的，尤其中英翻譯所遇到的時態問題。目前爲止，我們尚未對時間副詞詳細分類，或許值得朝這方向去做。不過，時間副詞不具衍生性(productivity)，在辭典中是列得完的一類，而且出現的位置也很有規律，幾乎都是在動詞前面，要讓電腦辨識，不是一件難事。我們的作法是在詞項下給詞類及時間特徵即可。

3. 時間概念的分類與特徵

前面我們談電腦如何去判定時間詞組，那麼時間詞組出現的位置是不是固定呢？就我們分析文章後，發現大部分的時間詞組都出現在動詞之前。而一般人對於時間的概念大致可以分爲三種：

一是時頻 (frequency)。表示動詞所表達的事件發生的頻率。它的形式特徵有"每"加時間量詞，如"每天"、"每年"；時間量詞的重疊貌，如"天天"、"年年"；還有頻率副詞，如"常常"、"經常"。時頻只能出現在動詞之前。如(23)

(23) 他每年去美國渡假。

另外，可能有人覺得在"他三天喝了三次水"中的"三天"有頻率的意味，實際上，它是受了"三次"的影響，"三天"應該還是指"喝了三次水"整個事件所花的時間長度。

二是指事件發生的時間，可以是時間座標上特定的一點，也可以是特定的一段時間。它的形式特徵分爲： a. 有特指定詞或指示定詞[張等 88]修飾的時間詞(組)，如 "這星期"、"去年"、"上個月"； b. 介詞時間片語，如 "從昨天到今天"； c. 和方位詞構成的時間詞(組)，如 "兩年後" ； d. 無標記特指時間表示法，如 "三點五十分"、"1983年"。以上的這些時間表示法,都是定指的，而且只能出現在動詞之前〔註8〕，如(24)

(24) a. 我們今天上午十點鐘開會。
b. * 我們開會今天上午十點鐘。

三是指事件所花的時間或狀態持續的期間。它的標準形式是數詞或數量定詞加時間詞泛指時間的長度，如"三天"、"整年"；有時也在這些詞組後加上"之中"、"以內" 强調時間的範圍。這一類可以出現在動詞前後，如(25)-(27)

(25) 他三天沒吃飯了。
(26) 他兩個月就完成了這個工作。
(27) 他的手指腫了兩個月。

第二和第三類時間概念的區別，主要在於定指及非定指之分別。經由這些分類,我們得知在動詞後的時間詞組，是表示事件所花的時間或狀態持續的時間，定指的時間詞不能出現在這個位置。時間概念的分類，還是有助於電腦剖析時，理解句意以及辨別合法與不合法句子的能力。

4.結語

電腦剖析漢語，多以動詞的格框爲重要剖析訊息[Chen et al. 88]，以找到動詞的必要論元爲第一要務。本論文也是以這個原則爲出發點，我們實際分析了八篇去年十二月號天下雜誌上的文章，發現時間詞組的問題不容忽視。首先要讓電腦剖析時，能判別那些是表示時間的成分，不致於和動詞的必要論元搞混。由本文第二節得知電腦基本上可以從三方面來判定時間成分：

1. 時間名詞(組)或時間副詞，直接由詞項上的時間特徵看出。

2. 帶事件特徵的詞組＋時間方位詞，由特徵共存限制
([+EVENT]&[TLOC]⊃[+TIME])得知。

3. 介詞＋帶事件特徵的詞組，由特徵共存限制
([－N －V [＋TEMP]] & [＋EVENT]⊃[＋TIME])得知。

確定電腦能辨認漢語中的時間成分後，我們將三個時間概念粗淺地做一個界定，希望藉由不同的形式特徵以及出現的位置，進一步的幫助電腦理解語意或找出不合法的語句。當然，在這篇短文裡，想要完全掌握抽象而活潑的時間概念，似乎不簡單，例如"他每天跑三點半"這些習慣用法(idiom)，需要藉助經驗法則(heuristics)來處理，我們就不多作解釋。因此，本文完全是以電腦剖析語言的觀點，來研究整理時間成分，相信能解決大部分的問題。

☆☆本文之研究得中央研究院計算中心與工業技術研究院電子工業研究所合作之中文詞知識庫計畫(Chinese Knowledge Information Processing CKIP)之補助，並承陳克健、黃居仁兩位教授悉心指導以及詞庫小組各同仁閱後提出許多寶貴意見，本人由衷感謝！此外，謝謝楊淑吟、李惠君兩位小姐百忙中爲我打出這篇論文。

[附註]

1. 目前詞庫小組的分類系統中，只有氣象動詞這一類(如："下雨"、"出太陽")的主語(即必要論元)是時間詞。

2. 在詞庫小組的分類系統中，"前"是特指定詞、"三"是數詞定詞，這裡所指的定量式，可能是一個 、或二個、或三個定詞和量詞構成定量式，至於定詞和定詞間的共存限制，見張等(1988:12)。

3. 雖然"今、明、昨、去、初、周"後面可接的時間量詞非常有限，但爲了語法分類上的一致性，我們仍將這些單詞放入定詞中。

4. "半"和"多"出現的位置有兩種，一是一般定詞所在的位置、即[__量詞]，如：多年、半天；一是不同於其它定詞的位置、即[數詞定詞　量詞__(時間名詞)]，如：三年多、三個半月。

5. 其中"世紀、年度、月、星期、禮拜、小時、學期"兼具名詞的特性。

6. "前、之前、後、之後、內、以內、之內、中、之中、間、之間、中間、當中、頭、裡、底"也是空間方位詞，例如：" 桌子前 "、" 校園裡 "。

7. 我們規定動詞組或句子一定帶有事件特徵。

8. 如果說定指的時間詞可以出現在動詞後面，那也只有右邊的這種情形："他搭五點十分的(火車)"。"的"一定要保留，這裡牽涉到"的"的語法功能，本文暫不做進一步的探討。

[參　考　書　目]

1. 呂叔湘 (1984)　現代漢語八百詞　香港商務印書館。

2. 陸儉明、馬眞 (1985)　現代漢語虛詞散論　北京大學出版社。

3. 陳克健，陳正佳，林隆基 (1986) "中文語句分析的研究--斷詞與構詞"，技術報告 TR-86-004，南港中央研究院資訊所。

4. 張麗麗等 (1988)　國語的詞類分析(修訂版) 技術報告 0002，南港中央研究院計算中心。

5. 魏文貞、黃居仁 (1988) "中文的條件句"(待刊中)。

6. Chao, Yuen Ren (1968) A Grammar of Spoken Chinese. Berkeley and Los Angeles: University of California Press.

7. Chen, C.D., K.J. Chen & L.S. Lee (1986) " A Model for Lexical Analysis and Parsing of Chinese Sentences, " Proceedings of 1986 ICCC, Singapore, PP. 33-44.

8. Chen, K.J., L.L. Chang, C.R. Huang, C.C. Hsieh (1988) " A Classification of Chinese Verbs for Language Parsing, " CPCOL , Toronto.

9. Gazdar, G., E. Klein, G. Pullum, I. SAG (1985) " Generalized Phrase Structure Grammar, " Cambridge : Harvard University Press .

10. Li, Charles N. & Sandra A. Thompson (1982) Mandarin Chinese: A Functional Reference Grammar. Berkeley and Los Angeles: University of California Press.

11. Paris, M.C. (1988) " Durational Complements and Verb Copying in Chinese, " Tsing Hua Journal of Chinese Studies, New Series XVIII, No2, Hsinchu.

12. Sells, Peter (1985) "Lectures on Contemporary Syntactic Theories," Stanford : CSLI .

13. Tai, James H.Y. (1984) " Verbs and Times in Chinese : Vendler's Four Categories, " Lexical Semantics, pp.289-296, Chicago: Chicago Linguistic Society.

The Processing of English Compound and Complex Words in an English-Chinese Machine Translation System

Shu-Chuan Chen* **and Keh-Yih Su****

***BTC R&D Center**
2F, 28 R&D Road II
Science-based Industrial Park
Hsinchu, Taiwan, R.O.C.

****Department of Electrical Engineering**
National Tsing Hua university
Hsinchu, Taiwan, R.O.C.

ABSTRACT

In a machine translation system the information of the words of the source language should be available before any translation process can begin. The information of simple words can be obtained only by entering a word with all its relevant information into the lexicon. On the other hand, compound words and complex words, it seems, can be handled in a satisfactory way by lexical redundancy rules, and will thus help keep down the size of the lexicon. This paper argues that lexical redundancy rules are not as useful as they may seem to be for a machine translation system, and both their limitations and functions will be examined in depth. In addition, detailed discussions on the various problems that may arise during analyzing and translating of compound and complex words are presented.

1. Introduction

In a machine translation (MT) system the information of the words of the source language should be available before any translation process can begin. The information of simple words can be obtained by entering a word with all its relevant information into the lexicon. On the other hand, compound words and complex words at one extreme may be expected to be exhaustively listed in the lexicon [Zhan87] or at the other extreme be handled in a satisfactory way by lexical redundancy rules. However, it is obvious that since new compounds and complex words are created from day to day, they are impossible to be exhaustively listed. And as it would be made clear in the following discussions that the predictability, or regularity, of derivational words, inflectional words, or compounds is of limited use as far as translation is concerned, the lexical redundancy rules used to account for these words often fall short of their functions when applied in a MT system. Competent strategies are needed to successfully handle these two types of words to guarantee correct parsing and translation, and also help keep down the size of the lexicon.

In this paper, the various problems encountered in the morphological analysis module of the BTC English-Chinese MT system are discussed and possible solutions are proposed. The discussion will focus on the role of lexical redundancy rules in a MT system; and the issue as to whether English compound and complex words used in such a system should be derived from their stems solely through lexical redundancy rules. At last, we will look into the problems of processing multi-affix words and compounds of different formations.

2. English Compound Words, Complex Words, and Lexical Redundancy Rules

In English, new words may come into being through the process of derivation, inflection, or compounding. These processes, distinct from other less productive word formation devices, e.g. clipping, acronym, etc., create new words by adding new morphemes. Compounding creates words by adding one base to another and the forms created are called compounds. Derivation and inflection produce words by adding an affix to a base. Complex words, often used by linguists to mean exclusively for formations by the addition of derivational affixes to compounds, will be used in this paper to cover forms with either derivational or inflectional affixes for the reason that they are both created by affixation and thus require similar operations in a MT system. Words formed by these processes are large in number and bear a fixed phonological, syntactic[1], and semantic relation either to the stem of the complex word or to the grammatical head of the compound word.

For a MT system like ours whose input is written strings rather than spoken words, the syntactic and semantic predictability of compound words and complex words is of special interest. Due to the predictability, it appears that these words can be recognized and analyzed by lexical redundancy rules, and need not be listed as separate lexical items in order to reduce the memory space of lexicon. Lexical redundancy rules are intended to assign default form class, semantics, and other attributes to a group of words that share formal and functional resemblance. As an example, complex words ending in the suffix -ment, such as *arrangement*, *puzzlement*, etc., are all nouns and have a common meaning of "the result of " the action indicated by the verb base, and so on [Quir85]. And these words can be generated (in a

language generation system) or analyzed (in a language analysis system) by a redundancy rule of the following simplified form:

V + -ment —> N
Meaning : the result of V
Chinese translation : same as V

The arrow indicates that the word created by adding the suffix *–ment* to a verb is a noun with its Chinese translation identical to the base. An example involving compounds can readily be cited as well. However, lexical redundancy rules are not as useful as they appear in a MT system for the reasons to be discussed in the following section.

3. Limitations of Lexical Redundancy Rules in a MT System

Ideally, the use of lexical redundancy rules in a MT system will help restrict the lexicon to a reasonable size, and thus keep down the space allocated for storing lexical items. Nevertheless, the question as to whether a compound or complex word should be entered into lexicon, or whether they should be analyzed by rules, is not merely a matter of the size of lexicon, especially when the time spent on searching and analyzing a lexical item and the memory taken up by lexicon is trivial. (It is observed that in our system morphological analysis, including I/O, takes up less than 5% of the total processing time.)

The main concern of a MT system is to render a suitable translation from the source language to the target language. To this end, several conditions have to be satisfied, and they are the determining factors as to whether a compound or complex word should be built into lexicon or not.

1. Compositionality of translation. The translation of a multi-affix word is not necessary compositional, meaning the translation of such a word is not necessarily the composite of the respective translations of the affixes plus that of the stem [Zhan87]. For one thing, the suitable translation is subject to Chinese word-formation rules[2]; for the other, if there already exists in Chinese an established term for the same idea expressed by a given English word, the established word are most likely to be used as the corresponding translation. In the absence of compositionality, correct translation can not be obtained by general rules. In this case, the multi-affix word has to be entered into the lexicon. As an example, the derivational word *reconfigurability* is formed by attaching *re-* to the root *configure*; then *-able* to *reconfigure*; and finally *-ity* to *reconfigurable*. Provided that *re-* is given the default translation " 重新 ", *–able* " 可以 ", *-ity* " 性 " and *configure* " 配置 ", the Chinese translation of *reconfigurability* is not likely to be the composite of the translations of *re-*, *-able*,, and *-ity* plus that of *configure*, that is " 可以重新配置性 ". A potential candidate is " 重構性 "[3]. The same criterion goes for compounds. An example of it is *flesh-and-blood*. When translated compositionally, it would be " 肉和血 ". However, the corresponding institutionalized translation of the compound is " 血肉之軀 "
2. Adequate information for rendering correct translation. A variety of morphological, syntactic, and semantic information is needed for an English word to be correctly translated into Chinese. If any single piece of information of an English lexical item failed to be obtained through default assignment by lexical redundancy rules, this word has to be

entered into the lexicon. For example, it is necessary to consult the subcategorization restrictions of the Chinese words (Mandarin, to be more precise, since the BTC MT system is actually English to Mandarin) " 老 " and " 舊 " to determine which word the English word *old* should be translated. The rule is basically that if *old* modifies an animate noun, it should be translated as " 老 " and never " 舊 ". Suppose that the word *dancer* is generated by a redundancy rule that derives nouns by adding *-er* to verbs and stipulates that the derived noun must indiscriminately be of the attribute "inanimate" (which can be animate as well), *dancer* will be erroneously labeled as an inanimate object. This will result in the translation of *old dancer* into " 舊的舞者 " , and not the correct " 老的舞者 ". Therefore, words like *dancer* must be listed in the lexicon.

3. Ability to identify elements in a compound. Compounds may take the form of two separate words, such as *hard copy*, or a hyphenated word, such as *hard-copy*. The elements of a compound can also be combined together as a single word, such as *hardcopy*. The last form proves to be very difficult in identifying its composite morphemes. In this case, despite the regular syntactic and semantic ties between a compound and its elements, compounds of this makeup have to be built into the lexicon.
4. Productivity of affixes. The productivity of an affix also plays a role in the admissability of a word into the lexicon. Words derived by an affixation of limited productivity should be entered as separate lexical items, because they are very few in number. Otherwise, the addition of a non-productive morphological rule may increase the complexity of the system and the processing time as well. For example, the prefix *step*, denoting kinship, is no longer productive [Baue83], and we should enter all the words with this prefix[4] into the lexicon. However, a risk in deciding to leave out a marginally productive affixation rule is that as it is not actually extinct, occasional coinings are still possible. There is a tradeoff to be made in this regard.

4. Functions of Morphological Redundancy Rules in a MT System

The above criteria will eliminate most compound words and some complex words from the possibility of being analyzed by lexical redundancy rules. Although the use of redundancy rules is restricted by the concern for rendering a correct translation, they are still important in three areas. First, since inflectional morphemes preserve the category of their stems, and the corresponding Chinese translations of inflected forms are highly regular, the majority of them should be handled by rules.

Second, redundancy rules can be used to predict the possible category for a word not in the lexicon by examining just the affixes attached to it. For example, if the word *absentmindedness* is not in the lexicon, while *absentminded* is, a rule that identifies a word which is made up of an adjective plus the suffix *-ness* is a noun, the word *absentmindedness* will be given the correct category. This makes it possible to assign a correct category to a word, and which is one of the prerequisites in producing a correct parse tree for a given construction. Once the right structure is obtained, the whole construction will be correctly translated as a result.

Third, redundancy rules are of avail in giving a suitable translation to words not in the lexicon by considering the semantic relation they bear to the stem of a complex word or to the head of a compound word. In line with the rule that gives category to *absentmindedness*, the Chinese translation of the same word can be obtained by giving *-ness* a default translation.

Thus lexical redundancy rules are helpful in providing as much information as possible in both parsing and transfer phases. This is the general idea and technique behind the "fail-soft" in a MT system [Benn85]. The conclusion to be drawn regarding the use of lexical redundancy rules is that: Compound words and complex words should be built into lexicon if good translation is not available through default assignment by lexical redundancy rules. On the other hand, since new words are constantly created, lexical redundancy rules are indispensable.

In Sections 5 and 6, we will examine the internal structure of compound and complex words, and the effect it has on the processing of these words in a MT system.

5. Processing of Compound Words

Among the three types of compounds noted in section 3, only compounds with elements separated by a space or hyphen are of interest as far as processing is concerned. The type of compounds spelt as a single word will all be listed as lexical items in our system, because they are difficult to process.

As with compounds composed of elements separated by a space, quite a few are established compounds, and this type of compounds is rather productive in coining new ones, found particularly in the terminology used in a specific field of study. As for the kind of rules needed to analyze these compounds, it can be either a morphological rule or a syntactic rule. The former will recognize the compounds at the phase of morphological analysis, which is prior to syntactic analysis. But this can alternatively be done during syntactic analysis; that is, compounds of separate elements are treated like a phrase in order to eliminate the need of an extra operation during morphological analysis. Thus, the processing of a noun compound like *prototype development system* can be left until syntactic analysis phase to be parsed as a noun phrase. This can be done because phrases and compounds share quite a lot of common ground in their internal structure; in other words, word syntax is on a par with phrase structure [Tang88].

The most frequently encountered compounds are the hyphenated compounds, and it is this type of compounds for which lexical redundancy rules are of the greatest use. Hyphenated compounds used as adjectives are extremely productive and most of them are the instances of occasional coinage. Established ones are fewer in comparison to occasional creations. For instance, compounds made of cardinal plus noun, such as *40-word* in *40–word lexicon*, are extremely productive.

Formally, several individualities of hyphenated compounds are noteworthy. First of all, they may take an entire phrase as its elements, e.g. *higher-than-average (an adjective phrase)* in *higher-than-average wages*, and *do-it-yourself (a verb phrase)* in *do-it-yourself approach.* Second, suffixes may be attached to the last element of a compound which does not normally take such suffixes when used as an independent word. For example, the noun in a compound expressing physical attribute might take the past participle ending, e.g. *leg* in *three-legged table.* And in compounds that express fraction, ordinals might take the plural ending, e.g. *third* in *two-thirds*[5]. In addition, a number of grammatical relationships are possible between the components in a compound, and different types of meaning and translation will thus result. For example, in a noun-verb compound, the grammatical relation between noun and verb may be instrument-action, such as *petrol-lighter*, in which *petrol* is the instrument the lighter uses. Whereas, in *fire-lighter*, *fire* is the object of the action *light.*

The formal and semantic attributes of hyphenated compounds observed above have the following effects on processing and translating these words. First, for words like *three-legged* and *two-thirds*, special rules have to be constructed for handling the irregular inflection.

Second, detailed rules have to be worked out to pinpoint the grammatical relation between the elements of hyphenated compounds in order for them to be correctly translated into Chinese. For instance, the corresponding translation of the instrument-verb compound *voice-controlled* may employ " 由 " to express the instrumental case, such as " 由聲音控制 ". While the corresponding translation of the object-verb compound *letter-writing* is simply placing the object after the verb " 寫信 ".

In view of the fact that the grammatical relationship of the elements within a compound is difficult to define, and the translation is far from certain even if the precise relation can be identified, therefore, the vast majority of these words have to be built into the lexicon. Nevertheless, there are two cases in which correct translation is possible without resort to lexicon. For a group of compounds that have the same stem and the stem also has a fixed translation in Chinese, translation rules can be constructed specifically for this stem. For example, there are a lot of compounds involving the stem *oriented* in their formation, such as *screen-oriented*, *row-oriented*, *column-oriented*, to name just a few. A rule to the effect that *noun-oriented* will be translated to "noun- 導向 " will be sufficient. On the other hand, compounds like *three-legged* which contains the same items and word order as in a corresponding noun phrase *three legs* can be handled by the very set of transfer rules constructed for translating English phrases into Chinese. So, the phrase *three legs* when translated into Chinese needs a classifier " 隻 " before " 腳 " to give " 三隻腳 ". The same holds for a compound containing identical elements and functioning as a modifier of another noun, i.e. *three-legged* as in *three-legged table*, whose translation will thus be " 三隻腳的桌子 "

Third, English phrase compounds are phrase in nature and, when used as a modifier of nouns, correspond closely to the structure of Chinese noun phrases: when modifying a noun, phrasal modifier and clausal modifier, are pre-modifier rather than post-modifier of nouns in Chinese. For example, in English *three-year-old* can be a noun or a modifier of noun, as in *a three-year-old girl*, which is equivalent to *a girl who is three years old.* Both the phrase compound *three-year-old* or the noun phrase *three years old* will be translated identically as " 三歲 ". Hence, the translation of phrase compound can be taken care of by transfer rules as well.

6. Processing of Complex Words

An English complex word exhibits several characteristics that are pertain to the processing of complex words in a MT system. First, English inflectional affixes are all suffixes, while derivational affixes can be either prefixes or suffixes. Second, in terms of the number of derivational and inflectional affixes, a complex word may consist of more than one derivational affix, with an additional inflectional suffix outside these derivational affixes. For example, *configurabilities* is formed by adding derivational suffixes *–able* to *configure*, *-ity* to *configurable*, and the inflectional suffix *–s* to *configurability*. Third, in terms of the number of prefixes or suffixes, a complex word may have more than one prefix or suffix. For example, *unrerunability* has two prefixes *un-* and *re-* and two suffixes *–able* and *-ity*. Fourth, suffixation, but not prefixation, may cause changes in the orthography of the stem forms[6]

For example, the suffix *-able* when attached to a verb ending in *e*, will sometimes delete the final *e*, e.g. *consume* becomes *consumable*.

Based on the characteristics of complex words observed above, the processing of words with only prefixes, words with only suffixes, and word with both prefixes and suffixes each requires different operations. For words with prefixes alone, de-prefixation is followed by dictionary look-up to check if the stem can be found in the lexicon. If the word is found then no prefix should be further removed. If a stem can not be located in the lexicon, two things are possible. First, there is no such word in the lexicon and thus it should be assigned the category specified by the rule in order for the sentence in which it occurs to be successfully parsed. Second, if there is another prefix after the current one, further de-prefixation will unravel the stem. The operations de-prefixation requires are depicted in Figure 1:

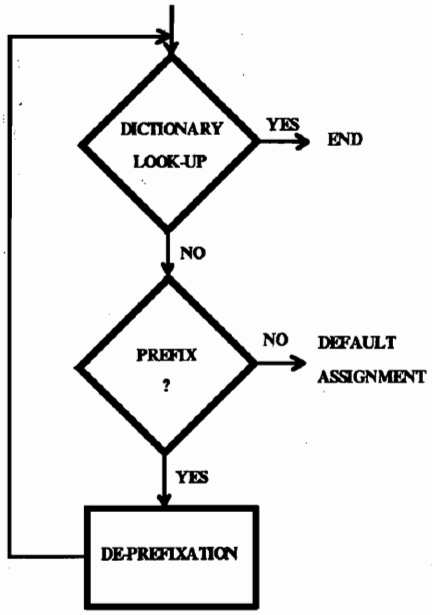


Fig. 1 THE FLOW OF DE-PREFIXATION

For words with suffixes alone, de-suffixation is likewise followed by dictionary look-up to check if the stem can be found in the lexicon . However, if a stem can not be found in the lexicon, three things are possible. First, it may due to the fact that there is no such word in the lexicon and a suitable category should be assigned. Second, it may be that there is another suffix before the current suffix. In this case, further de-suffixation is needed. Third, it is also possible that suffixation process has altered the orthography of the stem; and only after the original form has been restored, can dictionary loop-up be performed to see if another suffix should be removed. For example, after *-able* is removed from *consumable*, the form *consum* is not a word, and an *e* has to be restored. De-suffixation requires the following operations in Figure 2:

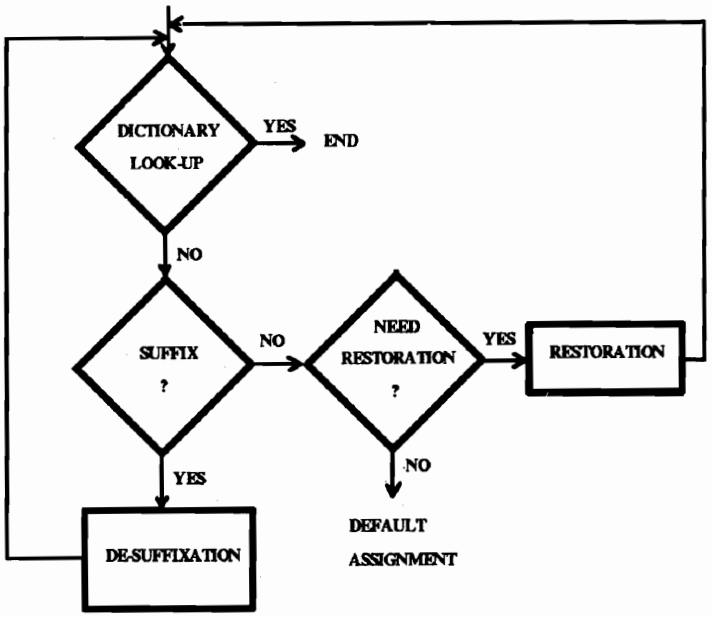


Fig. 2 THE FLOW OF DE-SUFFIXATION

To make the situation more complicated, words containing prefixes in addition to suffixes call for both de-prefixation and de-affixation. This involves a back-and-forth check on both ends of a word. The check can be initiated at either end. Once the category matches the specification in a prefixation rule or that in a suffixation rule, de-affixation has to be done. For example, to de-affixize *reconfigurable*, either de-prefixation or de-suffixation can be tried first. If we start with the prefix, de-prefixation will fail since *re-* is a prefix that must be attached to a verb, but *configurable* is an adjective. At this point, we have to restart with de-suffixation, *–able* will be removed to yield *reconfigure*. After *-able* is removed the remaining form is *reconfigure*, de-prefixation can now be executed to remove *re-* and leaves *configure*

De-suffixation, and de-affixation in general, includes a check in category. If the stem does not match the category specified by the rule concerned, de-suffixation should not be carried out. The importance of matching category in the process of de-suffixation lies in the fact that it helps determine if restoration is in order. For example, after taking off the ending *-ing*, *using* will become *us*, which is not a verb and cannot be the right stem. Therefore it is obvious that a final *e* must be missing.

The processing of complex word, especially multi-affix complex words, may pose a number of problems:

1. The major problem is that for words that are not in the lexicon, there is no way of telling if they contain affixes or not. For a newly-coined word *prerechit* , we are not sure whether it contains a prefix, *pre-*; or two prefixes, *pre-* and *re-*; or no prefix at all. In this case, the principles of assigning default category and default Chinese translation may result in wrong guesses.

2. As pointed out above, after a suffix has been removed, if the remaining part cannot be found in the lexicon, it is likely that there is another suffix before the current suffix. In this case, if there is no change in orthography, further de-suffixation will unravel the stem. On the other hand, if there is a change in spelling, it is hard to detect if the remaining part is a word not in the lexicon or a word with more suffixes. There are, however, two possible ways to solve this problem. For instance, after *-ity* is detached, the remaining part of the

word *executability* is *executabil*, because the suffix *-able* has been transformed. In this case, we can either stipulate that if there is a string *abil* occurs before *-ity*, then *abil* should be restored to *-able*. The other way is simply to treat *ability* as a single suffix. Thus no further analysis of the internal structure of *ability* is necessary. The latter is a better solution for the sake of simplicity in processing.

3. The restoration of base words can be time-consuming. For example, the rule that derives nouns by adding the suffix *-sion* to a verb can cause a base form to lose its final *e*, such as *confusion*; or *t*, such as *conversion*; or *de*, such as *explosion*, etc. Every possibility has to be tried to restore the verb. To remedy this problem, if a given operation applicable to only a handful of words, these words might as well be listed in the lexicon. If we choose to do so, however, new words can not be accounted for if they happen to need restoration of this sort. Here, we are faced with another tradeoff.

7. Conclusions

In this paper, we provide a comprehensive look at the functions and limitations of lexical redundancy rules used in analyzing compound and complex words in a MT system. The conclusion is that since redundancy rules most of the time cannot guarantee correct translation of compound and complex words, it is suggested that redundancy rules be reserved for analyzing words that are occasionally coined in order for the construction to be parsed successfully. In the paper, various problems concerning the processing of compound words and complex words are examined, and possible solutions are proposed. Nevertheless, the problems presented in this paper are by no means exhaustive, and there are other difficulties in processing compound and complex words that are worth noting, such as the treatment of words like *passers-by*, which has an inflected form as the first element, and so on. In addition, idioms or collocations can also be regarded as a special case of compounds and are needed to be studied further. These issues, however profound they may be, are out of the scope of the current paper.

8. Acknowledgement

We are grateful to Professor Ting-Chi Tang at Foreign Languages Department of National Tsing Hua University for his helpful discussions and critical comments on this paper. Special thanks are due to the colleagues in BTC R&D Center for their support and encouragement.

NOTES

1. What we mean by syntactic relationship is mainly about the relationship in the word class between the composite elements and the whole compound or complex word.

2. Discussion of Chinese morphological rules is beyond the scope of this paper. For detailed discussions of Chinese morphological rules, please refer to [Tang88].

3. The Chinese translation of *configure* given in the English-Chinese Dictionary of Computing Technique is " 配置 ", and one of the translation of *reconfiguration* is " 重新配置 ". However in the compound *reconfiguration system*, *reconfiguration* is translated as " 重構

". This is also true for five other compounds containing *reconfiguration*. Based on this, *reconfigurability* is given the translation of " 重構性 "

4. There are only eight of them: *stepbrother*, *stepchild*, *stepdaughter*, *stepfather*, *stepmother*, *stepparent*, *stepsister*, and *stepson*.

5. Strictly speaking, the suffixes are added to the compound as a whole when functioning as an adjective , not to an individual component.

6. The prefix *in* also causes changes in spelling to the initial consonant of the base through an assimilation in pronunciation, e.g. *in* becomes *il* before the lateral *l* as in *illegal*; *in* becomes *im* before a labial as in *impossible*, etc. Since the prefix is no longer in productive use due to the competing prefix *un*, it is safe to state that prefixation does not cause any changes in the spelling of the base.

REFERENCES

[Baue83] Bauer, L., *English Word-Formation*, Cambridge University Press, Cambridge, Great Britain, 1983.

[Benn85] Bennett, W.S., "The LRC Machine Translation System," *Computational Linguistics*, Vol. 11, NOs. 2-3, pp. 111-119, April-September 1985.

[Biss85] Bissantz, A.S. and K.A. Johnson ed., "The Minimal Units of Meaning: Morphemes", *Languages Files*, The Ohio State University Department of Linguistics, 3rd ed., Advocate Publishing Group, Ohio, U.S.A., 1985.

[Hutc86] Hutchins, W.J., *Machine Translation: Past, Present, Future*, Market Cross House, West Sussex, England, 1986.

[Quir85] Quirk, R., S. Greenbaum, G. Leech, and J. Svartvik, *A Comprehensive Grammar of the English Language*, Longman Group Limited, Essex, England, 1985.

[Tang88] Tang, T-C., *Studies on Chinese Morphology and Syntax*, Student Book Co., Ltd., Taipei, Taiwan, 1988.

[Vasc85] Vasconcellos, M. and M. Leon, "SPANAM and ENGSPAN: Machine Translation at the Pan American Health Organization" *Computational Linguistics*, Vol. 11, Numbers 2-3, pp. 122-136, April-September 1985.

[Zhan87] Zhang, Liangping, and Shengxin Chen, "Ambiguity Processing in English-Chinese Machine Translation", Conference on Translation Today, Hong Kong, 1987.

名山出版社 (左宜有, 左宜德主编) 電腦資訊科學辭典 (English-Chinese Dictionary of Computing Technique (Data & Information)), 臺北, 1983.

DISAMBIGUATION OF PHONETIC CHINESE INPUT BY RELAXATION–BASED WORD IDENTIFICATION

Charng-Kang Fan and Wen-Hsiang Tsai
National Chiao Tung University
Hsinchu, Taiwan 30050
Republic of China

I. INTRODUCTION

Among the various Chinese input methods for computers, the national phonetic input method is the most favored one by casual users in Taiwan. Possible reasons include the following: (1) users need not decompose each character into parts which in most cases are puzzling non–conventional Chinese radicals; (2) everyone learns the national phonetic symbols in primary school; and (3) the number of the national phonetic symbols is only 37. Also, there exists a well–known ordering among the symbols. Hence it is easy for most people to memorize them well, which facilitates the finding of proper symbol keys, in contrast with the difficulty of searching the unnatural radical keys required by radical–based input methods.

Since Chinese characters are monosyllabic, a character can be represented by a syllable which usually consists of one or two vowels and an optional consonant plus a tone marker. The phonetic input method is to key in the syllables and convert them into corresponding characters.

Though the national phonetic input method is convenient for casual users, an inherited drawback does exist. Since only about 1300 distinct syllables are used for more than ten times of Chinese characters, the number of homonyms per syllable is quite large. Hence

ambiguities exist in determining the correct character for a given syllable.

II. KNOWN APPROACHES FOR PHONETIC INPUT

According to whether the processing is performed with or without human intervention; whether the processing unit is a character, a word, or a sentence; and whether the contextual relationship of adjacent syllables are used or not, existing methods are briefly surveyed as follows.

(1) Most available Chinese systems [1] require a user to input phonetic symbols, syllable by syllable, and leave the homonym resolution to the user. The user is required to select manually the desired character for an input syllable among a list of homonyms displayed on the computer screen.

(2) Wan, Saiton, and Mori [2] described a method by which users manually inserted a word break in a continuously entered Pinyin string and issued the conversion command to have the computer convert the sequence of syllables between two breaks into characters.

(3) Ho et al. [3] described a method for segmenting a sentence into segments. It uses three special classes (the 'pure word head,' the 'pure word tail,' and the 'pure monosyllabic word') of words as segment markers. The syllables between the markers can then be converted into characters automatically by matching against the stored word dictionary. But emphasis was not put on the use of the method for ambiguity resolution.

(4) Chen et al. [4] proposed an automatic continuous conversion algorithm to convert a string of syllables into characters word by word. The system has a dictionary for word look up, and a file to handle exceptions. Several ad hoc rules (such as "previous word first," "preference for highest usage frequency," etc.) are also employed to resolve the ambiguous cases. It did not use more general contextual information existing in a sentence.

(5) Lin and Tsai [5] proposed an automatic ambiguity resolution method by a relaxation process. It regards the phonetic input method as a task of assigning each individual syllable

to a proper character. Relaxation iterations are applied to reduce the ambiguities in the assignments, using the co–occurrence statistics of syllable pairs to compute the initial probability values and the compatibility coefficients needed in the relaxation process. The contextual information of a sentence is utilized, but only pairwise neighboring character relationships are explored; word relationships are not utilized.

III. PROPOSED APPROACH TO DISAMBIGUATION OF PHONETIC INPUT BY WORD IDENTIFICATION

A. Ideas

A new approach to automatic disambiguation of phonetic input by a relaxation–based word identification process is proposed. This approach is applied to a string of phonetic symbols or syllables which constitute a sentence. The approach is based on the following consideration: (1) the smallest meaningful unit of Chinese is word; (2) there are very few or even no homonym for each multi–syllabic word in contrast to a large number of homonyms for each monosyllabic word; and (3) there exists useful contextual relationship among the words in a sentence, which will be described later. Compared with Lin and Tsai's method [5] which assigns syllables to characters and utilizes the co–occurrence relationships of pairs of characters, the proposed approach assigns syllables to words and utilizes the adjacency relationships of neighboring words. Thus it is a word–oriented approach which employs the more meaningful contextual constraint information among the words in an input sentence.

By regarding the phonetic input text as a string of syllables, the proposed approach basically is a process of word identification which assigns syllables to words. For example, in the sentence

"you gu wun ti rcu jien yi bi jiau hau."[1]

(It is better that the suggestions be proposed by the consultant.),

although the numbers of homonyms are 11, 8, 7, 7, 3, 26, 50, 7, 8, and 2 respectively for the syllables according to the Eten Chinese system, the sentence is composed of (or should be segmented into) the words as "you" (by), "gu wun" (the consultant), "ti rcu" (propose), "jien yi" (the suggestions), "bi jiau" (comparatively), and "hau" (good, better), regardless of the number of the enormous combinations of homonyms (approximately 10^9 for this sentence). The essence of the proposed approach is to apply the word identification technique to the syllables of a sentence so that the goal of converting the syllables into characters can be accomplished simultaneously when the segmentation of the sentence into words is completed.

Relaxation is a problem solving paradigm which is used in a lot of class assignment or labelling problems to handle the situations of ambiguities. It iteratively employs the contextual information to modify the previous judgement, which lessens the ambiguities and finally converges to the most likely choice. The application problems to which relaxation has been applied includes noise removal, edge detection, scene labelling, image analysis, handwritten character recognition, etc. [6–15].

B. The Syllable–to–Word Assignment Problem

Let S represent an input sentence and W_j be an arbitrary Chinese word which are composed of n and m syllables, respectively, as follows:

---

[1]The Suen's phonetic symbols [17] are used to facilitate the pronunciation and reading of the Chinese characters.

$$S = s_1 s_2 \ldots s_n,$$

$$W_j = s_{w_{j1}} s_{w_{j2}} \ldots s_{w_{jm}},$$

where s_i or $s_{w_{jk}}$ denotes a syllable. Let

$$Q = \{s_1, s_2, \ldots, s_n\},$$

$$W = \{\text{all Chinese words}\},$$

then a syllable–to–word assignment A is defined as a mapping from Q to W:

$$A : Q \rightarrow W$$

such that the expression $W_j = A(s_i)$ means to assign s_i, $1 \leq i \leq n$, to the word $W_j = s_{w_{j1}} s_{w_{j2}} \ldots s_{w_{jm}}$, indicating s_i as one of the composing syllables of W_j (i.e., there exists an integer k, $1 \leq k \leq m$, such that $s_{w_{jk}} = s_i$). We will also use $s_i \rightarrow W_j$ or A(i,j) to denote the meaning of $W_j = A(s_i)$. Word identification is thus a consistent syllable–to–word assignment problem such that each syllable in a sentence is correctly assigned to the composing words of the sentence, and the assignment of each syllable is compatible with one another.

C. Initial Probability Values for Relaxation

There usually exist, in a sentence S, multiple words to which a syllable s_i may be assigned. Let AA_i denote the set of all assignments which assigns s_i to such words, i.e.,

$$AA_i = \{ A(i,j) \mid W_j \text{ is in S and } W_j = A(s_i) \}.$$

Let P_{ij} denote the probability estimate that s_i is assigned to W_j, and P_{ij}^0 the initial probability value of P_{ij}. It is proposed to define P_{ij}^0 as

$$P_{ij}^0 = \text{count}(W_j) \,/ \sum_{W_k \epsilon AW_i} \text{count}(W_k)$$

where $\text{count}(W_k)$ is the usage frequency count of word W_k which can be collected in advance, and AW_i is the set of all possible words in a sentence S containing syllable s_i, i.e.,

$$AW_i = \{ W_j \mid A(i,j) \epsilon AA_i \}.$$

D. The Relationships Among Neighboring Words

Following Fan and Tsai [15], the relationship between two neighboring words W_a and W_b in a sentence can be categorized into five classes.

(1) Interleaving, that is,

a. $W_a \neq W_b$ and

b. there exists a W_c such that W_c=suffix(W_a) and W_c=prefix(W_b), or W_c=suffix(W_b) and W_c=prefix(W_a), where the terms prefix and suffix represent the leading and the trailing strings of syllables in a word respectively.

(2) Containment, that is,

a. $W_a \neq W_b$ and

b. W_a=suffix(W_b) or W_a=prefix(W_b) or W_b=suffix(W_a) or W_b=prefix(W_a).

(3) Identity, i.e., $W_a = W_b$.

(4) Adjacency, i.e., suffix(W_a) and prefix(W_b) are adjacent in the sentence (assuming W_a is in front of W_b). This can further be classified into two cases as follows.

a. Loose adjacency such that suffix(W_a)·prefix(W_b) is not a word.

b. Intimate adjacency such that there exists another word W_c which is formed with suffix(W_a) and prefix(W_b).

(5) Irrelevancy, that is, $W_a \neq W_b$ and they are positionally apart from each other in the sentence.

Classes (1) through (4) will also be called relevancy relationships, in contrast to the irrelevancy relationship.

The relationship between two assignments can be defined in terms of the relationship between the involved words. Assignments A(i,j) and A(h,k) are said to be mutually

(1) neutral, if W_j and W_k are irrelevant or if W_j and W_k are loosely adjacent;

(2) supportive, if W_j and W_k are identical;

(3) opposing, if W_j and W_k are of the relationship of containment or interleaving;

(4) quasi–opposing, otherwise (i.e., if W_j and W_k are intimately adjacent).

E. Compatibility Coefficients of Syllable Assignments for Relaxation

Different relationships between assignments result in different effects of the relaxation iterations. Given two assignments A(i,j) and A(h,k), if they are mutually neutral, then neither of them will affect the other. If they are mutually supportive, then both of syllables s_i and s_h are assigned to the same word W_j, thus both supporting the identification of word W_j. If the assignments are mutually opposing, then either the two words W_j and W_k are interleaving, or one of them is contained in the other, either case reflecting a conflict situation. Finally, if the assignments are mutually quasi–opposing, they may or may not oppose each other, depending on the effect of the intermediate words constructible from the syllables in W_j and W_k.

Thus the compatibility coefficients C(ij;hk), indicating the degree of supporting or opposing by A(h,k) to A(i,j), can be derived based on the relationship of A(h,k) with respect

to A(i,j). The supportive ones are given positive C(ij;hk) values; the opposing ones, negative values; and the neutral ones, the values of zero. More specifically, the compatibility coefficients are defined in this approach as follows:

C(ij;hk) =

(1) 1, if A(i,j) and A(h,k) are mutually supportive;

(2) 0, if A(i,j) and A(h,k) are mutually neutral;

(3) –0.5, if A(i,j) and A(h,k) are mutually quasi–opposing;

(4) –1, if A(i,j) and A(h,k) are mutually opposing.

F. The Relaxation Iteration

P_{ij}^{0} is the initial estimate of the probability value for Assignment A(i,j). With the supporting or opposing information from its neighboring assignments, the probability estimate P_{ij} is modified in each iteration in the relaxation process to reflect the contextual constraints among the words in the sentence. Let the probability estimate for A(i,j) after the rth iteration be denoted as P_{ij}^{r}, then it is proposed to compute the updated probability estimate P_{ij}^{r+1} after the (r+1)th iteration as follows:

$$P_{ij}^{r+1} = \frac{P_{ij}^{r}\,(1+q_{ij}^{r})}{\sum_{AA_i} P_{ij}^{r}\,(1+q_{ij}^{r})} \qquad (1)$$

with

$$q_{ij}^{r} = \frac{\sum_{s_h \in ENS(i,j)} \left(\sum_{A(h,k) \in ENA(i,j)} C(ij;hk) * P_{hk}^{r} \right)}{ENN(i,j)}, \qquad (2)$$

where ENS(i,j) is the set of effective syllables which are relevant to A(i,j), ENA(i,j) is the set of effective neighboring assignments for A(i,j), and ENN(i,j) the cardinality of ENS(i,j).

More specifically,

$$ENS(i,j) = \{ s_h \mid h \neq i, \text{ and there exists an } A(h,k) \epsilon AA_h \text{ such that } A(h,k) \text{ and } A(i,j) \text{ are not mutually neutral } \};$$

$$ENA(i,j) = \{ A(h,k) \mid s_h \epsilon ENS(i,j) \};$$

$$ENN(i,j) = \| ENS(i,j) \|.$$

The inner summation (embraced by the parenthesis) in the numerator of Eq. (2) is the effect of a certain effective neighboring syllable s_h on the probability distribution P_{ij} during the (r+1)th iteration. The outer summation (the whole numerator part) in Eq. (2) is the total effect of the effective neighboring syllables of A(i,j) on P_{ij}. This total effect is normalized by the number of effective neighboring syllables ENN(i,j) of A(i,j) to keep the value of q_{ij}^r within the range [–1,1]. Note that $P_{ij}^{r+1} \geq 0$ and $\sum_{AA_i} P_{ij}^{r+1} = 1$.

The termination condition for the relaxation process must be defined. The most frequently used one is to set an upper limit on the process iteration times or to define a threshold value for the P_{ij} values. Fan and Tsai [15] proposed another condition which is adopted in this study. The condition requires that the P_{ij} value of any desired assignment of each syllable become greater than a pre–defined threshold value and that this P_{ij} value be increased for the last two iterations.

When the termination condition is satisfied, for each syllable within the sentence, the word of the assignment with the highest P_{ij} value is denoted as the word the syllable is assigned to, and then the corresponding characters are determined, too.

IV. EXPERIMENTAL RESULTS

A prototype system is built to test the applicability of the proposed approach. The computer is an IBM PC/AT compatible machine, with 640K RAM and 20M hard disk. A computer dictionary which includes about 4100 word entries is used for the experiment. Each word entry contains its length, phonetic codes, Chinese internal codes (Big 5 codes), and its usage frequency count. The word usage frequency counts recorded in [16] are directly used in the dictionary. The phonetic codes of the characters in a word are used as the index to search for the word in the word dictionary. The dictionary occupies 320K bytes of disk space.

Test data are selected from the articles in the " gwo yu ri bau." They are coded in phonetic codes. Each phonetic code represents a syllable and occupies two bytes. The test data include 1318 sentences and 16347 syllables in total.

The processing procedure for each sentence is as follows. First, the dictionary is checked to find all possible words (each word being a string of syllables) corresponding to the syllables of the sentence. Then the syllable–to–word assignment is made. With the usage frequency counts of the words, the initial probability values of the assignments can be computed, and the adjacency relationships of the words can be analyzed to compute the compatibility coefficients between the assignments. Finally, relaxation iterations are performed until the termination condition is met. The result is printed as a segmented sentence with syllables converted into characters and words identified with spaces as markers between them.

The experimental result is summarized as follows. With the termination condition described as in Section III.F, and the threshold value of 0.8, the correct conversion (converting the syllables to Chinese characters) rate is 96.91%; the average processing speed is 0.97 second per syllable; and the average number of iterations per sentence is 17.2. Some examples of the results are shown in Figure 1. The underlined characters are errors.

V. CONCLUSIONS AND SUGGESTIONS

A new approach to Chinese phonetic input is proposed. The input method is regarded as a problem of assigning syllables to words. The advantages of the proposed method are that there are very few or even no homonyms for multi–syllabic words and that the adjacency relationships among words can be utilized for word identification based on the relaxation technique. The feasibility of this approach is proved by its high character conversion rate.

The following are some suggestions for further study.

(1). collection of more word entries in the dictionary to improve the syllable conversion rate. However, the contents of the dictionary essentially will still be limited. Also there exists trade–off between the dictionary volume and the processing speed.

(2). inclusion of the formation rules of compound words and word groups. Such rules may link lots of single syllables to form multi–syllabic strings, and so reduce syllable conversion ambiguity.

REFERENCES

[1]. "Evaluation Report of Chinese Input Methods and Input Devices," Institute for Information Industry, Taipei, Taiwan, R. O. C., June 1987.

[2]. S. K. Wan, H. Saiton, and K. Mori, "Experiment on Pinyin–Hanzi Conversion Chinese Word Processor," Computer Processing of Chinese and Oriental Languages, Vol. 1, No. 4, pp. 213–224, Nov. 1984.

[3]. W. H. Ho, C. C. Hsieh, K. Mei and C. T. Chang, Automatic Recgnition of Chinese Words, National Taiwan Institute of Technology, Taipei, Taiwan, R. O. C., 1983.

[4]. S. I. Chen, C. T. Chang, J. J. Kuo, and M. S. Hsieh, "The Continuous Conversion

Algorithm of Chinese Character's Phonetic Symbols to Chinese Character," Proceedings of 1987 National Computer Symposium, Taipei, Taiwan, R. O. C., Dec. 1987, pp. 437–442.

[5]. M. Y. Lin and W. H. Tsai, "Removing the Ambiguity of phonetic Chinese input by the Relaxation technique," Computer Processing of Chinese and Oriental Languages, Vol. 3, No. 1, pp. 1–24, May 1987.

[6]. A. Rosenfeld and A. C. Kak, Digital Picture Processing, Vol. 2, Academic Press, New York, 2nd ed., 1982.

[7]. A. Rsoenfeld, R. A. Hummel and S. W. Zucker, "Scene Labelling by Relaxation Operations," IEEE Tran. Syst., Man, Cybern., Vol. SMC–6, pp. 420–431, 1976.

[8]. A. Rosenfeld, "Iterative Methods in Image Analysis," Pattern Recognition, Vol. 10, pp. 181–187, 1978.

[9]. W. S. Rutkowsky, "Recognition of Occluded Shapes Using Relaxation," Computer Graphics and Image Processing, Vol. 19, pp. 111–128, 1982.

[10] S. Peleg, "A New Probabilistic Relaxation Scheme," IEEE Tran. PAMI, Vol. PAMI–2, No. 4, pp. 362–369, 1980.

[11]. O. D. Faugeras and K. Price, "Improving Consistency and Reducing Ambiguity in Stochastic Labelling: an Optimization Approach," IEEE Tran. PAMI, Vol. PAMI–3, pp. 412–424, 1981.

[12]. R. A. Hummel and S. W. Zucker, "On the Foundations of Relaxation Labelling Processing," IEEE Tran. PAMI, Vol. PAMI–5, pp. 267–287, 1983.

[13]. M. Y. Lin, and W. H. Tsai, "A New Approach to On–line Chinese Character Recognition by Sentence Contextual Information using the Relaxation Technique," Proceedings of International Conference on Computer Processing of Chinese and Oriental Languages, Toronto, Canada, August 1988.

[14]. S. I. Hanaki and T. Yamazaki, "On–line Recognition of Handprinted Kanji Characters," Pattern Recognition, Vol. 12, pp. 421–429.

[15]. C. K. Fan, and W. H. Tsai, "Automatic Word Identification in Chinese Sentences by the Relaxation Technique," to appear in Computer processing of Chinese and Oriental Languages.

[16]. I. M. Liu, et al., Frequency Counts of Frequently Used Chinese Words, Lucky Book Co., Taipei, Taiwan, R. O. C. 1975.

[17]. C. Y. Suen, Computational Studies of the Most Frequent Chinese Words and Sounds, World Scientific Publishing Co., Singapore, 1986.

sentence 83
由於愛面子的人太多，
sentence 84
所以應酬的風氣越來越盛，
sentence 85
不管甚麼事，
sentence 86
先吃一頓再說；光吃還不夠，
sentence 87
還得要有于興節目，
sentence 88
於是各種飯店常有變相營業的情形發生，
sentence 89
連帶地促使各種行業蓬勃發展

sentence 147
在童年的記憶裡，
sentence 148
我對外公的印像非常深刻
sentence 149
他常常到各地旅遊，
sentence 150
每到一個地方，
sentence 151
就把當地名產帶回來給我們這些孫兒吃

sentence 17
每逢閱讀的課堂，
sentence 18
大家便沒開眼笑，
sentence 19
欣喜萬分有說不出的快樂，
sentence 20
像是考了第一名一樣。
sentence 21
學校圖書室外面的走廊，
sentence 22
有一排整齊的鞋跪，

sentence 162
於是我加快腳步，
sentence 163
向他奔去，

Figure 1. Some testing results. Underlined ones are errors.

A NEW APPROACH TO QUALITY TEXT GENERATION

Jyun-Sheng Chang
Hwei-Ming Kou

Institute of Computer Science
National Tsing Hua University
Hinchu, Taiwan, Republic of China

1. INTRODUCTION

The need to study text generation is obvious. It is one of the two ways in which people communicate with one another. Thus, if we can simulate this capability computationally, we would be able to use the techniques in many applications such as (1) automatic generation of reports, manuals, and letters, (2) providing a natural language interface from a system to its users, (3) an nature language interpreter for reading and debugging information encoded in some formal notation such as a knowledge base and a software specification.

Text generation is already established as a research area within computational linguistics [Mann 1982]. It has a rather late start compared to other areas in computational lingustics. During 1970's, out of disatification for pre-prepared text (canned text), researchers began to study ways of generating sentences automatically [Goldman 1975, Grishman 1979 and Spapiro 1979]. In the 1980's, the focus has shifted toward the generation of discourse (cohesive text with many sentences, either in producing a monologue or engaging in a dialogue) [Derr-McKeown 1984, Man

1984, McDonald-Pustejovesky 1985 and McKeown 1985] and the methodology used in text generation [Danlos 1984 and Vaughan-McDonald 1986]. Thus far, there have been only a few experimental systems that generate text in a technically interesting way or are based on sound linguistic theories. However the progress in bettering the understanding of human text production and techniques for simulating this capabiliy, is considerable.

2. PREVIOUS WORK

The generally accepted model of text generation consists of mainly two phases: deep generation and surface generation. The deep generation phase takes the representation of meaning or knowledge and produces a sequence of ordered messages. The surface generation phase then convert each of these messages into a sentence.

The deep generation phase determines what to say (content determination) and when to say what (discourse structure). There are two major tasks in surface generation: syntactical choice and lexical choice. A syntactical pattern must be chosen to realize the sentence. And each entity in the message must have a proper wording for it.

Systems requiring text generation include dialogue systems, question-answering systems, systems that validate natural language input by paraphrasing, expert systems with explanation capability, story and document generation systems.

2.1 CONTENT DETERMINATION

The first task in text generation is to determine what to say which generally depends on the context and purpose of the system. A randon sentence generator obviously could not care less about the content that it produces [Fredman 1969 and Parisi-Giorgi 1985]. The content of a paraphrase is whatever the user has just input [Goldman 1975 and McKeown 1979]. In a question-answering system, the question is parsed, tranformed into some kind of query for the underlaying database. And the result of the query is the content of the answer [Grishman 1979]. McDonal and Conklin pointed out that in describing a picture, a good stategy is to say whatever are most salient [Conklin-McDonald 1982]. Sometimes, the content is diminishable when the purpose of text generation is merely to say something and passes as a person [Boden 1976].

The information resulting from this phase may be represented in various forms: predicate calculus [Grishman 1979], conceptual dependency structures (CD) [Goldman 1975], semantic nets [Minsky 1981, Simmons-Slocum 1972 and McKeown 1985], or frames [Woods 1975 and Mauldin 1984].

2.2 DISCOURSE STRUCTURE

There are many systems in the literature capable of generating multi-sentence text: Simmons and Slocum built a system to generate English sentences from semantic networks [Simmons-Slocum 1972]. Davey's PROTEUS takes the sequence of moves played in a tic-tac-

toe game and produces a paragraph of commentary of the game [Davey 1975, Mann 1982, and Richie 1984]. The BLAH system generates multi-sentence explanation of the reasoning process taken by the system [Weiner 1980]. Meehan's TALESPIN produces multi-paragraph stories [Meehan 1977]. However these systems focus on knowledge needed for generation and its representation. The organization of text is either given (as in PROTEUS) or fixed (as in TALESPIN).

The systems that have a module determining discourse structure and represent the knowledge about discourse structure explicitly, include the Knowledge Delivery System (KDS) [Mann-Moore 1979 and 1981], BLAH, the explanation module for an expert system [Weiner 1980], and TEXT, a system that answers questions about the structure of a database [McKeown 1985].

BLAH mimics the simple way that people use to explain something or justify a statement to organize the text for explanation. TEXT goes considerably behind BLAH's simple formulation of discourse structure in the three ways: (1) TEXT includes discourse strategies in the form of ATN graphs, for many more discourse goals in addition to explanation. (2) These discourse strategies are based on naturally occurring text. Thus they reflect the discourse patterns which are effective and most often used. (3) Due to the nondeterminism of ATN's, these discourse strategies capture a notion of variability which is resolved by focus of attention and semantic relations in the text. The text generated in a top-down, goal-directed fashion, looks well-structured,

cohesive, and with a purpose.

Unlike BLAH and TEXT, KDS lacks an explicit representation of knowledge about discourse structure. It employs a rule-based planning strategy to organize information into discourse. The rules used have a strong bottom-up, data-driven flavor.

Finally there is the Rhetoric Structure Theory (RST) [Mann 1983], a descriptive theory on discourse structure. The author claimed that it can be turned into a constructive process through the use of a planning strategy with various rhetoric structures reguarded as means of realizing the goal of text generation. But that remains to be seen.

2.3 SEMI-SURFACE GENERATION

Going from deep generation to surface generation, there is an issue that needs to be resolved. That is the problem of how much information to put in a sentence: Whether to put a lot of information in one complex sentence or put them in 2 or 3 simple sentences. We call this consideration the *semi-surface generation*.

Davey used a fixed strategy in this regard [Davey 1975]. Derr and McKeown recognized the interplay of this decision and shifting in focus of attension in the text [Derr-McKeown 1984]. McDonald studied the encyclopedia articals on African tribes and found that this decision has a lot to do with the prose style of the text generated [McDonald 1985].

2.4 SURFACE GENERATION

There are essential three methods for surface generation: canned text, template, and direct translation. Error message generated by compilers is typical example of canned text. Early expert systems use templates to generate explanations; a template is associated with each rule and the explanation consists of the templates associated with the rules fired. Canned text and templates are fast, easy to construct but must be anticipated in advanced. Consistency and closure are difficult to achieve.

In order to generate sentences of higher quality consistently and to ensure closure, one needs to translate directly a message into a sentence using some form of grammar. Three components are needed for this process: (1) a formal representation of the sentence structure in the language, (2) a dictionary containing various information such that proper words may be chosen to represent concepts and entity conveyed in the message, (3) a way of doing syntactical and lexical choice.

Several grammar formalisms have been used for surface generation: (1) Systemic grammar [Halliday 1973, 1976, and 1985], (2) Transformational grammar founded by Noam Chomsky, (3) ATN grammar [Woods 1970], (4) The Linear String Parser (LSP) [Sager 1981].

The BABEL paraphrasing system uses discriminate nets to help select lexical items and rely on an ATN grammar to generate sentence structure [Goldman 1975]. The The question-answering

system by Grishman uses LSP [Grishman 1979] as the grammartical formalism for generation as well as parsing. The sentence generator Kafka [Mauldin 1984] in the XCON/XSEL system uses transformational grammar [Mauldin 1984]. So do the CO-OP paraphraser [McKeown 1979] and a system that generates English sentences for instructional purpose [Bates-Ingria 1981]. An explanation module for a student advisor expert system uses functional grammar [Winograd 1983], which a grammatical formalism based on many ideas from systemic grammar.

System grammar is used by the sentence generators in PROTEUS, the PENMAN/Nigel system, and Patten's system [Davey 1975, Mann 1983, Matthiessen 1983, and Patten 1985]. There is a growing consensus among researchers that systemic grammar is the grammar of choice for text generation.

2.5 METHODOLOGY FOR TEXT GENERATION

Most text generation systems are a one-pass process through the content determination, deep generation, semi-surface generation, and surface generation phases. Vaughan and McDonald observed that people write and then rewrite again and again; revision seem to be a large part of writing process for people. Thus they propose a *revisional model* of text generation to simulate this human strategy of writing [Vaughan-McDonald 1986].

Within this model, text is first generated in a straightforward fashion, without attempting much global arrangement. Then the

system iterates through a process of recognition, editing, and regeneration. The recognition phase essentially finds out the places where changes can be made to enhance cohesion of the text. The editing and regeneration phases implement these changes. The authors argued that using this kind of model will reduce the complexity of the text generator. The KDS system follows the revisional model. A revisional module is also planned for the Penman system. However, the feasibility of this model is difficult to assess at this point of time.

Danlos observed that the syntactical choice is sometimes influenced by the choice of lexical items and suggested against strict separation of lexical and syntactical choices [Danlos 1984].

3. A MODEL FOR TEXT GENERATION

This section presents the new approach taken in our text generation system. The system is intended as a test bed for experimenting with new ideas and for understanding the text generation needs in different environments and for different languages.

Our objectives in implementing the system include (1) to base the system in solid linguistic theories [Halliday 1973, Halliday-Hasan 1976, and Hudson 1971], (2) the ability to handle situations where the information needed for text generation is not pre-stored within but rather needs to be acquired from the user, (3) the ability to generate text whose pattern may be determined more or

less beforehand (goal-driven) [McKeown 1985] or may be dictated by the information available (data-driven), (4) the adaptability to different styles intended for different kinds of users or purposes [McDonald-Pustejovesky 1985].

Currently, we are considering the following as our domains for text generation: (1) English business letters [keshi 1987], (2) User's manuals for computer systems in English and Chinese, and (3) A paragraph-level target language (Chinese or English) generator for a machine translation system.

In order to achieve the above goals, we take the following approach to implement our system: (1) The system uses a representation which can reflect existence as well as the lack of information. (2) In stead of producing an ordered sequence of messages, the deep generator produces a partilly ordered sequence of propositions with functional marking representing the rhoritical or cohesive relation among the elements in the propositions. (3) The system uses a hybrid strategy which covers the whole spectrum of goal-driven and data-driven strategies. (4) An intermediate phase, called *semi-surface generation*, between deep generation and surface generation, is included, to serve the purpose of reflecting different prose styles. (5) The surface generator produces sentences using a systemic grammar [Hudson 1971].

3.1 CONTENT DETERMINATION

We propose using a frame-based knowledge base to represent knowledge known a priori and as a information acquisition scheme.

If all there is to say is known before text generation, then the representation will be all filled up. On the other hand, if there is information yet unknown at generation time, there will be unfilled slots in the frame system. Then this knowledge base can be used to drive an input module to acquire needed information from the user. Thus we can use a uniform representation for situations where the information needed to generate the text is either pre-stored in the system or needed to be acquired from the user.

3.2 DEEP GENERATION

We propose a deep generation method which is inspired by Discourse Strategies (DS) in McKeown's TEXT system and Mann's Rhetoric Structure Theory (RST). There are two processes in our deep generation phase: a top-down, goal-directed process and bottom-up, data-driven process.

The top-down process uses an ATN-like representation of overall strategy of determine content and organization of the text and go through the recursive representation like travelling a tree from the root (the goal or purpose of the text) down to its leaves. A leaf of the tree is either a single proposition (message) to be converted to a sentence or a demo to start an instance of the bottom-up process.

The bottom-up process uses rules based on (1) focus of attention, (2) identities among the elements of propositions, and (3) semantic relavance between propositions, to pick out propositions and give them a partial order. This is not only a process of finding what propositions should be included in a sentenec but also a process finding and marking the cohesive links in the content. These markings are subsequently used in surface generation for such activitie as pronomial, demonstractive, verbal, and clausal substitutions, ellipsis, selection of conjunction, and lexical choice [Halliday-Hasan 1976]. The result of this phase is an totally ordered sequence of packages where each package contains a set of marked propositions in partial order.

This 2-phase , mixed-strategy approach covers the wole spectrum from the strictly goal-directed strategies to the strictly data-driven strategies. Thus the system can be tuned to adapt to diiferent text generation situations.

3.3 SEMI-SURFACE GENERATION

The partially ordered propositions produced in the deep generation phase subsequently go through a filtering phase. We propose a rule-based approach to determine whether to pack propositions in a sentence or to leave them along so one proposition will produce one sentence in the final surface generation phase.

There should be two kinds of rules: (1) rules that pack

propositions based on shifting of focus of attention and degree of identity among propositions (2) meta rules that assign priorities to the first kind of rules.

The first kind of rules will enhance local cohesion in the text, while the second kind of rules can be used to reflect prose style.

3.4 SURFACE GENERATION

Systemic grammar is used in the surface generation for the following reasons: (1) It is based on function of language and emphasizes the mechanism of choice according to function. That corresponds closely to the nature of the generation process. (2) The phases before surface generation produce a lot of functional feactures on which the system grammar is mainly structured. (3) Systemic grammar encode lexical choice and syntactial choice in one notation. Thus the problem pointed out by Danlos can be handle more easily using systemic grammar.

4. CONCLUSION

We have proposed in this paper a framework for text generation which is based the system in solid linguistic theories. The system can handle different situations and generate text either by following a pre-determined pattern or by adjusting to what is present to be conveyed. And it is possible to tune the system so that text of different styles can be generated to serve different kinds of users or purposes.

References

Appelt 1983 *Telegram: A Grammar Formalism for Language Planning*, Proceedings of the 21st Annual Meeting of the ACL pp. 74-78.

Bates 1981 *Controlled Transformational Sentence Generation*, Proceedings of the 19th Annual Meeting of the ACL, pp. 153 158.

Boden 1976 *Artificial Intelligence and Natural Man*, Basic Books, New York, pp. 95 111.

Davey 1975 *Discourse Production*, Edinburgh University Press, Edinburgh.

Danlos 1984 *Conceptual and Linquistic Decisions in Generation*, Proceedings of the 22nd Annual Meeting of the ACL, (COLING 84), pp. 501-504.

Derr-McKeown 1984 *Using Focus to Generate Complex and Simple Sentences*, Proceedings of the 22nd Annual Meeting of the ACL, (COLING 84), pp. 319-326.

Goldman 1975 *Sentence Paraphrasing from a Conceptual Database*, Comm. ACM 18:2, pp. 96 106.

Granville 1984 *Controlling Lexical Substitution in Computer Text Generation*, Proceedings of the 22nd Annual Meeting of the ACL, (COLING 84), pp. 381-384.

Grishman 1979 *Response Generation in Question-Answering Systems*, Proceedings of the 17th Annual Meeting of the ACL, pp. 99-101.

Halliday 1973 *Explorations in the Functions of Language*, Edward Arnold, London.

Halliday 1975 *System and Function in Language*, Oxford University Press, London.

Halliday 1985 *An Introduction to Functional Grammar*, Edward Arnold, London.

Halliday-Hasan 1976 *Cohesion in English*, Longman, London.

Hudson 1971 *English Complex Sentences: an introduction to systemic grammar*, North-Holland, Amsterdam.

Keshi 1987 *A Knowledge-based Framework in an Intelligent Assistant for Making Document*, Abstracts of the International Conference on AI, (AI 87 JAPAN), Osaka, Japan, pp. 286-294.

Mann-Moore 1979 *A Snapshot of KDS: A knownledge Delivery System*, Proceedings of the 17th Annual Meeting of the ACL, pp. 51-52.

Mann-Moore 1981 *Computer Generation of Multiparagraph English Text*, AJCL 7:1, pp. 17-29.

Mann 1982 *Applied Computational Linguistics in Perspective: Proceedings of the Workshop - Text Generation*, AJCL 8, pp. 62-69.

Mann 1983 *An Overview of the Nigel Text Generation Grammar*, Proceedings of the 21st Annual Meeting of the ACL, pp. 79-84.

Mann 1984 *Discourse Structures for Text Generation*, Proceedings of the 22nd Annual Meeting of the ACL, (COLING 84), pp. 367-375.

Matthiessen 1983 *Systemic Grammar in Computation: The Nigel Case*, Proceedings of the 22nd Annual Meeting of the ACL (COLING 84), pp. 155-164.

Mauldin 1984 *Semantic Rule Based Text Generation*, Proceedings of the 21st Annual Meeting of the ACL (COLING 84), pp. 376-380.

McCoy 1982 *Augmenting a Database Knowledge Representation for Natural Language Generation*, Proceedings of the 20th Annual Meeting of the ACL, pp. 121-128.

McDonald 1985 *A Computation Theory of Prose Style for Natural Generation*, Proceedings of the 2nd Conference of the European Chapter of the ACL, pp.187-193.

McDonald-Conklin 1982 *Salience: The Key to the Selection Problem in Natural Language Generation*, Proceeding of the 20th Annual Meeting of the ACL, pp. 129-135,

McDonald-Pustejovesky 1985 *A Computational Theory of Prose Style for Natural Language Generation*, Proceedings of the 2nd Conference of the European Chapter of the ACL, pp.

187-193.

McKeown 1985 *Discourse Strategies for Generating English Text*, Artificial Intelligence 27, pp. 1-41.

Minsky 1981 *A Framework for Representing Knowledge*, in *Mind Design*, edited by J. Haugeland, MIT Press, pp. 95-128.

Parisi-Giorgi 1985
GEMS: A Model of Sentence Production, Proceedings of the 2nd Conference of the European Chapter of the ACL, pp. 258-262.

Patten 1985 *A Problem Solving Approach to Generatin Text from Systemic Grammar*, Proceedings of the 2nd Conference of the European Chapter of the ACL, pp. 187-193.

Ritchie 1984 *A Rational Reconstruction of the Proteus Sentence Planner*, Proceedings of the 21st Annual Meeting of the ACL, (COLING 84), pp. 327-329.

Sager 1981 *Natural Language Information Processing*, Addison-Wesley, Reading.

Shapiro 1979 *Generalized Augmented Transition Network Grammars for Generation from Semantic Networks*, Proceedings of the 17th Annual Meeting of the ACL pp. 25-30.

Simmons-Slocum 1972
Generating English Discourse from Semantic Networmks, Comm. ACM 13:1, pp. 15-30.

Vaughan-McDonald 1986
A Model of Revision in Natural Language Generation Proceedings of the 24th Annual Meeting of the ACL, pp. 90-96.

Weiner 1980 *BLAH, A System Which Explains its Reasoning*, Artificial Intelligence 15, pp. 19-48.

Winograd 1983 *Language as a Cognitive Process, Volume 1: Syntax*, Addison-Wesley, Reading.

Woods 1970 *Transition Network Grammars for Natural Language Analysis*, Comm. ACM 13:10, pp. 591-606.

Woods 1975 *What is in a Link: Foundation for Semantic Network*, in *Studies in Cognitive Science*, D.G. Bobrow and A.M. Collins, eds., Acdemic Press, New York, pp. 35-82.

Functional Representation of Query Sentences and Meaning Determination of Elliptical Sentences

Hsi-Jian Lee & Shin-Jiang Hwang

李錫堅　　黃詩建

National Chiao Tung University

國立交通大學

Proceedings of ROCLING I(1988)
R.O.C. Computational Linguistics Workshops I pp 179-210

中華民國第一屆計算語言學研討會論文集 179-210 頁

Functional Representation of Query Sentences and Meaning Determination of Elliptical Sentences

Abstract: A dialogue model is provided to describe the contents of a dialogue process between a user and a database management system. This model can be used as an intermediate representation between query sentences in a natural language and the underlying database query language. It is also capable of keeping the dialogue information, including user query sentences and the associated responses, for later processing of elliptical sentences.

A query sentence is decomposed into two components, a query phrase and a data description phrase. The functional representations of both components are analyzed in detail. Five types of representations for elliptical sentences, including subsetting, ordinal, remaining, projection, and substitution, are presented. We also detailedly discuss the determination of full meanings of elliptical sentences based on both the dialogue convention and functional representations.

Keywords: dialogue model, query sentence, functional representation, elliptical sentence

1. Introduction

To provide a friendly user interface in a database management system for information retrieval is an important task. Database query languages[1] are commonly adopted for information retrieval. They do provide rigid notations for a user to state his query sentences precisely without worrying about the physical structure of the database. But in order to express query sentences skillfully, the user must have some knowledge about query languages and the structure of the underlying database systems. It is sometimes very inconvenient for a novice. Therefore, it is necessary to provide a presentation more intuitive than database query languages. According to the experiments of Hendrix *et al.*[2], natural languages as user interfaces for information retrieval can satisfy this need. They can also shorten learning time and thus encourages the using of database systems. In addition to providing more intuitive expressing method, natural languages can save the user from the trouble of dealing with the physical and even logical structure of the database.

When we use natural languages, it is necessary to provide a model to describe the meaning of a query sentences in a natural language and to organize the dialogue information so that query sentences and system responses can be kept. In this paper, we present a dialogue model to describe the information involved in a dialogue process in a Chinese Intelligent Database Assistant (CIDA) for retrieving library information. This model contains a list of items composed of user query sentences and system responses. The purpose of keeping information of the dialogue process is to enable the user to state query sentences referring either to previous responses or to *elliptical sentences*, whose meaning can be determined from a previous query sentence. For example, after the user issues the following query,

(1) qǐng xiǎnshì A.I. fāngmiàn de qīkān.

(Please display the title of journals related to Artificial Intelligence.)

he can state the following elliptical query sentence,

(2) DBMS?

instead of

(3) qǐng xiǎnshì DBMS fāngmiàn de qíkān.

The problem about representation of sentences, usually very complicated, is a typical issue in natural language processing. So long as the universe of discourse is limited in a special domain, it is possible to devise a compact, precise and perspicuous representation. In the model proposed in this paper, the representation of a query sentence is called a *functional form,* which takes into account mainly the function of a query sentence. The syntactic structure is also reserved for resolving the full meaning of an elliptical query, which can be exemplified as follows.

(4) qǐng xiǎnshì ACM chūbǎn de qíkān zhōng yǔ A.I. yǒu guān de.
(Please display the title of journals published by ACM and related with A.I.)

(5) ACM chūbǎn de qíkān zhōng yǒu nǎxiē yǔ A.I. yǒuguān ?

(6) yǔ A.I. yǒu guān de qíkān zhōng yǒu nǎxiē shì ACM chūbǎn de ?

The semantics of these three sentences are the same. For the representations in functional form, abbreviated as *functional representations* hereafter, sentences (4) and (5) should have the same representation since the processing about either the current sentence or subsequent sentences are all the same. However, they should have different representation from sentence (6). It can be seen from the situation when sentence (7) below follows them.

(7) nǎxiē shì IEEE chūbǎn de ?
(Which are published by IEEE ?)

If sentence (7) follows sentence (6), it means

(8) yǔ A.I. yǒuguān de qíkān zhōng yǒu nǎxiē shì IEEE chūbǎn de ?
(Among the journals related to A.I., which ones are published by IEEE ?)

But it is meaningless if sentence (7) follows sentences (4) and (5).

All sample sentences in CIDA, as shown in Fig. 1, are classified into two classes: *basic sentences* and *elliptical sentences*. In the class of basic sentences, those without "*e*" in the sentence number, their meanings can be interpreted from themselves. On the other hand, in the class of elliptical sentences, those with "*e*" attached to the sentence number, their meanings must be interpreted by referring to the previous context.

(1) lièchū suǒyǒu ACM chūbǎn de qǐkān.
(2) lièchū 3 zhǒng IEEE chūbǎn de zázhì zhōng yǔ réngōng zhìhuèi yǒuguān de.
(3) qǐng xiānshì zhìliàokùxìtǒng fāngmiàn de shūjí de jiàqián jí chūbǎnshāng.
(4*e*) lièchū nàxiē jǔ zhuānjiā xìtǒng yǒuguān de zázhì de chūbǎn niándaì.
(5*e*) lièchū qǐzhōng Knuth xiě de.
(6*e*) lièchū tāmende neìróng.
(7*e*) lièchū qǐzhōng dì 4 běn de neìróng.
(8*e*) qǐng lièchū qǐyúde.
(9*e*) lièchū qǐyú yǔ réngōng zhìhuèi yǒuguān de.
(10) yǒu nǎxiē Winston xiě de shū ?
(11) nǎxiē zázhì shì ACM chūbǎn de ?
(12) North Holland chūbǎn de zázhì zhōng yǒu nǎxiē yǔ réngōng zhìhuèi yǒuguān ?
(13*e*) qǐzhōng yǒu nǎxiē IEEE chūbǎn de zázhì ?
(14*e*) qǐzhōng yǒu nǎxiē zázhì shì IEEE chūbǎn de ?
(15*e*) háiyǒu nǎxiē shì Knuth xiě de ?
(16*e*) háiyǒu nǎxiē réngōng zhìhuèi fāngmiàn de zázhì shì IEEE chūbǎn de ?
(17) The Art of Computer Programming de zhuòzhě shì sheí ?
(18*e*) neìróng shì shéme ?
(19) The Art of Computer Programming shì sheí xiě de ?
(20) The Art of Computer Programming fàngzaì nǎlǐ ?
(21*e*) chūbǎn daù dìjǐqí ?

Fig. 1. Sample Sentences in CIDA

The intelligent information systems incorporating natural languages as a front end of database systems include GUS[3], TEAM[4], KID[5], LUNAR[6], LDC[7], FRED[8] and LADDER[2]. Among all the aspects of the above systems, we are interested in the meaning representations and their treatments of elliptical inputs. Only GUS, LUNAR and LADDER, among these systems, can accept elliptical inputs.

LADDER uses syntax tree based on the semantic grammar as its meaning representation. An elliptical input can be accepted only if its syntax tree is analogous with a partial tree of a previous sentence. This is the most common type of ellipses. GUS uses frames to represent query sentences and controls the processings. The system asks the user a planned sequence of questions in order to obtain the full specifications of user query sentences. Though it can understand a few mixed initiated utterances by key-word matching, the overall interaction is guided by the system. LUNAR uses extended notational variants of the ordinary predicate calculus as a meaning representation language; it determines the meanings of elliptical and anaphoric expressions according to both syntactic structure and the language, which is in logical form.

There are some researches, though not real systems, focusing on meaning representation for information retrieval and on appropriateness of the meaning representation for determination of ellipses. Nash-Webber[9] proposed a formal meaning representation, and argued that logical meaning representation is superior to semantic network, especially in determination of ellipses. Horrigan[10] tested his dialogue model for the real dialogue between passengers and a clerk at an information booth in a train station. Spiegler[11] proposed a notation to represent user query sentences but he did not address the problems of ellipses and anaphora.

For an elliptical sentence, we should determine its full meaning by applying the syntactic and semantic information of both previous and current sentences and the domain knowledge. The approaches for the determination problem can be classified into three kinds. The first approach, adopted in GUS[3], is to construct the complete sentence from a sentence fragment and then parse it. The second one, adopted in LADDER and INLAND[2], is to match the elliptical syntactic structure of an elliptical sentence with that of a previous meaning complete sentence. After an analogous pattern is found, it is used to replace the current syntactic structure to construct a com-

plete syntactic structure. The last one, adopted in this system as well as LUNAR[6], infers the meaning of an elliptical sentence from the functional representations of the context and the current input. In the determination process, source sentences and syntactic trees are not involved.

Sec. 2 describes the method to represent query sentences in the functional form. The analysis of utterances based on their functions is also presented. Sec. 3 shows the functional representations of ellipical sentences. Sec. 4 describes the determination of elliptical sentences according to their function types. The full meaning of an elliptical sentence is evaluated by taking into account the structures of the current and previous context, responses and the dialogue convention. Some conclusions are given in Sec. 5.

2. The Dialogue Model

The dialogue model is actually a representation of the dialogue process. A typical dialogue process contains a series of exchanges, each of which composed of a query sentence and its corresponding response. Following the principle in the design of functional form, two sentences should be mapped into the same meaning representation if there is no difference in the processings of subsequent query sentences; conversely, they should be mapped into different representations.

2.1 Overall Description

The full specification of the dialogue model is given in Appendix. The top level specifications are :

DialogProcess == [Exches]

Exches == Exch | Exch, Exches

Exch == [Query, Response]

That means a dialogue process is composed of a series of exchanges and each exchange contains a query part and a response part. The response part, containing responsed information of the query sentence, is used to reduce the need of repetitive

access to the database and to resolve elliptical expressions with numeric determiners. This can be seen from the following two successive query sentences.

(9) qǐng xiǎnshì shuǒyǒu ACM chūbǎn de qīkān.

(Please display all titles of journals published by ACM.)

(10) qǐngwèn dì 3 běn fàng–zài nǎlǐ ?

(Where does the third one put?)

In this paper, we focus mainly on the representation of the query part. It is specified as

Query == basic(Complete) | elliptical (Ellipsis, Complete).

It means that a query sentence may be either a basic sentence or an elliptical sentence. There are two constituents in the functional representation of an elliptical sentence, "*Ellipsis*" and "*Complete*". "*Ellipsis*" denotes the original meaning-incomplete representation of an elliptical sentence and "*Complete*" represents its corresponding meaning-complete representation, derived from "*Ellipsis*" and the context.

Generally speaking, a query sentence is composed of two components: *a query phrase* and *a data description phrase*, where the latter phrase determines entities from which some information should be retrieved, and the former phrase determines the query type of the sentence; that is, it determines the information which should be retrieved from entities described by the data description phrase. For example, in the following sentences :

(11) Artificial Intelligence cóng nǎ–yi–qī dìng–qǐ ?

(What is the issue number from which Journal of Artificial Intelligence is subscribed ?)

(12) On Conceptual Modelling fàng–zài nǎlǐ ?

(Where does 'On Conceptual Modelling' put ?)

(13) yǒu jǐ zhǒng réngōngzhìhuì fāngmiàn de zázhì ?

(How many kinds of journals are related to Artificial Intelligence ?)

the underlined phrases are query phrases and the others are data description phrases. In the design of functional form, it is very important to decompose a sentence into a query phrase and a data description phrase.

2.2 Representations of Query Phrases

As mentioned above, the functional form is mainly used for describing the function of a query sentence. In the following, we will describe three functional types of query phrases: presentation, aggregation and predicate.

2.2.1 Presentation Query Type

A presentation query phrase is specified to show some information on line. This is the most typical of query phrases for information retrieval. Some examples of the presentation query type are "qǐng xiǎnshì" (please display), "qǐng lièchū" (please list), "yǒu nǎxiē" (which ones), "yǒu nǎ jǐ zhǒng" (which kinds), "yǒu nǎ jǐ běn" (how many books), etc. This type of query sentences can be specified as

Complete == present(DataSetDescriptor, Attributes),

where "DataSetDescriptor" denotes the representation of the data description phrase and will be described in the next subsection, and "Attributes" denotes the information to be retrieved from items of the data constrained by "DataSetDescriptor". The following sentences are typical examples.

(14) qǐngwèn Natural Language Processing shì shéi xiě de ?

(Who is the author of Natural Language Processing ?)

(15) qǐng xiǎnshì Natural Language Processing de zhùzuò .

(Please display the author of Natural Language Processing ?)

(16) yǔ Computer Graphics yǒuguānde zázhì yǒu nǎxiē ?

(How many journals are related to Computer Graphics ?)

(17) yǒu nǎxiē guānyǔ Computer Graphics fāngmiàn de zázhì ?

(18) qǐng lièchū yǔ Computer Graphics yǒuguān de zázhì.

2.2.2 Aggregation Query Type

An aggregation query phrases is usually used to enquire the number of items in a data set. It is specified as

Complete == count(DataSetDescriptor, Unit),

where "Unit" may be an element of the set { zhǒng, běn, qǐ, lèi, . . . }. Query sentences of this type include, for example, "yǒu jǐ běn", "yǒu jǐ zhǒng" and "yǒu jǐ qǐ". The following sentence is a typical example:

(19) yǒu jǐzhǒng A.I. fāngmiàn de shū ?

(How many kinds of books are related to A.I. ?)

2.2.3 Predicate Query Type

Predicate query phrases usually appear in Yes/No questions, and are represented as predicates in functional form. In this paper, we provide only a few predicates because most Yes/No questions can be replaced pragmatically by WH questions[2]. This fact will be explained in Sec. 3 in more detail. A sentence containing "yǒuméiyǒu" is represented as "exist(DataSetDescriptor)", "shìbúshì Borrower jiè de" as "lentBy(DataSetDescriptor, Borrower)", and "yǒuméiyǒu bèi jièzǒu" or "shìbúshì bèi jiè le" as "lent(DataSetDescriptor)". The following is a typical example:

(20) yǒuméiyǒu A.I. fāngmiàn de shū ?

(Is there any book related to A.I. ?)

2.3 Representations of Data Description Phrases

There is a difference between the representation of a data description phrase and that of a query phrase: the mapping from a query phrase onto its representation in functional form ignores the syntactic structure information while the mapping from a

data description phrase preserves some syntactic structure information. The data description phrases are divided into five types as illustrated below.

2.3.1 Restriction Type

The restriction type of data description phrases is the most common type which can be specified as

DataSetDescriptor = restrict(DataSetDescriptor, Constraint) | cida | dataSet(N),

where "DataSetDescriptor" is defined recursively and may denote either the overall database, "cida", or a data set corresponding to a previous sentence specified by a number, "dataSet(N)"; "Constraint" denotes the conditions derived from the data description phrase. For a basic sentence, the typical "DataSetDescriptor" is "restrict(cida, Constraint)", which means the data in the underlying database satisfying the constraint "Constraint". If a "DataSetDescriptor" contains "dataSet(N)", it denotes an elliptical sentence; for example, "restrict(dataSet(5), Constraint)" denotes the data in the data set of the fifth query sentence satisfying "Constraint". The full specification of constraints is given in Appendix. Two phrases of this type and their corresponding representations are shown as follows.

(21) "suǒyǒu ACM chūbǎn de zázhì "
(All journals published by ACM)
restrict(cida, and([publisher(ACM), bookType(journal)]))

(22) "yǔ A.I. yǒuguān de shū zhōng 1980 nián hou`chūbǎn de "
(All books related to A.I. and published after 1980)
restrict(restrict(cida, and([field(ai), bookType(book)])),
gt(publishedYear, year(1980)))

Compare the following two phrases and their corresponding representations.

(23) yǔ A.I. yǒuguān de zázhì zhōng ACM suǒ chūbǎn de

(Among the journals related to A.I., which are published by ACM?)

restrict(restrict(cida, and([field(A.I.), bookType(journal)])),

publisher(ACM)),

(24) ACM suǒ chūbǎn de zázhì zhōng yǔ A.I. yǒuguānde

(Among the journals published by ACM, which are related to A.I. ?)

restrict(restrict(cida, and([publisher("ACM"), bookType(journal)])),

field(ai)).

From this comparison, we can see that the syntactic structure of data description phrases is preserved. The reason why the two phrases have been mapped onto different representations has already been explained in Sec. 1.

2.3.2 Indefinite Specification Type

In English, "some" is used to express indefinite specification. In this paper an indefinite specification phrase is defined syntactically as

<Indef-Spec> :== jǐ <unit> | <Number> <Unit>,

<Unit> :== běn | zhōng | cè | qǐ,

Phrases like "5 běn", "3 zhǒng" and "4 qǐ" belong to this catogory. In functional form, it is specified as

DataSetDescriptor == some(DataSetDescriptor, Unit, Number),

where "DataSetDescriptor" in the right hand side denotes the data set modified by the indefinite specification phrase. For example,

(25) qǐng xiǎnshì 5 běn Knuth suǒxiě de shū.

(Please show 5 Knuth's books.)

is represented as

DataSetDescriptor = some(restrict(cida, and([author(Knuth),

bookType(book)])), ben, 5).

2.3.3 Ordinal Specification Type

An ordinal specification phrase appears only in an elliptical sentence and is used to specify one or more definite items of the responsed part of a previous sentence. It is described as

DataSetDescriptor == ordinal(DataSetDescriptor, Unit, Orders)

where "Orders" is a list of numbers, each of which denotes an ordinal or an index of the referred sentence in context, and "DataSetDescriptor" is derived by considering the context and will be explained in Sec. 4. The phrase, for example,

(26) dì 3 běn yǔ A.I. yǒuguān de shū.

(The third book related to A.I.)

is represented as

ordinal(dataSet(N), ben, [3]),

and the phrase

(27) qízhōng dì 4 běn

(Among these, the forth one)

is represented as

ordinal(dataSet(N), ben, [4]).

2.3.4 Remaining Type

A remaining data description phrase also appears only in an elliptical sentence and is used to specify the data which would be determined from previous sentences. It is specified as

DataSetDescriptor == diff(BaseSet, Complements)

BaseSet == DataSetDescriptor

Complements == dataSet([Numbers]).

This specification means that the remaining data description phrase specifies the data derived by substracting the data in "Complements" from "BaseSet". Some phrases of

this type are shown as follows.

(28) hái yǒu nǎxiē

(How many remained ?)

(29) qíyú yǔ A.I. yǒuguān de

(Others related to A.I. ?)

(30) qíyú de

(Others ?)

Let us look at the following scenario.

(31) A.I. fāngmiàn de zázhì zhōng yǒu nǎxiē shì ACM chūbǎn de ?

(Among the journals related to A.I., which are published by ACM ?)

(32) nǎxiē shì IEEE chūbǎn de ?

(Which are published by IEEE ?)

(33) nǎxiē shì NorthHolland chūbǎn de ?

(Which are published by North Holland ?)

(34) qíyú de nē ?

(How about the others)

The remaining phrase of sentence (34) obviously means all the journals related with A.I. except those published by ACM, IEEE and North Holland. Thus it is represented as

diff(restrict(cida, and([field(A.I.), bookType(journal)])),

dataSet([31,32,33])).

3. Functional Form of Elliptical Sentences

The strategy for representing an elliptical sentence is to map it onto an elliptical functional form with partial meaning and then construct its complete functional form from the functional forms of the context. There are five classes of ellipses in CIDA, including subsetting, ordinal, remaining, projection, and substitution. They are

categorized according to their functions.

3.1 Subsetting Ellipses

A subsetting ellipsis specifies a subset of the responses of a previous sentence. The subset satisfies the additional constraints in the elliptical sentence. For instance, the phrase "qǐzhōng nǎxiē shì ACM chūbǎn de" (Which are published by ACM?) designates the books or journals published by ACM in the responses of a previous sentence.

The representations of subsetting ellipses are derived from the elliptical sentences containing such subpatterns as "qǐzhōng nǎxiē . . . yǔ . . . yǒuguān" (Among these, which are related to ... ?) or "qǐzhōng . . . suǒ xiě de shì nǎxiē" (Among these, which are written by ...?). The phrases are represented as

RefExp == subset(Constraint, Attributes),

where "Constraint" and "Attributes" mean the same as before.

Two sample phrases and their corresponding functional representations are shown as follows.

(35) qǐzhōng nǎxiē yǔ A.I. yǒuguān ?

(Among these, which are related to A.I. ?)

subset(field(A.I.), ?Attributes),

where "?Attributes" denotes omitted specification.

(36) qǐzhōng ACM suǒ chūbǎn de fàng–zài nǎlǐ ?

(Among these, where do the ones published by ACM put ?)

subset(publisher(ACM), location),

3.2 Ordinal Ellipses

Ordinal ellipses are derived from elliptical sentences containing such subpatterns as "dì N běn" (the nth one), "dì N zhǒng" (the nth kind) and "qǐzhōng dì N lèi" (the

nth type among these). Phrases of this type identify one or more items in the responses of a previous sentence, either basic or elliptical. They are represented as

RefExp == ordinal(Constraint, Orders, Attributes, Unit).

In the following, some sample phrases and their representations are shown.

(37) qǐzhōng dì 3 zhǒng fàng zài nǎlǐ ?

(Where is the third kind put on ?)

ordinal(?Constraint, [3], location, kind)

(38) dì 4 běn A.I. fāngmiàn de shū

(The 4th book related to A.I. ?)

ordinal(and([field(A.I.), bookType(book)]), [4], ?Attributes, ben)

(39) dì 3,4,6 běn

(The third, fourth and sixth books)

ordinal(?Constraint, [3, 4, 6], ?Attributes, ben)

2.4.3 Remaining Ellipses

Remaining ellipses are the elliptical sentences containing such subpatterns as "qǐyúde" (others), "qǐtā (others), "háiyǒu nǎxiē", (others ?),etc.. A remaining phrase specifies the differences of two sets. It is formally represented as

RefExp == complement(Constraint, Attributes).

A case in which the phrases of remaining type appear has been illustrated in Sec. 2.3.4. Other cases will be illustrated in Sec. 4. Here we show some sample phrases and their representations.

(40) qǐyúde zuòzhě

(Other authors ?)

complement(?Constraint, author)

(41) háiyǒu nǎxiē DBMS fāngmiàn de shū

(How many other books related to DBMS ?)

complement(and([field(DBMS), bookType(book)]), ?Attributes)

3.4 Projection Ellipses

A projection ellipsis projects an attribute of an item of a previous response. The function of this type of phrases is analogous to the projection in relational algebra, a data base model. The representation is specified as

RefExp == attriName(Attribute).

Some sample phrases and their representations are shown as follows.

(42) fàngzài nǎlǐ

(Where are they ?)

attriName(location)

(43) shì shéi xiě de

(Who write it ?)

attriName(author)

3.5 Substitution Ellipses

Any sentence of this type consists of just a noun phrase which denotes an instance of an attribute. This phenomenon is very common in most natural languages, such as English and Chinese. It is the type of ellipses which can be processed in LADDER[2]. To give an example, if an ellipsis, "ACM nē", follows the sentence "yǒu nǎxiē IEEE chūbǎn de zázhì," it means "yǒu nǎxiē ACM chūbǎn de zázhì." The representations are specified as

instanceOf(Attribute, Val).

Two typical sentences and their representations are shown as follows.

(44) Knuth

instanceOf(author, Knuth)

(45) 1986 nián 8 yuè

instanceOf(date, date(1986, 8))

4. Determination of Elliptical Sentences

We have divided the elliptical functional form into five classes. Here, we follow this classification to discuss the determination of the full meanings of elliptical sentences. Since a query sentence can be decomposed into a query phrase and a data description phrase, the determination of the full meaning also employs two processes to determine these two phrases. In this paper we mainly focus on the determination of the data description phrases, which are based on both the dialogue convention and functional representations.

4.1 Subsetting Type

Case 1.

Consider the following sequence of query sentences.

(46) Knuth suǒ xiě de shū zhōng yǒu nǎxi ē yǔ Algorithm yǒuguān ?

(Among the books written by Knuth, which are related to algrithms ?)

(47) nǎxiē yǔ DBMS yǒuguān ?

(Which are related to DBMS ?)

Sentence (46) is a basic sentence while (46) is an elliptical one. According to the dialogue convention, the elliptical sentence is interpreted by replacing the constraint phrase with that of the corresponding basic sentence because both sentences are analogous syntactically. As an example, the sentence (47) is interpreted as

(48) Knuth suǒ xiě de shū zhōng yǒu nǎxiē yǔ DBMS yǒuguā ?

(Among the books written by Knuth, which are related to compiler ?)

Formally, sentence (46) is represented as

present(restrict(restrict(cida, and([author(Knuth), bookType(book)])),

field(Algorithm), ?Attributes),

and sentence (47) is originally represented as

subset(field(DBMS), ?Attributes),

From the substructure of the representation of sentence (46), it can be seen that "field(Algorithm)" is the representation of the additional constraint phrase, "yǔ Algorithm yǒuguān". Thus we can replace it with "field(DBMS)" to construct the complete representation of sentence (46) because they are analogous. In summary, the constructed complete representation of sentence (46) is

present(restrict(restrict(cida, and([author(Knuth), bookType(book)])),
field(DBMS)), ?Attributes).

Case 2.

Consider the following sequence of query sentences.

(49) yǒuméiyǒu A.I. fāngmiàn de shūjǐ ?
(Are there books related to A.I.?)

(50) yǒu jǐ běn shì Rich xiě de ?
(How many books are written by Rich?)

In the above, sentence (49) is a basic sentence while (50) is an elliptical one. This case is different from case 1 in that no constraint in sentence (49) can be replaced with that of the elliptical sentence. Ideally, the complete data description may be constructed by concatenating the data description phrase of sentence (49) with the constraint phrase of sentence (50) such as

count(restrict(restrict(cida, and([field(A.I.), bookType(book)]))
author(Rich)), aBook)

In practice, however, the complete representation of sentence (50) is

count(restrict(dataSet(49), author(Rich)), ben).

It is because the former representation is not efficient during the period of retrieving

the data from the underlying database.

Case 3.

Now, we illustrate a more complicated case which will indicate the necessity of matching analogous patterns. Consider the following scenario:

(51) yǒu nǎxiē A.I. fāngmiàn de shūjí ?

(Which books are related to A.I.?)

(52) yǔ Expert System yǒuguān de yǒu nǎxiē ?

(Among these, which ones are related to expert system?)

(53) qízhōng nǎxiē shì 1980 nián hòu chūbǎn de ?

(Which ones are published after 1980?)

(54) nǎxiē shì 1980 nián yǐqián chūbǎn de ?

(Which ones are published before 1980?)

(55) yǔ Natural Language Processing yǒuguān de yǒu nǎxiē ?

(Which ones are related to natural language processing?)

According to the domain hierarchy, sentence (52) refers to sentence (51). By the dialogue convention, the sentence that sentences (53) and (54) refer to is (52) rather than (51). Formally speaking, the constraint of sentence (52) is represented as "field(Expert System)" while those sentences (53) and (54) are "gt(publishedDate, year(1980))" and "lt(publishedDate, year(1980))" respectively. Thus, the latter constraint is concatenated with the former one to form a new complete functional representation. As for sentence (55), it refers to sentence (51) according to the domain hierarchy. The above illustrates that the semantics and domain knowledge may affect the results of the determination.

4.2 Remaining Type

The intuitive meaning of an elliptical sentence of remaining type is analous to that of sentences containing the subpattern of "<primary data description phrase>

except <some others>". Formally speaking, an elliptical sentence of this type is determined as the representation containing a partial representation of "diff(BaseSet, Complements)", where "BaseSet" is the representation of "<primary data description phrase>" and "Complements" is that of "<some others>". Acordingly, the processing of this type of elliptical sentences is to determine the two data sets, "BaseSet" and "Complements".

Case 1.

Consider the following sequence of query sentences:

(46) Knuth suǒ xiě de shū zhōng yǒu nǎxiē yǔ Algorithm yǒuguān ?

(47) nǎxiē yǔ DBMS yǒuguān ?

(56) qíyúde nē ?

(Others)

This case is extension of Case 1 in Sec. 4.1; the structure we will discuss here is sentence (56). By the dialogue convention, sentence (56) asks in what other fields than Algorithm and DBMS the books written by Knuth are. Ideally, the data of this type include the data corresponding to the primary data description phrase, "Knuth suǒxiě de shū" except the data responsed for either the basic sentence (46) or the elliptical sentence (47). Formally, the context pattern can be recognized from the functional representations of these sentences. For example, the representations of sentences (46) and (47) are

present(restrict(restrict(cida, and([author(Knuth), bookType(book)])),
field(Algorithm)), ?Attributes)

present(restrict(restrict(cida, and([author(Knuth), bookType(book)])),
field(DBMS)), ?Attributes).

The author restricted in both the representations indicates this fact. So far, we can determine that the "DataSetDescriptor" of the functional representation of the current elliptical sentence is

DataSetDescriptor = diff(restrict(cida, and([author(Knuth), bookType(book)])),

dataSet([46, 47]).

where "restrict(cida, and([author(Knuth), bookType(book)]))" is "BaseSet" and "dataSet([46, 47])" is "Complements". As for the query phrase, we can easily decide that the queried data is the fields because there is a constraint - field(Field), where "Field" is either "Algorithm" or "DBMS", in the representation of each sentence. Thus the complete functional representation of sentence (56) is determined as

present(DataSetDescriptor, field),

where "DataSetDescriptor" is described as above.

Case 2.

Consider the query sentence following sentences (49) and (50):

(57) qǐyúde nē ?

(Others?)

The "dataSetDescriptor" of sentence (57) is determined as

DataSetDescriptor = diff(dataSet(49), dataSet(50).

where "dataSet(49)" is "BaseSet" and "dataSet(50)" is "Complements". For the query phrase, it is determined that the queried data is about the authors because the representation of sentence (50) contains the constraint phrase representation: "author(Rich)". Thus the complete representation of sentence (57) is determined as

present(DataSetDescriptor, author),

4.3 Substitution Type

An elliptical sentence of the substitution type usually contains only a short noun phrase or even just a noun. It is also a very common type in natural languages such as English or Chinese. Consider the following sequence of query sentences:

(58) yǒuǎxiē 1980 nián gòurù de shūjǐ

(Which books are bought in 1980?)

(59.a) 1981 nián nē ?

(How about 1981?)

Intuitively, the second sentence means

(59.b) yǒunǎxiē 1981 nián gòurù de shūjǐ

(Which books are bought in 1981?)

Formally, sentence (58) is represented as

present(restrict(cida, and([eq(boughtDate, year(1980)),
bookType(book)])), ?Attributes),

and (59.a) is originally represented as

instanceOf(year, 1981),

By matching "year" in the two representations, we can determine the second sentence as

present(restrict(cida, and([eq(boughtDate, year(1981)),
bookType(book)])), ?Attributes).

Next, compare the following two sequences of sentences.

(60) yǒunǎxiē Codd suǒ xiě de shū ?

(Which are the books written by Codd ?)

(61) Ullman ?

(60) yǒunǎxiē Codd suǒ xiě de shū ?

(62) qízhōng nǎxiē yǔ DBMS yǒuguān

(Among these, which are related to DBMS ?)

(63) nǎxiē yǔ Expert System yǒuguān

(Which are related to Expert System ?)

The elliptical sentence in the first sequence belongs to the subsetting type and that in the second case belongs to the subsetting type. It can be seen easily that the elliptical sentences in the first sequence can not be stated in the way as those in the second

sequence and vice versa.

4.4 Ordinal Specification Type

Ordinal ellipses are very useful as a user interface in a natural language. By using these ellipses, a user can briefly state the constraints about his interested data and then query about them closely. Consider the following sequence of query sentences:

(64) qǐng xiǎnshì yǒuguānyú Natural Language Processing fāngmiàn de shū.
(Please display the titles of books related to natural language processing.)

(65) qǐzhōng dì 5, 6 , 8 běn fáng–zài nǎlǐ ?
(Where are the fifth, sixth and eighth ones ?)

(66) qǐng xiǎnshì dì 3 běn de mùlù.
(Please show the table of contents of the third one.)

For the sentence (65), the ordinals specify the items in the data responsed for sentence (64). By the dialogue convention, sentence (66) specifies the third item in sentence (64).

Consider another sequence of query sentences shown as follows:

(67) qǐng xiǎnshì yǒuguānyú Expert System fāngmiàn de shū.
(Please display the titles of books related to Expert System.)

(68) qǐng xiǎnshì yǒuguānyú Natural Language Processing fāngmiàn de shū.
(Please display the titles of books related to Natural Language Processing.)

(69) qǐng xiǎnshì dì 3 bèn Expert System de shū.
(Please display the third one about Expert System.)

Obviously, sentence (69) specifies the third item in sentence (67) rather than sentence (68). Comparing this example with the preceding one, we find that if an ordinal specification is not followed by a constraint phrase, it is determined as referring to the item in the most recent sentence without the ordinal specification. By matching the

constraint of an elliptical sentence with that of the referred sentence, we can determine correctly the ordinal specification.

Another convention about the ordinal specification should be addressed also. Consider the following sequence of query sentences:

(70) yǒunǎxiē Natural Language Processing fāngmiàn de shū.

(How many books are related to Natural Language Processing?)

(71) dì 1 běn fàng–zài nǎlǐ ?

(Where is the first one?)

(72) qǐng xiǎnshì qí jiàqián.

(How about the price?)

(73) dì 5 běn nē ?

(How about the fifth one?)

(74) dì 7 běn nē ?

(How about the seventh one?)

The query phrases of sentences (73) and (74) are omitted. By the dialogue convention, the user must desire to know the locations and prices of the fifth and seventh books in sentence (70). That is, sentences (73) and (74) must inherit the queried information or attributes of the books mentioned in the previous sentence.

4.5 Projection Type

An elliptical sentence of the projection type may be just a predicate or a noun phrase which is the name of an attribute. It describes a new data set different from any previous data. It is used primarily to incrementally query about interested data. This type of sentences is always determined as referring to the most recently activated data set.

The following sentences show some examples and their corresponding functional representations.

(75) zuòzě shì shéi ?

(Who is the author?)

attriName(author)

(76) shì shéi xiě de ?

attriName(author)

(77) zuòzě ?

attriName(author)

Though the above sentences have different syntactic structures, they have a common function; that is, they all project an attribute of the data of a previous sentence.

It is easy to determine the full meaning of elliptical sentences of this type since they refer to the most recent data set. Consider the following sequence of query sentences:

(78) yǒunǎxiē A.I. fāngmiàn de shū ?

(What are the books related to A.I. ?)

(79) shì shéi chū bǎn de ?

(Who are the publishers ?)

(80) héshǐ chūbǎn de ?

(What are they published ?)

The constructed complete functional representation has the format of

present(dataSet(N), Attribute),

where "N" denotes the index of the referred sentence in the dialogue model and "Attribute" denotes the attribute name in the elliptical sentence.

5. Conclusions

In this paper, we have presented functional representations of query sentences. The representations describe the semantics of query sentences and reserve some messages of syntactic structures. The problem of determining the data description phrase

of an elliptical sentence based on the functional representation in Chinese dialogue has been exploited. During the process of discussion, it is demonstrated that determining the missing components in an elliptical sentence must consider both the semantic attributes and the syntactic structure. The determination of query phrases remains to be a topic for further research.

In summary, a friendly and high level user interface should conform to the following guidelines.

- It must be able to be accepted like a natural language.
- It should provide the mechanism to express rigid combination of logical connectives.
- It should provide the mechanism for stating elliptical sentences to incrementally query objects.
- The logical structure of database must be transparent and the knowledge about task domain should be built into the user interface.

For the design of a good dialogue model, some conclusions are listed below. First, the responses should be kept in the model so that the elliptical sentences can be resolved correctly and repetitive access to the underlying database system can be avoided. Second, the meaning representation must be able to describe the necessary syntactic and semantic messages. Third, the meaning representation must be powerful enough so that the dialogue convention can be easily implemented into the determination processes. Fourth, the meaning representation should be database independent and language dependent so that the implementation can be independent of the underlying database system. Finally, the dialogue model should provide a mechanism so that the determined elliptical sentences can be expressed.

References

1. C. J. Date, *An Introduction To Database Systems*, vol. I, 1985.
2. G. G. Hendrix, "Developing a natural language interface to complex data," *ACM Transactions on Database Systems*, vol. 3, pp. 105-147, 1978.
3. D. G. Bobrow, R. M. Kaplan , M. Kay , D. A. Norman , H. Thompson, and T. Winograd, "GUS, a frame-driven dialog system," *Artificial Intelligence*, vol. 8, pp. 155-173, 1977.
4. B. J. Grosz, "TEAM : an experiment in the design of transportable natural-language interfaces," *Artificial Intelligence*, vol. 32, pp. 173-243, 1987.
5. H. Ishikawa, Y. Izumida, T. Yoshino , T. Hoshiai , and A. Makinouchi, "KID designing a knowledge-based natural language interface," *IEEE EXPERT*, pp. 57-71, 1987.
6. W. A. Woods, "Semantics and quantification in natural language question answering," *Advances in Computers*, vol. 17, pp. 1-87, 1978.
7. B. W. Ballard, J. C. Lusth, and N. L. Tinkham, "LDC-1: a transportable, knowledge-based natural language processor for office environments," *ACM Transactions on Office Information Systems*, vol. 2, pp. 1-25, 1984.
8. G. Jakobson, C. Lafond, E. Nyberg, and G. Piatetsky-Shapiro, "An Intelligent Database Assistant," *IEEE Expert*, pp. 65-79, GTE Laboratory, Inc., Summer 1986.
9. B. Nash-Webber and R. Reiter, "Anaphora and logical form:on formal meaning representations for natural language," *IJCAI*, vol. 1, pp. 121-131, 1977.
10. Horrigan, "Modelling Simple Dialogs," *IJCAI*, vol. vol. 1, p. 88, 1977.
11. I. Spiegler, "Modelling man-machine interface in a data base environment," *International Journal of Man-Machine Studies*, vol. 18, pp. 55-70, 1983.

Appendix Logical Specification of the Dialogue Model

```
DialogProcess == [Exches]

Exches == Exch | Exch, Exches

Exch == [Query, Response]

Query == basic(Complete) | elliptical (Ellipsis, Complete)

Complete == present(DataSetDescriptor, Attributes) |
      count(DataSetDescriptor, Unit) | exist(DataSetDescriptor) |
      lentBy(DataSetDescriptor, Borrower) | lendable(DataSetDescriptor) |
      lent(DataSetDescriptor)

DataSetDescriptor == cida | DataSetPointer | RefExp
      some(DataSetDescriptor, Unit, Number) |
      restrict(DataSetDescriptor, Constraint) |
      diff(BaseSet, Complements) | ordinal(DataSetPointer, Orders)

DataSetPointer == dataSet(Number)

RefExp ==
        complement(Constraint, Attributes) |
        ordinal(Constraint, Orders, Attributes, Unit) |
        subset(Constraint, Attributes) |
        instanceOf(Attribute, Val) | attriName(Attribute)

Unit == ben | aJournal | kind | year

Constraint == '?Constraint' | not(Constraint) | and(Constraints) |
          or(Constraints) | RelOp(Attribute, Val)

Constraints == [Constraints0]

Constraints0 == Constraint | Constraint, Constraints0

RelOp == eq | ne | ge | gt | le | lt

Attributes == [Attributes0]

Attributes0 == '?Attributes' | Attribute | Attribute, Attributes0
```

```
Attribute == bookType | author | cost | contentTable | donator |
        field | publisher | publishedDate | location | id | borrower
        bookName | journalName | beginDate | endDate
        donatedDate | vol | journalNo | lentDate | date
Val == '?Val' | Number | String | Date | Vol | JournalNo | Year | Month
Date == date(Year, Month)
Year == Number
Month == Number
JournalNo == journalNo(Year, AjournalNo)
Val == Number
AjournalNo == Number
BaseSet == DataSetDescriptor
Subsets == dataSet([Numbers])
Numbers == Number | Number, Numbers
Complements == dataSet([Numbers])
Orders == Order | Order, Orders
Order == OneOrder | OrderRange
OneOrder == Number
OrderRange == (StartNumber, EndNumber)
StartNumber == Number
EndNumber == Number
Borrower == String
ElliQuery == Complete
Ellipses == Ellipsis | Ellipsis, Ellipses
Ellipsis == RefExp | ElliQuery
Attributes == Attribute | Attribute, Attributes
Response == [Informations]
```

Informations == Information | Information, Informations

Information == (DataDescriptor, DataList)

DataDescriptor == [Attributes]

DataList == [ValLists]

ValLists == ValList | ValList, ValLists

ValList == [Vals]

Vals == Val | Val, Vals

Acknowledgement

This research work was supported by ERSO, ITRI, Hsinchu, Taiwan, under contract, MIST-E76006(1987) and by Taiwan International Standard Electronics Ltd..

The Parsing Environment for Mandarin Syntax

I-Peng Lin, Shuan-Fan Huang, Hsin-Hsi Chen & Ka-Wai Chui

林一鵬　黃宣範　陳信希 徐嘉慧

National Taiwan University

國立台灣大學

Proceedings of ROCLING I(1988)
R.O.C. Computational Linguistics Workshops I pp 211-214

中華民國第一屆計算語言學研討會論文集 211-214 頁

The Parsing Environment for Mandarin Syntax

I-Peng Lin, Shuan-fan Huang, Hsin-Hsi Chen and Ka-Wai Chui

Department of Computer Science and Information Engineering

abstract

'Syntax' is the sentence-formation component in grammar, specifying how sentences are constructed out of phrases, and phrases out of words. It is necessary to set up a parsing environment to implement the Mandarin syntax in computer. This paper thus attempts to introduce this parsing enviroment .

There are three types of sentences in total: declarative sentences, imperative sentences and questions. The declaratives can further be divided into simple, complex and compound constructions. Simple sentences are defined as those consisting of a single predicate. When there are more than one predicate within a sentence, it is called complex sentence. We classify 3 types of complex constructions. The first of which results from subcategorization. The second and the third types are the verb-reduplicated and the serial-verb constructions. Compound sentences presuppose the presence of conjunctions to link up two or more phrases or clauses. As imperative constructions are concerned, they have their basis on their declarative counterparts, yet with a number of restrictions, e.g. the optional subject has to be the second person, the construction has to be unmarked aspectually. Finally, the different types of questions in Mandarin include question-word questions, yes/no questions, choice questions, A-not-A questions, tag-questions, embedded questions. The linguistic device to generate all of them is the context-free X-bar Phrase Structure Rules.

The parsing environment for Mandarin syntax is indeed based on the data analyses of the various types of sentences discussed above. The Prolog-based bottom-up parser can even tackle the problems of movement transformation by means of three principles in the Government-Binding theory, namely the Empty Category Principle, the C-Command Principle and the Subjacency Principle. A sequence of translation rules is given to add these linguistic principles to the general grammar rules, the leftward movement grammar rules, and the rightward movement grammar rules, respectively. The empty constituent problem is solved to allow the trace to be the first element in the grammar rule body. A special data structure for extraposition list is proposed to transfer the movement information from the bottom to the top. Based upon this structure, the fastest merge algorithm is designed. Those

unnecessary *merge* predicates can be eliminated with the help of the transitive relation. Thus, the new design not only extends the original bottom-up parsing system with the movement facility, but it also preserves the parsing efficiency.

Criteria for the Classification of Lexical Categories in a Syntax-Oriented Parsing System

Yu-Ling Una Shiu* and Keh-Yih Su**

* **Institute of Linguistics**
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.

** **Department of Electrical Engineering**
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.

ABSTRACT

In Natural Language Processing systems, different classification of lexical categories will lead to different set of rules, and thus different kinds of analyses, therefore the choice of a good category system is very important to the efficiency and to the memory load of the overall parsing system. Unfortunately, although every parsing system has a set of lexical categories, the issues as to whether these category systems are properly chosen, and the factors for evaluating the adequacy of the classification of lexical categories has generally been ignored. Especially, things go worse in research areas, such as Mandarin NLP field, where many fundamental issues are just beginning to be explored, the lack of a good category system apts to obstruct an in-depth research.

In this paper, we propose eight criteria for the classification of lexical categories in a syntax-oriented parsing system. These criteria are **syntax dominance, descriptive power, simplicity, explicitness, mutual exclusion, collective exhaustiveness, applicational efficiency**, and **conventionality**. Each of them is clearly defined and illustrated by Mandarin examples. Furthermore, the tradeoffs among these criteria are also taken into consideration. These criteria and the discussions of tradeoffs will be helpful in serving as a guide for designing and evaluating a category system.

I. Introduction

In Natural Language Processing (NLP) systems, the classification of lexical categories is the fundamental work. NLP first takes a lexical analysis which reads and converts the input into a stream of tokens, which are then analyzed by the parser. So the choice of the inventory of tokens and the assignment of tokens to words, i.e., the classification of lexical categories, are the very jobs of lexical analysis. Since the quality of such classification will usually affect the performance of its following stages (such as syntactic analysis, semantic analysis), a good category system is indispensible to NLP.

Although there are some existing Mandarin lexical category systems, most of them are not proposed for computer parsing (such as [Chao 68], [Lyu 80]). Their adequacy in NLP is quite doubtful. At the mean time, though a variety of parsing algorithms have been proposed, the syntax-oriented parsing algorithm is still a very popular one within the NLP community ([Su 87], [Char 1986]).[1] Because syntax-orientded parsing systems and non-syntax-oriented ones may have different requirements on their lexical categories, even if some category systems are constructed mainly for parsing (such as [CKIP 86, 88]), further examination is still needed to ensure their suitability for syntax-oriented parsing systems. However, it is quite surprising that there are still no objective, explicit, and rigorous criteria available in literatures to judge the quality of a category system. The lack of a set of clearly-defined criteria makes the comparison and examination of different lexical category systems a very hard task, if possible at all.

In response to such demand, we propose eight criteria in this paper to evaluate different category systems within the syntax-oriented parsing framework. Each criterion is clearly defined and illustrated by Mandarin examples. Furthermore, tradeoffs among these criteria are also discussed. These criteria and the discussions of tradeoffs can serve as a helpful guide for analysis and evaluation of a category system.

II. Criteria for the Classification of Lexical Categories

In order to facilitate the analysis and comparison of different category systems within the syntax-oriented parsing framework, eight criteria are set up as follows:

1. Syntax Dominance : *considering only the phenomena of syntactic distribution*

In a syntax-oriented parsing system, words should be classified only according to their syntactic distribution. To make this criterion more clear, we can regard each lexical entry as a set of attribute-value pairs [Gazd 85]. Each pair encodes a piece of linguistically significant information, such as its character(s), its phonological representation, its possible position(s) in the sentence, its semantic meaning(s), its pragmatic function(s), etc.. Then, this criteria in fact says that, the only attributes we should take into consideration at the stage of lexical category classification are those capable of informing us what positions in the sentence a given entry can occupy. The reason is that if we intend to encode every pieces of information by the classification of lexical categories, the resultant consequence will be that we have to assign every word an independent lexical category. Such classification is certainly meaningless and useless. Thus, our consideration must be selective. While, in a syntax-oriented parsing

system, the main purpose of its analysis is to render correct syntactic structures for input sentences. Naturally, under this approach, any non-syntactic information is irrelevant.

Take Mandarin sentential particles as an example.[2] This set of items, such as 'incoative' *le*, 'presuppositional' *de*, 'friendly reminding' *o*, etc., always occur in the sentence-final position. If we recognize Mandarin sentential particles as an independent lexical category, say PART, their characteristic syntactic distribution can be nicely captured by a rule, as shown in (1):

(1) S′ —> S (PART)

However, among these particles, three of them carry interrogative information, namely, 'Yes-No question' *ma*, 'expecting addressee's participation' *ne*, and 'conjectual' *ba*. Although these interrogative particles obviously have a special semantic meaning (or pragmatic function) which can turn declaratives into questions, they do not differ with other sentential particles in their syntactic distribution. Thus, according to the criterion of **Syntax Dominance**, interrogative particles should not form a separate lexical category.

However, one point worth noting here. Systems using a syntax-oriented parsing algorithm are not unable to manage semantic information at all. But they always handle semantic messages by checking semantic attributes in lexical entries rather than by the classification of lexical categories.

2. Descriptive Power : *adequate descriptions of linguistic phenomena provided by a category system*

Generally speaking, a category system which can adequately account for more syntactic phenomena is better than one which can do less. Consider Mandarin sentential particles again. Two of their detailed distributional phenomena are observed. First, in a simple sentence, two sentential particles may co-occur in succession, with *le* and *de* alternating in penultimate position and other sentential particles absolute sentence-final position. Second, in a complex sentence, only *le* and *de* can be attached to an embedded clause, others must have a scope over the whole matrix sentence ([Shiu 88]). Following the above observation, category systems which contain only one sentential particle category like (2) will fail to capture these empirical facts. The best solution they could offer is shown in (3):

(2) PART : *le, de, ma, ne, ba, o,* etc.

(3) a. S″ —> S ′ (PART)

b. S′ —> S (PART)

c. VP —> V (S′)

However, this analysis allows too many ungrammatical sentences, such as below:

*(4) Da-shiung shiantzai bu du-buo **ba le** ?
Da-shiung now not gamble BA LE

*(5) Yi-jing yi-ding huei sheng-chi **o de** !
Yi-jing surely will get angry O DE!

A more powerful Mandarin category system should have two separate categories for sentential particles, as shown below:

(6) a. PART′ : *le, de*

b. PART″ : *ma, ne, ba, o,* etc.

With the two separate lexical categories in (6), we can came up with a more adequate analysis, presented in (7), which can correctly rule out ungrammatical sentences , like (4), (5), and sanction grammatical sentences, such as (8), and (9).

(7) a. S″ —> S′ (PART2)

b. S′ —> S (PART1)

c. VP —> V (S′)

(8) Da-shiung shiantzai bu du-buo **le ba** ?
Da-shiung now not gamble LE BA
' (I suppose) Now, Da-shiung won't go gambling (any more), will he ? '

(9) Yi-jing yi-ding huei sheng-chi **de o** !
Yi-jing surely will get angry DE O !
' (Let me tell you) Surely, Yi-jing will get angry ! '

However, we have to mention that the criterion of **Descriptive Power** must balance with the next criterion **Simplicity**. Their tradeoffs will be discussed in Section III.

3. Simplicity *: using as few categories as possible; avoiding any redundant classification*

This criterion is supported by both computational and linguistic considerations. Computationally, the set of lexical categories is relevant to the set of grammar rules. Usually, an increase in the number of lexical categories implies the simultaneous need of a larger set of grammar rules. In a syntax-oriented parsing system, the parsing table is constructed by expanding the grammar rules, therefore, the growth of rules will certainly lead to the enlargement of the parsing table, and thus the increase of memory load. The number of lexical categories is therefore preferred to be as few as possible. Linguistically, simplicity usually correlates with maximal degree of generalization. Any redundancy will certainly destroy the elegance of an approach. Thus, the most economical solution is usually the best choice.

For illustration, consider Mandarin common nouns and proper names. Many category systems make distinction between them for the reason that only common nouns can be preceded by determiner-measure compounds (D-M compounds). However, this observation is not entirely right. Consider the following counterexamples:

(10) **Na wei Li shiaujie** tzou guo lai le.
That M Li Miss walk along LE
' That woman called Miss Li are walking along. '

(11) Women ban shang you **liang ge Wang-shiau-wu**.
We class LOC have two M Wang-shiau-wu

' There are two persons named Wang-shiau-wu in our class. '

Sentence (10) and (11) show that there is no problem for proper names to co-occur with D-M compounds. Removing this distinction, proper names and common nouns in fact have the same possible syntactic positions. Thus, they should be combined into one single category according to the criterion of **simplicity**.

4. Explicitness : *defining precise scope for each category by using rigorous definitions*

The definition of each lexical category must be rigorous enough to yield desirable classification. Since NLP system is usually a teamwork, the use of vague or ambiguous definitions for lexical categories will result in a lot of mis-labeling which will destroy the consistency of the overall category system.

For example, if nouns are vaguelly defined as 'names of entities', it will be hard to judge whether 'linguistics'*yu yian shiue* or 'dragon'*lung* should be considered as nouns or not, because *yu yian shiue* is not a concrete object, and *lung* does not exist at all. But they indeed share similar syntactic distribution with other nouns. Thus, a more explicit classification of lexical categories should define Mandarin nouns as 'what can be modified by D-M compounds or possessive expressions'. Such definition can then correctly assign the category noun to *lung* and *yu yian shiue* without confusions.

5. Mutual Exclusion : *complementary classification; avoiding overlapping*

If the domains of different categories overlap within a category system, it implies that words in the overlapping areas will be assigned with more than one category labels. Such multi-categoried words are the main source of lexical ambiguities. Except for some homographs which will be inevitable in raising such parsing difficulty, a good lexical category system should avoid this type of ambiguity with mutual exclusive classification.

For instance, some Mandarin category systems follow the traditional classification of English and recognize verbs and adjectives as two separate categories. But, in fact the so-called adjectives can be further divided into two groups (Figure 1), namely, predicative adjectives and non-predicative adjectives. Predicative adjectives, like *piauliang, gau, gaushing,* etc., are very similar to verbs in their syntactic behaviors in that they can form A-NOT-A constructions, be modified by adverbs, act as the main predicates of sentences, etc. (Figure 2). This is exemplified as follows:

(12) a. Yi-jing **lai-bu-lai** ?
Yi-jing come-not-come
' Does Yi-jing come or not come ? '

b. Yi-jing **piauliang-bu-piauliang** ?
Yi-jing pretty-not-pretty
' Is Yi-jing pretty or not pretty ? '

(13) a. Da-shiung **hen shihuan** Yi-jing.
Da-shiung very like Yi-jing
' Da-shiung likes Yi-jing very much. '

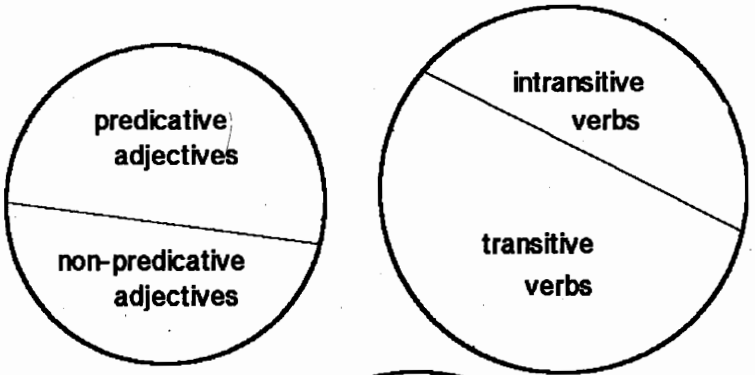


Figure 1

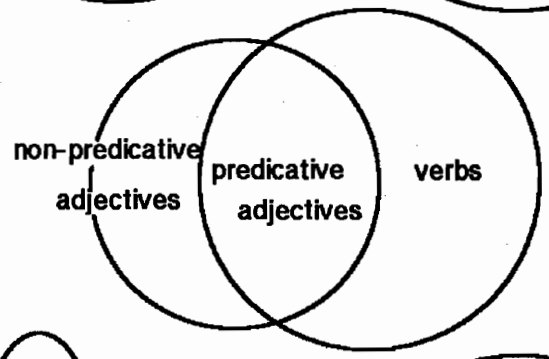


Figure 2

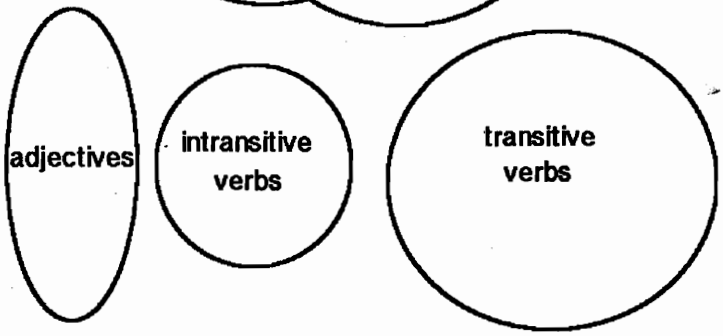


Figure 3

b. Da-shiung **hen gau** .
Da-shiung very tall
' Da-shiung is very tall. '

(14) a. Da-shiung bu **shuohua**
Da-shiung not talk
' Da-shiung does not talk. '

b. Da-shiung bu **gaushing** .
Da-shiung not happy
' Da-shiung is not happy. '

If the above syntactic positions are specified both in the definitions of verbs and adjectives, all predicative adjectives will unfortunately bear two category lables. Such analysis greatly increases the load of a syntax-oriented parsing system.

A better solution is to classify non-predicative adjectives, like 'the most important' *shouyau*, 'chronic' *manshing*, etc. as a single category. Further, since non-predicative

adjectives are usually intransitive, they can combine with intransitive verbs to form another lexical category. Thus, according to the criterion of **Mutural Exclusion**, the plausible classification of verbs and adjectives may be like as shown in Figure 3.

Careful readers may notice that this criterion may also contradict with the criterion of **Simplicity**. Their tradeoffs will also be discussed later.

6. Collective Exhaustiveness *: thorough classification; the union of every classes should be equal to the whole set of data*

This criterion requires that each word must be classified into at least one class. Words without category labels will fail to be accessed by the parser. Thus, even a small set of words, like Mandarin particles, which do not occur in English at all, should not be neglected in a Mandarin category system.

7. Applicational Efficiency *: yielding the most desirable and economical processing with regard to its special application domain*

A sophisticated category system should also take its application domain into consideration. Different application fields may have different requirements for their category systems. For example, in a Machine Translation System (MTS), the categories of a source language and those of a target language may had better carry some kind of correspondance. The MTS with this kind of lexical category classification can deal with its transfer rules better. Under this consideration, a Chinese-to-English MTS may prefer Mandarin predicative adjectives, separated from intransitive verbs, also form an independent category. Since adjectives and verbs really distinct in English, such separation in Mandarin will make the interlanguage corresponse easier to be captured and is helpful in render correct translations.

8. Conventionality *: following established conventions*

If the above criteria are equally satisfied, a category system with more conventional notations, and more standard definitions will be more transparent to the linguists and have better mnemonic quality. For instance, the notations of lexical categories such as noun, verb, adjective ... etc. have been widely-adopted. And most people in linguistics or NLP community have a general idea about their scope of classification. Thus, a catgory system following this way of classification will be conceptually easier and make more ready-made analyses available. This criteria is therefore proposed as the last point.

III. Tradeoffs among the Criteria

So far, we have explained and illustrated the proposed criteria. Ideally, a good category system should closely obey all of them. However, the complicated linguistic phenomena of natural languages and the inherent limitations of computational devices preclude them from coexisting optimally within most category system. Thus, we will briefly discuss the main tradeoffs among them. The considerations are separately represented as follows:

1. Descriptive Power & Simplicity

With the same number of lexical categories, a category system having more descriptive power is superior; with the same descriptive power, a category system having a smaller set of lexical categories is more desirable. But if there are contradictions between these two criteria, the tradeoffs between them should be carefully considered.

For example, there are dependences between Mandarin measures and nouns. Different classes of nouns require different classes of measures, such as plants can be preceded by *ke, ju, tsung*, etc., animals by *jr, chiun, wo,* etc., and furnitures by *tau, tsu, jian,* etc.. If we classify nouns and measures into various categories according to their co-occurrence, the descriptive power of the category system will be increased. But such classification is trivial and will create numerous additional lexical categories which rersults in a plenty of additional rules. For the sake of memory load, this analysis is not welcome. The preferred alternative is to encode such co-occurrence restrictions by using attributes. By virtues of checking attributes, we can still correctly constrain the co-occurrence between measures and nouns, and reach a desirable consequence without using a clumsy set of grammar rules.

Thus, based on the consideration of descriptive power and simplicity, we suggest that only general syntactic phenomena should be handled by the classification of lexical categories.

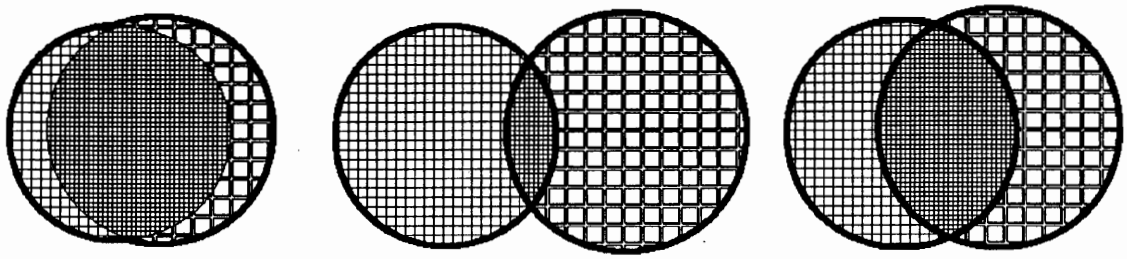
2. Simplicity & Mutual Exclusion

If two classes of words share some possible syntactic positions, we suggest the following solutions :

A If the syntactic distribution of two classes overlaps a lot, these two classes should be combined into one lexical category, and using condition check in grammar rules to specify their distinction (Figure 4. a). For example, some pronouns really cannot be preceded by D-M compounds, such as '(politely) your father' *lin-tzuen*, '(modestly) my son' *shiau-chiuan.* But their other position are just similar to those of nouns. Thus, more efficiently, we should regard them as a single lexical category, and using condition check to prevent pronouns co-occurring with measures.

B If the distribution of two classes of words overlaps just in a few cases, they should be classified into two separate lexical categories, and with the few words appeared in the overlapping positions labeled two category lables (Figure 4. b).

C If the overlapping section of the distribution of two classes of words is approximately equal in size to the two distinct non-intersecting sections, then the most efficient way is to classify all three of them into independent categories (Figure 4. c). This is the method suggesting for handling traditional adjectives and verbs.



a. b. c.

Figure 4 Overlapping between the distribution of two classes of words

3. Simplicity & Applicational Efficiency

If some important applicational advantages can be captured by further classification of lexical categories, the criterion of **Simplicity** has to give way to such special demand. For example, as we have mentioned above, though the adjective category in Mandarin is not theoretically needed and will add the number of lexical categories, however, in a Chinese-to-English MTS, this category is still preferred for the efficiency of overall processing.

4. Descriptive Power & Applicational Efficiency

Generally speaking, a category system with more descriptive power is favored. But if concerning applicational efficiency, some reservation about descriptive power may be made. Some classification of lexical categories may really add to the descriptive power, however, if such information is of little use or even irrelevant to its applicational domain, such classification is just useless and not worthy of implementation.

IV. Conclusion

Before a satisfactory category system can be determined, a good set of criteria for evaluating them should be set up first. This paper proposes eight criteria for the classification of lexical categories in a syntax-oriented parsing system. Each of them is examplified by Mandarin data. Since there are no standard lexical category system in Mandarin NLP as yet, this set of criteria can helpfully serve as a guide for designing a good category system and as a reference for examining existing category systems. Further, some tradeoffs among these criteria are indicated. Since there are still too many controversies in Mandarin syntax, it is beyond our ability to provide a detailed quantification of these tradeoffs. Nevertheless, the general direction is pointed out.

NOTES

1 The distinction between syntax and semantics is not a clear-cut matter, thus we use the term **Syntax-oriented**.

2 The Romanization system adopted in this paper is Mandarin Phonetic Symbols II (MPS II), which is formally announced by the Ministry of Education in 1986.

ACKNOWLEDGEMENTS

We are indebted to Prof. Keh-Jiann Chen for making valuable advices on an earlier draft of this paper, to Prof. Ting-Chi Tang for helpful comments on Mandarin examples, and to Prof. Chu-Ren Huang and Prof. Samuel Wang for useful suggestions. We are also grateful to all the members in BTC R&D Center and CKIP, for their discussions and goodwill. Special thanks are due to Behavior Tech. Computer Corp. (BTC), for her full financial support. Responsibility of errors is, of course, ours.

REFERENCES

[Chao 68] Chao, Yuen-Ren. A Grammar of Spoken Chinese. Berkeley : University of California Press (1968).

[Char 86] Charniak, Eugene & Drew Mcdermott. Introduction to Artifical Intelligence. Addison-Wesley Publishing Company, Inc. (1986).

[CKIP 86] 中文詞知識庫小組, 國語的詞類分析, 技術報告T002, 中研院計算中心：台北南港 (1986).

[CKIP 87] Chang, Li-li, Huang, juei-chu, Chang, Li-ping, Wei, Wen-chen, Cheng, Ya-hsia, Chen, Keh-jiann, Tseng, Shih-shyeng, Hsieh, Ching-chun. "Classification and Co-occurrence Restrictions in Chinese Simple Noun Phrases". The Chinese Language Society. (1987)

[CKIP 88] 中文詞知識庫小組, 國語的詞類分析（修訂版）, 中研院計算中心：台北南港 (1988).

[Gazd 85] Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, an& Ivan Sag. Generalized Phrase Structure Grammar. Oxford : Blackwell. (1985)

[Lyu 80] 呂叔湘, 現代漢語八百詞, 商務印書館香港分館 (1980).

[Lyu 81] 呂叔湘, 試論非謂形容詞, 中國語文, 1981, no. 2, (1981).

[Nire 87] Nireburg, Sergei. ed. Machine Translation : Theoretical and Methodological Issues. Cambridge : Cambridge University Press. (1987)

[Shiu 88] Shiu, Yu-Ling & Chu-Ren Huang. "Unification-based Analysis and Parsing Strategy of Mandarin Particle Questions". To appear in 1988 Proc. of International Computer Symposium, Taipei, Taiwan, Dec. 15–17, 1988. (1988)

[Su 87] Su, Keh-Yih, Jing-Shin Chang, & Hsue-Hueh Hsu. "A Powerful Language Processing System for English-Chinese Machine Translation." 1987 Proc. of Int. Conf. on Chinese and Oriental Language Computing, pp. 260–264, Chicago, June 15–17, 1987. (1987)

[Tang 77] 湯廷池, 動詞與形容詞之間, 華文世界, 1977, 9 : pp. 30-40, also in 國語語法研究論集,

1979, pp. 161-167. 台北：學生書局. (1977)

[Wilk 75] Wilks, Yorick. "An Intelligent Analyzer and Understander of English", Communications of the ACM, Vol 18, no 5, pp. 264–274, May 1975. (1975)

Descriptive Language as a Linguistic Tool

Mei-Hui Su* and Keh-Yih Su**

***BTC R&D Center**
2F, 28 R&D Road II
Science-Based Industrial Park
Hsinchu, Taiwan, R.O.C.

****Department of Electrical Engineering**
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.

ABSTRACT

In developing a natural languages processing system, the linguist needs to transfer a large amount of linguistic knowledge into the system. Therefore, the transferring and the maintaining of the linguistic knowledge in a working system become a crucial issue. If the linguist transfers the linguistic knowledge indirectly through the computer engineer, the productivity of both the linguist and the computer engineer might be lowered. So it is better for the linguist to transfer the knowledge directly into the system.

In this paper, we propose a descriptive language that is tailored specifically for direct transferring and maintaining of the linguistic knowledge needed in our system by our linguists. We will also discuss the criterions for designing the descriptive language and the design experiences gained from a completed descriptive language and its environment. In addition to offering a simple and direct way to express linguistic knowledge, descriptive language unifies the knowledge into a clear and definite form. As a direct consequence, it is easier to build an environment that can maintain the global knowledge consistency needed in a natural language processing system.

Introduction

In a natural language processing (NLP) system, after a research topic is completed by the linguist, the knowledge obtained must be transferred into the working system in order to be utilized during the natural language processing. The traditional approach of transferring the knowledge to the system indirectly is depicted in the following figure.

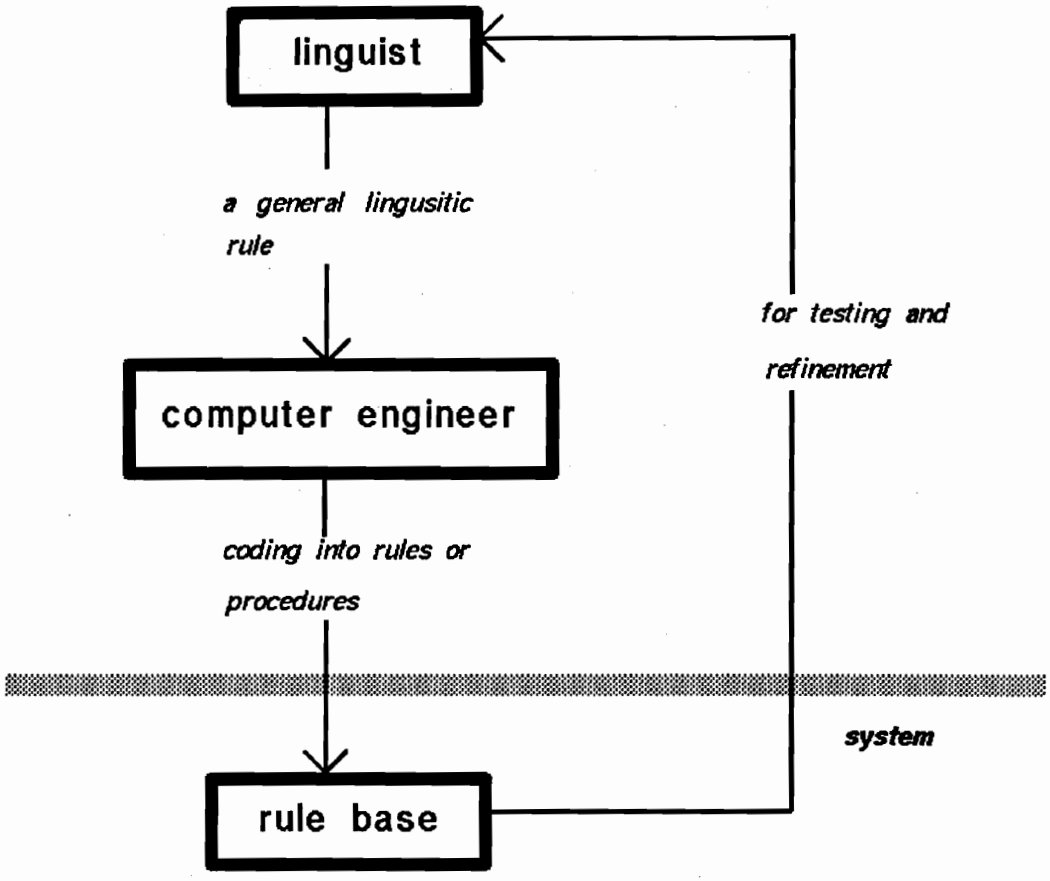


Figure 1. Indirect knowledge transferring approach

In this figure, the linguist passes the rule to the computer engineer after a research topic is completed. Then, after the rules are implemented by the computer engineer, the linguist will test the system to see if it works as expected. Several places can go wrong in this process. In the first place, the linguist might have concluded an incorrect knowledge and so the system will not work as expected. In the second place, the computer engineer might have misunderstood the request and implement the wrong rules. As a result, the linguist will get an incorrect response from the system. And in the third place, the computer engineer might implement the request erroneously and so the linguist will get a wrong feedback from the system.

This indirect approach introduces two extra error sources in comparison to that of a direct approach in which the linguist would interact directly with the system. Above all, the more

severe problem of using the indirect approach is that if an error occurred, it is hard to pinpoint the source of error. Therefore numerous looping through the flow is likely before the initial request is correctly implemented. As a result, the productivity of both computer engineer and linguist is lowered. Hence, it is essential to have the linguist interact directly with the system instead of through the computer engineer. This can be solved by providing a tool that is designed specifically for handling linguistic knowledge and so that the linguist can interact with the system directly.

This linguistic tool might be an user-friendly support environment or it might go as far as a complete high-level linguistics-oriented programming language like LINGOL [PRAT 73], MUIR [WINO 86] and PERIPHRASE [BEES 88]. For our machine translation system, we developed a descriptive language which is a high level declarative language for the linguist to develop and to maintain the linguistic modules we have, to minimize the computer engineer's involvement during the knowledge transferring process and to provide a friendly knowledge transferring environment for the linguist. The purpose of developing this descriptive language is not to create a metalanguage [SHIE 86] which is intended to handle different linguistic grammar formalisms but as a tool that meets the specific needs of our linguists and their knowledge.

In addition to the reasons mentioned above, descriptive language has another advantage. The advantage is that it allows the linguistic knowledge to be represented and maintained in a clear and definite way. This is very important in a natural language processing system with an on-going research in linguistic phenomena. The reason is that to maintain a consistent view of all linguistic knowledge for the NLP system is not a simple task. But a complete and consistent view is necessary if the output quality of the working system is to be at its best. Therefore, It will be of advantage if the linguistic knowledge are kept in a definite form. In this way the descriptive language will make it easier to build the environment for the computer to support the knowledge consistency needed by the working system.

Criterions of Descriptive Language

In designing a descriptive language it is important to keep the following three criterions in mind. The first criterion is that its interface must be user-friendly and the format should be declarative. The rule format should be declarative because declarative rule is easier to write and it screens the knowledge representation from the issues of the underlying system. This is important because the user will be the linguist instead of the computer engineer. Therefore the user should not be hindered with any detail of the system. The second criterion is that it should be linguistic felicitous [SHIE 85][SHIE 86]. This implies that the descriptive language should be flexible and powerful enough to let the linguist represent their knowledge and it should also be natural in the linguistic sense. This will make the maintaining of the knowledge easier for the linguist. The third criterion is that the linguistic information stored internally should be in a form such that a runtime efficient driving mechanism is possible to implement. This is a must if we do not want to slow down the speed of the natural language processing system.

Format of a Descriptive Language

In our system, we have a linguistic module called the condition and action. The linguistic rules in this module place condition tests on grammar rules being applied and adopt some

actions if the conditions are satisfied. It is similar in idea to the condition and action used in the ATN [WINO 83]. Following is a subset of the syntax of the descriptive language defined for the condition and action rules.

test2 : ASSIGN what TO where1
ftype : TERMINALWORD (where1 . < aRVARIABLE >) . wpropty
where1 : PARENT | CHILD | aCVARIABLE

In the above, the lower case terms are the non-terminals and the upper case terms are the terminals or the reserved words in our grammar. This syntax is a context-free grammar with a total of 57 grammar rules. Following are some of the condition and action rules in our system written according to the descriptive language defined above.

1: *[assign (parent.<R_Head>).<A_Aspt> to parent]*

This is an example of attribute percolation from a child to its parent. This rule assigns the aspect attribute of the child, which is the head in the current node's role register [WINO 83], to the current node.

2: *[terminalword(parent.<R_Head>).stem is ("more")]*

This is an example of a condition test that determines whether the terminal word governed by the head attribute of the current node is the word *"more"*. If the condition test fails, the current grammar rule being applied will be blocked and the next grammar rule will be tried.

3: *if [terminalword(parent.<R_Head>).Morf is ("er")]*
then [assign ("cprt") to parent]

This is an example of doing a conditional check before a direct assignment of an attribute to the current node.

Although each of the three rules mentioned above has different functions, their formats are similar. The rule format for this descriptive language was designed after several meeting with our linguist to make sure that it will give the expressive power they need and that it will be easy for them to understand and write the rules.

Design Experience

There are several linguistic modules that can be implemented using the descriptive language. They are the Condition and Action module, the Transfer Rule module, the Grammar Rule module, and the Knowledge module. Every module's knowledge is distinctly represented because their functions during the translation process are different. For example, the transfer rules are used in the transferring phase and the condition and action rules are used in the analyzing phase. Therefore the descriptive language will have different format for each module if we want to retain the properties that are best suited for their functions. As a result, each module will become an individual linguistic kernel within the descriptive language support system. A block diagram of this support system is shown in Figure 2.

In Figure 2 , each linguistic kernel in this support system will maintain their own independent rule base as their private data. If the linguists want to make changes to a kernel's rule base, they will have to interact directly with that kernel. At the present, one of

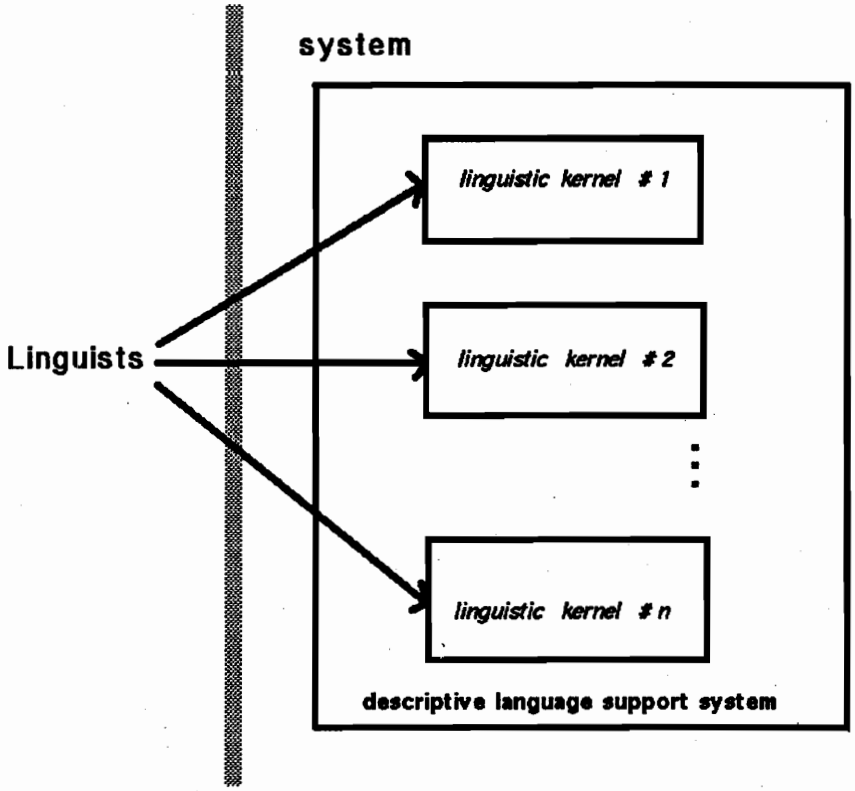


Figure 2. Descriptive language support system

the linguistic kernels is completed in our system and it is the condition and action kernel. A simple block diagram which shows how this kernel is implemented is in Figure 3.

The condition and action kernel is separated into two main distinct parts according to their basic intended purposes. The first is the user interface and rule base maintainer. The second is the code generator. The user interface will process the linguist's request to insert, update or delete rules. It will also make a quick syntactic check on the rule format and prompt the user if the rule is not in the correct format. This part also does the housekeeping chores like updating database's general information. The task of code generator is to take legal rules and generate procedures in C for each of the rules.

The condition and action rules written by the linguist through the descriptive language are very rigid in construction and rarely altered once they are written. Therefore developing a kernel for the Condition and Action module can be seen similar to that of developing a compiler which in this case will accept the linguistic knowledge rules as its input and produce codes recognizable by the system as its output.

Looking at the block diagram of Figure 3, if the user interface and the code generator blocks are combined, it is similar to the conventional compiler that takes in condition and action rules and produces the corresponding routines in C language. But there are several features in this kernel's approach that makes it different from the traditional compiler. The first feature is that the rules can be changed incrementally and are kept in a private rule base. The second feature is that this kernel has an on-line interactive user interface (it also has a batch-mode). The third feature is that this kernel will give immediate feedbacks to user on ill-formed rules. All three features mentioned above are intended to make this kernel user friendly and easy to use. And the last feature is that the object code is in C instead of in any lower level language. The reason for this is because the good readability of C procedures

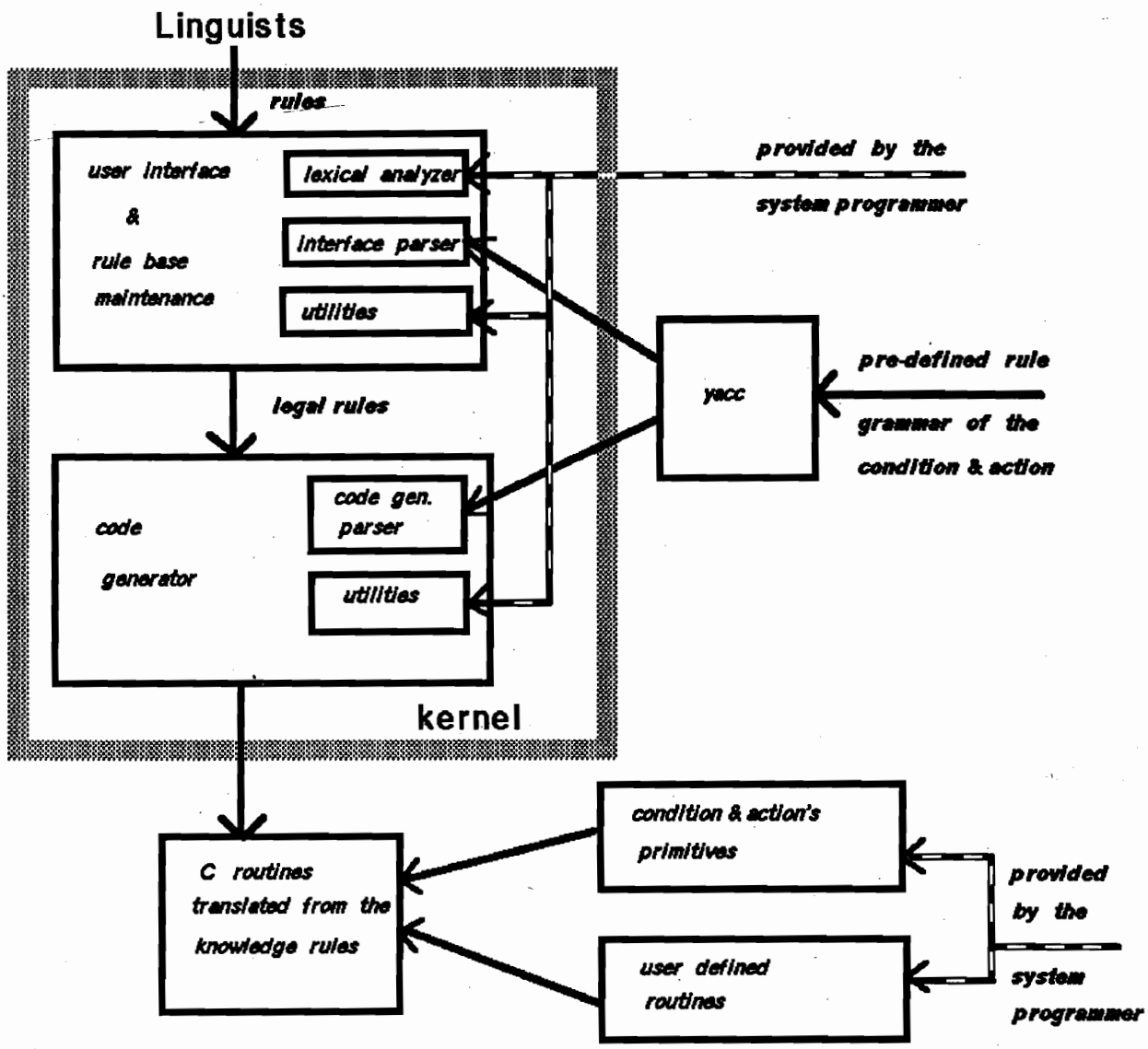


Figure 3. The condition and action kernel

which will be of advantage during debugging. The one extra step of compiling the C program will not matter much since it will not affect the working system at the runtime.

In this kernel, YACC is used in creating parsers for the user interface block and for the code generator block. The reason for having two parsers is because their functions are different and so their syntax rule grammars for the YACC are different. The parser for the user interface is for doing the syntax check and the parser for the code generator is for driving the code generation. With the rigid form of the condition and action rules, the parser produced by YACC is more than adequate. With the procedures translated automatically from the linguistic rules, this compiler solution frees the computer engineers from hand-coding each new procedure as rules are added or updated. Figure 4 shows how the condition and action module's knowledge is incorporated into the working system. In the figure, the C routines are first compiled into machine executable codes and then included into the working system.

Several design decisions were made in designing this kernel. First, at the user's level the condition and action rule remains in their declarative rule form for the linguistic friendliness.

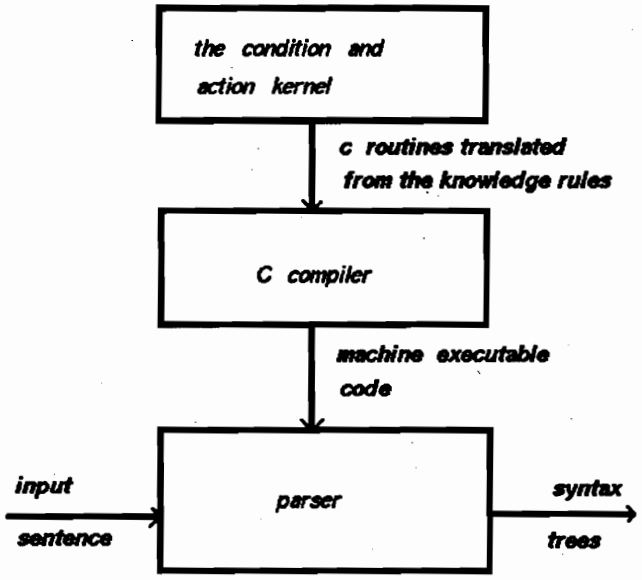


Figure 4. The incoporating of linguistic knowledge into the parser

But at the system's level, rules are translated into procedures to improve the run-time efficiency of the system. This is possible because of the stable nature of the condition and action rule. Rules of other kernels might not exhibit such property and so different design choice must be made. Another design decision is in allowing the inclusion of user-defined routines in contrast to expanding the rule grammar format to include exceptional and rare cases.

An actual session of this kernel accepting a condition and action rule is in appendix. Note, If a rule is ill-formed the system will let the user be aware of and immediately asks for a correct one.

Consistency Controller

With all the linguistic knowledge organized into rigid formats by using the descriptive language, it is simpler to construct the consistency controller, whose purpose is to maintain a consistent logical view of knowledge for our working system. Figure 5 is a general block diagram of the relation between the descriptive language support system and the consistency controller.

In Figure 5, The globally shared data and primitives are controlled by the consistency controller. For example, the category table which is referenced by every linguistic module will be kept here. When requested by the linguist through the descriptive language support system, the consistency controller will do all the global changes and updates the shared data. For example, if a grammar rule is altered, all the rules in other kernels that are dependent on that grammar rule will have to be checked for validity. It is the consistency controller's job to let the linguist know which one might need to be altered. The reason for this is to avoid missing any of the changes needed in other related kernels when the linguistic knowledge is altered in one of the kernels. With all linguistic modules integrated into one descriptive language support system, and with a centralized consistency controller the chance of inconsistency will be small.

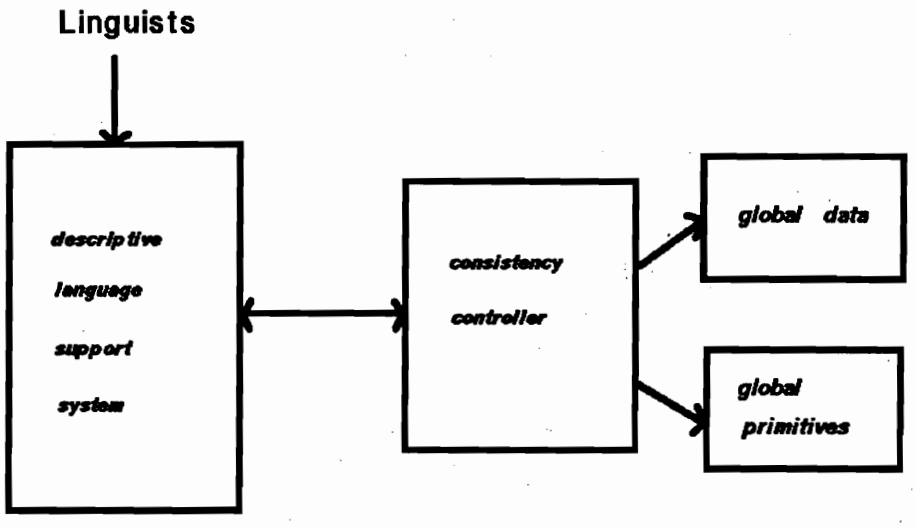


Figure 5. Consistency controller

Conclusion

In a natural language processing system, it is essential to have the linguist transfer and maintain the linguistic knowledge directly with the system. The descriptive language proposed is a linguistic tool that will support the linguists in transferring their linguistic knowledge directly into the system. Moreover, as a direct consequence of having a descriptive language and its support system, the linguistic knowledge will be organized into a clear and definite form. With this, a consistency controller can be constructed for maintaining the knowledge consistency needed in a natural language processing system.

For our machine translation system, a descriptive language is provided and the linguistic kernel for one linguistic module, the Condition and Action, is completed. This descriptive language will allow our linguist to write condition and action rules and transferring them into the system directly.

Appendix

This is an actual session of running the condition and action module.

```
=========================================================
||     WELCOME TO CONDITION AND ACTION MODULE           ||
=========================================================

your name :mei

Select one .. 0 - to exit,
              1 - to lookup,
              2 - to change condition and action,
              3 - to change user defined routine,
```

```
              4 - to initialize condition and action,
              5 - to initialize routines,
0 to 5 => 2


*************************************************************
**      Condition and Action                                **
*************************************************************

Select one .. 0 - to exit,
              1 - to add new condition and action,
              2 - to update condition and action,
              3 - to delete condition and action,
0 to 3 => 1

Name of the condition or action => test1
definition :        ( type "end" or return to exit )
>>[assign (parent.<R_head>) to parent]
>>

comment :         ( return to end )
>>test rule
>>

              /* If the input's format is incorrect, the */
              /* system will show the erroneous input and*/
              /* ask the user to try it again.           */

following is an erroneous input -- lets do it again
              /* system show the bad input to the user   */
-- temporary file --
[assign (parent.<R_head>) to parent]
--------------------



definition :        ( type "end" or return to exit )
>>[assign (parent.<R_head>).<A_aspt> to parent]
>>
accept !



Select one .. 0 - to exit,
              1 - to add new condition and action,
              2 - to update condition and action,
```

```
                3 - to delete condition and action,
0 to 3 => 0

are you sure of the changes? (y/n) y
```

References

[AHO 86] Aho, A.V., R. Sethi and J.D. Ullman, 1986. *Compilers, principles, techniques and tools*, Addison-Wesley, Massachusetts.

[BEES 88] Beesley, K.R. and D. Hefner, 1988. "PERIPHRASE: A High-Level Language for Linguistic Parsing", company communication note from *Automated Language Processing Systems*, Utah.

[BOGU 88] Boguraev, B., J. Carroll, E. Briscoe, and C. Grover, 1988. "Software Support for Practical Grammar Development," *Proceedings of the 12th International Conference on Computational Linguistics*, vol. 1, Budapest, Hungary. PP. 54-58.

[PERE 84] Pereira, F.C.N. and S.M. Shieber, 1984. "The Semantics of Grammar Formalisms seem as Computer Languages", *Proceedings of the Tenth International Conference on Computational Linguistics*, Stanford University, California. PP. 123–129.

[PRAT 73] Pratt, Vaughan R., 1973. "A Linguistics Oriented Programming Language", *Proceedings of the Third International joint Conference on Artificial Intelligence*, Stanford University, California.

[SHIE 84] Shieber, S.M., 1984. "The Design of a Computer Language for Linguistic Information", *Proceedings of the Tenth International Conference on Computational Linguistics*, Stanford University, California. PP. 362–366.

[SHIE 85] Shieber, S.M., 1985. "Criteria for designing computer facilities for linguistic analysis", *Linguistics*, pp. 189-211.

[SHIE 86] Shieber, S.M., 1986. *An Introduction to Unification-Based Approaches to Grammar*, CSLI, Stanford University, California.

[WINO 86] Winograd, T., 1986. "MUIR: A Tool for Language Design", technical report of *CSLI* , Report No. CSLI–87-81, CSLI, Stanford University, California.

[WINO 83] Winograd, T., 1983. *Language as a cognitive process*, Vol. 1, Addison-Wesley, Massachusetts.