Acquisition of Syntactic Properties of English Unknown Words from a Corpus

Tzusheng Pei**, Wei-Chuan Li*, and Tung-Ming Koo*

**Advanced Technology Center
Computer and Communication Research Laboratories (CCRL)
Industrial Technology Research Institute (ITRI)
E000, Blg. 11, 195 Chung Hsing Road, Section 4,
Chutung, Hsinchu, Taiwan, R.O.C.

*Application Software Department, CCRL, ITRI
W300, Blg. 14, 195 Chung Hsing Road, Section 4,
Chutung, Hsinchu, Taiwan, R.O.C.

Abstract

In parsing English sentences, it happens quite often that parsers come across unknown words. A method for obtaining the syntactic properties of unknown English words from a big corpus is presented. To what extend can the syntactic properties of unknown words be obtained is investigated from the view-point of a machine translation system. The acquisition method is based on the contexts of unknown English words in the corpus.

1. Introduction

In parsing natural languages, it happens quite often that parsers come across words which are not listed in the system dictionary. Little can the syntactic properties of the unknown words be acquired if there is one input sentence only. Suppose there is a big corpus, and all sentences in the corpus are syntactically correct. Normally the corpus covers a large amount of sentences in a specific area. For each unknown word, there is no or at least one sentence in the corpus containing that unknown word. For example, a corpus which consists of the first four volumes of UNIX operating system manuals may be a good one. The corpus covers most words widely used in the field of operating system, where some of the words, especially some computer terminologies , may not be listed in the dictionary of a specific machine translation system. The syntax of the source language [BERWICK85] [FASS89] [RUQIAN89] and semantic knowledge of unknown words [BERWICK89] [VELARDI89] can, to some degree, be obtained from the various contexts of the words in the corpus. In this paper, the focus will be on the syntactic properties of unknown words, and the acquisition process will be on a practical machine translation system. In general, the acquisition method can be used in or adapted to other practical machine translation systems. The dictionary can be expanded based on the information obtained. The syntactic properties obtained in this way are

subject to review by people before they are keyed into the system dictionary.

The English-Chinese machine translation system, CCRL-ECMT, formally ERSO-ECMT, has been developed in the Electronics Research and Service Organization (ERSO), ITRI, since July 1987. At present, the project is conducted in Computer and Communication Research Laboratories (CCRL), ITRI. In the machine translation system, an input sentence is preprocessed by the morphological module before being analyzed by the parser. Since there is no entry for unknown words in the dictionary, the morphological module can merely do primitive analysis for unknown words, such as eliminating "d" or "ed" for regular verbs, restoring the ending "ing" to "e" or simply eliminating the ending "ing", and eliminating the ending "s" or "es". Even such primitive analysis can not be done satisfactorily. The information obtained from the morphological processing will be used in later stages. The parser is based on Tomita's parsing algorithm [TOMITA86] with augmented syntactic and semantic tests under the context-free grammar rules. A contex-free rule in the system grammar is applied to construction of parsing trees when the parsing status passes the tests of the rule [TANG88]. The parsing trees constructed by the parser for an English sentence are all syntax trees which already have met the restrictions the tests impose upon.

The possible contexts of a word are the various situations, many of them recorded in the system dictionary, the word may be in. In general, the contexts of a word include information in dictionaries and acceptable sentences, such as semantic markers, verb-types, the grammar rules applied to the current segment of the input sentence containing the word, and the actual segments in acceptable sentences covering the word. In principle, the system dictionary should have abstracts of all possible syntactic and semantic contexts of words in sentences.

In the machine translation system, CCRL-ECMT, the relevant syntactic properties to be determined are as follows:

The first thing to do is to determine the part-of-speech of unknown words in each context. The part-of-speeches in the system are adjective, adverb, do-modal, article, be-verb, conjunction, have-verb, modal, noun, preposition, relative pronoun, "to" for infinitive, verb (other than be-verb and have-verb), subordinate conjunction, wh-word, wh-word as noun, and wh-word as conjunction. The verbs are further classified into 25 categories as shown in the Advanced Learner's Dictionary of Current English [HORNBY63]. A verb might be of several categories. Some categories of verbs may have prepositions, called particles, closely related to them. A verb in these categories and its related preposition normally appear at relative syntactic places in sentences at the same time. The entry of a verb in those categories should contain the related preposition(s). The verb and the related preposition(s) can be used as a restriction or a guide for parsing. The following

patterns show the verb types of 18 and 24 .

VP18: Verb18 + NP + PP,
VP24: Verb24 + PP.

The prepositions for these two verb types should be put in the entries of the verbs of type 18 or 24 in the system dictionary.

For example, "I sold the book to my friend.", "sell" and "to" are closely related in this context. In the sentence, "He succeeded in taking the examination.", "succeed" and "in" are used together. "to" and "in" should be included in the entries for "sell" and "succeed" respectively in the system dictionary.

For some adjectives, there are also particles closely related to them. An adjective of this usage should be used together with its related preposition, so the entry of the adjective should have the related preposition. For example, the preposition "to" normally follow the adjective "available" as in the phrase "available to everyone".

Some nouns have the same property of the adjectives mentioned above. A noun of this category may have a preposition following right after it. For example, ticket "to" Paris, introduction "to" natural language understanding, relation "between" the book and the orange. Depending on the number of English usages the dictionary should reflect, it may contain some more properties of words. The properties show the relations between the word and its various possible contexts.

In practice, it is more convenient to look up a comprehensive dictionary to get syntactic properties of the word unknown to the system than to use this acquisition method, but actually it is not the case. There are many words and/or their usages not covered in any dictinary, and the words and/or their usages are used quite often by people, especially in specific areas, and, furthermore, the problem itself is interesting. Therefore, automation or semi-automation of syntactic knowledge acquisition is still worth study.

2. Determination of the part-of-speeches of an unknown word

Suppose that the unknown word could be any part-of-speech in the grammar, and then the parser tries all part-of-speeches for the unknown word to construct parsing trees for an input sentence containing that word. For each possible syntax, a parsing tree comes out. A sentence may have more than one parsing tree. The parser analyzes the syntax of all sentences containing the unknown word in the corpus. In fact, the unknown word does not necessarily have that many part-of-speeches. For some assumed part-of-speeches, the parser might fail to construct a parsing tree for the sentence. In this case, the word does not have

these part-of-speeches.

In general, each parsing tree the parser has constructed does not necessarily correspond to an acceptable interpretation, because the syntactic structure might not be acceptable to people. Since the word is unknown, there is no way to judge the correctness of the sentence by the semantic information of the unknown word, but the semantic information of the context of the unknown word together with the part-of-speech of it is still useful for making judgement. The final judgement is always the job of people, not the computer system. Furthermore the etymological information of the unknown word, which can not be easily represented and processed on computer, is usually very important for making judgement. Actually, for some cases, the part-of-speech shown in a parsing tree is not a right one for that word. The possible reasons for this are some flaws of the context-free grammar of the parsing system which allows nonexistent syntax, even nonexistent usages, etc. Therefore the actual part-of-speeches of the unknown word will be subset of all the part-of-speeches of the unknown word in the parsing trees the parser has constructed from the sentences containing the unknown word in the corpus.

Suppose that there are N sentences in the corpus containing the unknown word W, and there are M part-of-speeches in the grammar. After having parsed the N sentences, a table is set up. The entry (i, j) of the table is 1 or a blank, where 1 indicates that the unknown word in the j sentence has the part-of-speech i, and a blank hasn't. A column which has two 1s in it indicates that the unknown word has two part-of-speeches and for each part-of-speech at least one parsing tree can be constructed. Each element of the last column is the result of the logical OR operation of all the elements to the left of it. This column shows the set of the possible part-of-speech(s) of the unknown word. An 1 at row i of the last column indicates that the word could have the part-of-speech i. The set of part-of-speeches indicated in the last column is defined as the maximum set.

The determination of part-of-speech(s) of an unknown word is not done alone. The other further properties, which will be discussed in the next section, are also acquired at the same time from the parsing trees constructed by the parser. All candidates for particles will be tested, and the verbs are divided into 25 types to be tested for each type.

Let's see an example. Suppose that there are 5 sentences in the corpus containing the unknown word W, and there are 5 possible part-of-speeches: n (noun), v (verb), a (adjective), z (adverb), p (preposition).

Sentence number \ Class	1	2	3	4	5	OR
1: n	1	1				1
2: v				1		1
3: a						
4: z	1					1
5: p					1	1

The last column is the logical OR operation of the preceding five columns. It shows that the part-of-speech of the unknown word in a sentence could be n, v, z, or p. For actual cases, the verb will be divided into 25 types, and there are various particles. Those usages are considered different and treated separately. In fact, this method gets more part-of-speeches than what the unknown word could have.

The approaching algorithm, which operates on the table mentioned above, is in the following. It picks a subset of the maximum set of part-of-speeches shown in the last column of the table, so, with the part-of-speeches in the subset for the unknown word, at least one parsing tree can be constructed for each sentence in the corpus containing the unknown word. It is based on the intuition that, in this way of selection, the subset, called minimum-covering set, will be as small as possible. A part-of-speech is said to cover a sentence with an unknown word if at least one parsing tree can be constructed for the sentence with the part-of-speech assigned to the unknown word. A minimum set is defined as a subset of the maximum set which has minimum number of part-of-speeches to cover all sentences in the corpus.

The procedure of the algorithm for obtaining minimum-covering set is as follows:

1). For sentences with only one possible part-of-speech for the unknown word, the part-of-speech should be selected to cover the sentence.

2). The part-of-speech which covers as many sentences as possible, excluding the sentences already covered, is selected.

3). Then select the second one and on until the part-of-speeches collected cover all sentences in the corpus containing the unknown word.

Some rarely used part-of-speeches might be lost in this

process. In general, the difference between the number of elements of a minimum-covering set and that of a minimum set is small. Depending on the order of the part-of-speeches being selected, there could be more than one minimum-covering set and more than one minimum set also. Even the numbers of elements of two minimum-covering sets could be unequal.

Up to this point, the maximum set and a minimum-covering set have been obtained. The actual part-of-speeches of the unknown word should contain the minimum-covering set and possibly the whole maximum set. The set difference between the maximum set and the minimum-covering set is interpreted as the set of part-of-speeches rarely used from the view-point of the parsing system. Some of the rarely used part-of-speech(s) might not be acceptable to people.

The useful information acquired for review by people is as follows:

1. The unknown word.

2. The maximum set and the minimum-covering set.

3. A list of the related nodes of the parsing tree, related particle, if any, and the related segment in the sentence for each usage of each part-of-speech in the maximum set. For example, the related nodes for verb type 18 should include Verb18, NP, PP, and the related preposition is also listed.

The same usage of a word in the corpus can be combined. The number of sentences in the corpus showing the same usage is also a useful information. It reflects the frequency of the usage of the word in the corpus. If the corpus is big enough, it actually reflects the frequency of the usage being used by people.

All information obtained along with the parsing system should be reviewed by people before they are keyed into the system dictionary.

3. Further syntactic properties

For further detailed syntactic properties mentioned above, the algorithms for dealing with the properties look for the particles (prepositions) which follow the target syntactic elements, such as a verb, an adjective, and a noun, in input sentences. The parser in the system can be easily modified to do this. The acquisition of the detailed properties and determination of part-of-speech are done at the same time.

The algorithms for the various VP-types are as follows:

The parser tries all verb types in the system grammar to construct parsing trees. The templates from the verb type definition below are used for parser to identify the particles.

VP18: Verb18 + NP + PP,
VP24: Verb24 + PP.

The assumed preposition is treated as a particle, and the parser tries to construct parsing trees. If at least one tree comes out, then the preposition could be a particle. Particles for adjectives and nouns are identified in the same way. Actually, not all particles obtained are acceptable to people. The particles should be reviewed by people before they can be keyed into the system dictionary.

4. Discussion and Perspectives

In order to be meaningful, the corpus should be big enough. It means that unknown words appear once in the corpus should also appear in various contexts in sentences of the corpus that reflect the words' part-of-speech(s) and common usages. The proportion among all usages in the corpus reflects the relative frequencies of the usages of the unknown word. This is the idea case. For a practical corpus in general, it might not contain that many sentences which reflect all usages of the word. From the practical point of view, the processing time for acquisition, mainly the total parsing time, for unknown words is the only limit to the size of the corpus. No statistics has been suggested, since it makes sense only when there is a very big corpus which covers enough sentences containing unknown words.

Here only a few syntactic properties have been investigated. More properties can be incorporated into the system. A lot of further work, such as acquisition of syntax of the source language and semantic information of unknown words, can be done on the parsing system.

In general, the reason of failing to construct a parsing tree for a sentence is that the sentence is ungrammatical from the view-point of the system grammar or one of the word has a new usage not recorded in the system dictionary or there is at least one unknown word in the sentence. In the second case, the acquisition process can not do anything, since there is no way to determine which word has the new usage being used.

Not all results of the acquisition are right, as indicated before, so they should be reviewed by people. Further experiment may indicate the degree the process can be automated and clarify some problems of automatic acquisition. In practice, many unknown words are nouns. Therefore, to set all unknown words as nouns for doing machine translation may be a good policy.

The system grammar is not perfect. It is possible that there are some correct syntax not covered in the grammar, so some usages of the unknown words might not be reflected in the parsing trees constructed. It means that the parser fails to construct parsing trees for those usages. Actually, it is hard or even not

possible to formulate a context free grammar which covers all acceptable English sentences.

There is an interesting phenomenon in English. In general, noun and preposition are not likely to be the part-of-speeches of a word at the same time, but, for noun and adjective pair, the likelihood is much higher. In this case, it is said that the interference between adjective and noun is high. Actually, for a practical machine translation system all nouns can be treated as adjectives. For some part-of-speech pairs, the interference among them is very small or negligible. For example, a word which can be used as a preposition is generally not to be used as other part-of-speeches. So, for some part-of-speeches, they can be eliminated in the syntactic contexts by the parser when a grammar rule is applied to construct a syntactic structure.

5. Acknowledgement

The paper is a partial result of the project No. 3131500 conducted by ITRI under sponsorship of the Ministry of Economic Affairs, R.O.C.

6. References

[BERWICK85] Robert C. Berwick, "The Acquisition of Syntactic Knowledge", The MIT Press, 1985

[BERWICK89] Robert C. Berwick, "Learning Word Meanings from Examples", Chaper three of "Semantic Structures", Edited by David Waltz, Lawrence Erlbaum Associates, Inc., Publisher, 1989, pp. 89-124.

[FASS89] Leona F. Fass, "Learnability of CFLs: Inferring Syntactic Models from Constituent Structure", SIGART Newsletter, April 1989, Number 108, Knowledge Acquisition Special Issue, pp. 175-176

[HORNBY63] A. S. Hornby, E. V. Gatenby, H. Wakefield, Oxford Advanced Learner's Dictionary of Current English, 2nd Edition, 1963

[RUQIAN89] Lu Ruqian, Liu Yinghui, Li Xiaobin, "Computer-Aided Grammar Acquisition in the Chinese Understanding System CUSAGA", IJCAI-1989, pp. 1550-1555

[TANG88] An-Ching Tang, Tzu-Cheng Huang, "Tomita 增強型 LR 剖析器在機器翻譯系統的實際應用",ROCLING I, 1988, pp. 241-256

[TOMITA86] M. Tomita, "Efficient Parsing for Natural Language", Kluwer Academic Publishers, 1986

[VELARDI89] Paola Velardi, Maria Teresa Pazienza, Stefano Magrini, "Acquisition of Semantic Patterns from a Natural Corpus

of Texts", SIGART Newsletter, April 1989, Number 108, Knowledge Acquisition Special Issue, pp. 115-123

Parsing-Driven Generalization for Natural Language Acquisition

Rey-Long Liu & Van-Wun Soo

劉瑞瓏 蘇豐文

Institute of Computer Science

National Tsing Hua University

Proceedings of ROCLING Ⅲ

中華民國第二屆計算語言學研討會論文集351～376頁

Parsing-Driven Generalization for Natural Language Acquisition

Rey-Long Liu and Von-Wun Soo

Institute of Computer Science

National Tsing-Hua University

ABSTRACT

Parsing is an important step in natural language processing. It involves tasks of searching for applicable grammatical rules which can transform natural language sentences into their corresponding parse trees. Therefore parsing can be viewed as problem solving. From this point of view, language acquisition can be generalized from problem solving heuristics. In this paper we show how learning methods can be incorporated into a wait-and-see parser (WASP), the problem solver. We call this approach parsing-driven generalization since learning (acquisition of parsing rules and classification of lexicons) is basically derived from the parsing process.

Three generalization methods are reported in this paper: a simple generalization mechanism, a mechanism of generalization by asking questions, and a mechanism of generalization back-propagations. The simple generalization mechanism generalizes from any two parsing rules whose action parts (right-hand sides) are the same while whose condition parts (left-hand sides) have a single difference. The mechanism of generalization by asking questions is triggered when a "climbing-up" move on a concept hierarchical tree is attempted and is necessary in avoidance of over-generalizations. The generalization back-propagation mechanism is to propagate a confirmed generalization of some later parsing rule back to its precedent rules in a parsing sequence and thus causes them to be generalized as well. This mechanism can save many questions to be asked. With the three generalization methods and a mechanism to maintain lexicon classification (the domain concept hierarchy), we have been able to show a plausible natural language acquisition model.

1. Introduction

Natural language acquisition has been an interesting and challenging research topic for both psycho-linguistists and computer scientists, although the former might be more interested in studying on human subjects while the latter on man-made machines. To understand how a natural language can be acquired by a machine can benefits from both studies. The psycho-linguistical study might be able to reveal many clues, constraints, and limitations regarding to natural language acquisition tasks that human actually face. These revelations may suggest us many guidelines for building computer systems that can automatically acquire a natural language. Our purpose in this paper, however, is not to show psycho-linguistic evidence on human natural language acquisition, rather is to build a computer model using machine learning techniques from artificial intelligence study [7] to demonstrate the feasibilities of natural language acquisition on machines.

There are many reasons to study natural language acquisition on a machine. One among them is to remove the current difficulty of having to construct and maintain a large set of lexicons and grammatical parsing rules in a complex natural language processing system. Since human natural language is ever-growing and evolutionary in nature, no natural language processing system with a complete set of both lexicons and grammatical rules can possibly be built. One solution to this is to seek ways of incrementally acquiring grammatical rules from training examples of sentences while keeping the coherence of the system. To achieve this, the system must have the capability of performing generalization over training examples. Traditional artificial intelligence researches have developed many techniques to perform generalizations [7].

Parsing involves tasks of searching for applicable grammatical rules which can transform natural language sentences into their corresponding parse trees. Therefore a successful parsing sequence can be treated as a solution to a parsing problem. By viewing natural language parsing as problem solving, the language acquisition tasks

can be achieved to some extent by generalizing from problem solving heuristics. We call this approach parsing-driven generalization since learning (acquisition of parsing rules and classification of lexicons) is basically derived from the parsing process.

1.1 Related Works

Berwick [3] uses word features to perform conservative generalizations (the Subset Principle) but leaves their acquisition as an open problem. The acquisition of semantic and syntactic features of lexicons is important for the language development of children (Selfridge [14], Pustejovsky and Bergler [12], Berwick [2] and Zernik [21]). In the system, lexicons are classified in a concept description hierarchy. However, such a concept hierarchy is not built priorly. Instead, it is constructed during the learning process. This contrasts to Mitchell's version space approach (Mitchell [8]), where the domain concept hierarchy must be built before a generalization/specialization process can be carried out.

It has been argued by many researchers (Pinker [11], Wexler and Culicover [16], and Berwick [3]) that no negative examples are necessary in a language acquisition situation. That is, children usually acquire a language by imitating from parents' conversation. It is rare the case that parents teach illegal sentences to their children. However, children do sometimes generate illegal sentences due to overgeneralization from incomplete language acquisition. To avoid overgeneralization, it is necessary for a language acquisition system to be able to ask questions before performing generalization. This is similar to MARVIN's approach of learning (Sammut and Banerji [13]). The generalizer performs experiments by generating sentences and testing their validity by asking the trainer. When the system responses an invalid sentence (might be syntactically invalid such as *He eat an apple,* or might be semantically invalid such as *An apple runs),* the possible generalizations are prohibited. This generalization method shifts the burden of generating negative examples from trainers to learners. Trainers do not have to care about the current state of learners.

Traditional EBL (Utgoff [15], Ellman [4] and Mitchell et. al. [9] [10]) uses high-level domain knowledge to guide correct generalizations. Since parsing is viewed as problem solving, the powerful learning methods such as the Explanation-Based Learning (EBL) can be incorporated into the language acquisition system. In language acquisition, Zernik [20] and Zernik [19] employ the EBL method to acquire phrases and idioms respectively. However, it seems to be impossible to construct a complete knowledge base to perform EBL (Yu [18]). In our system, no such knowledge base is assumed. Rather, we incorporate the "back-propagation" concept of the EBL into the generalizer. The generalizations of the last fired rule are back-propagated when they are confirmed by the trainer.

2. The Parsing Device

The parsing device of this language acquisition system is based on the Wait-And-See strategy.

2.1 The Wait-And-See Strategy

Marcus [6] proposed the Wait-And-See strategy to parse natural languages. It is based on a "determinism hypothesis" which says that natural languages can be parsed by a computationally simple mechanism without backtracking. A Wait-And-See Parser (WASP) is like a production system, where the grammar and parsing heuristics are expressed in terms of rules (parsing rules) which are composed of condition and action parts. Two major data structures are required:

1. active node stack: a pushdown stack of incomplete constituents, and
2. lookahead buffer: a small constituent buffer containing constituents which are complete, but whose higher grammatical function is as yet uncertain.

The rules in a WASP are partitioned into rule packets. Each rule packet contains rules for a particular configuration of the constituent in the top of the node stack. For example, if the top of the node stack contains a VP, the corresponding rule packet for

the VP is activated. However, the selection of rules to fire may depend on the contents of the lookahead buffer and the node stack.

The action operators in a WASP defined by Marcus [6] are:

1. ATTACH: attach the constituent at the top of the buffer stack (X) to the top of the node stack (Y). X is popped from the buffer and becomes the rightmost daughter of Y,
2. DROP: Y is popped from the node stack and pushed onto the buffer stack, and
3. CREATE: push a new active node onto the node stack.

More detailed descriptions for a WASP can be found in Liu and Soo [5] and Marcus [6].

2.2 Extending the WASP

In fact, the creation of an S, an NP, a VP, and a PP may be delayed until its first component is parsed and dropped. At that time, the creation action can be deterministically followed by an ATTACH action which attaches this parsed and dropped component. For example, consider the sentence *I ate an apple.* At the beginning, before creating an S node, the parser might first create an NP node which will attach the noun *I* and then be dropped to the buffer stack. Then, an S node is created, and the parser automatically attach this NP as its first son. Thus, we have an improved version of the action CREATE:

> CREATE: push a new active node onto the node stack, **and** perform the ATTACH action.

This improvement has two advantages:

1. when the first component is parsed and dropped, the parser can lookahead more information, and

2. since the creation action is followed by an attach action, the total number of actions for transforming the input sentence to its parse tree is reduced.

For example, consider again the above sentence *I ate an apple*. If the traditional mechanism (as described in the above section) is employed, there are 15 actions in the solution path:

((CREATE S) (CREATE NP) (ATTACH) (DROP) (ATTACH)
(CREATE VP) (ATTACH) (CREATE NP) (ATTACH) (ATTACH)
(DROP) (ATTACH) (DROP) (ATTACH) (DROP)).

However, if the improved mechanism is employed, there are only 11 actions in the solution path:

((CREATE NP) (DROP) (CREATE S) (CREATE VP) (CREATE NP)
(ATTACH) (DROP) (ATTACH) (DROP) (ATTACH) (DROP)).

This will certainly reduce the number of acquired rules.

Since the rules in a WASP are uniform and suitable for language acquisition systems, it is adopted as the parsing device of the learning system. The parsing operators (ATTACH, DROP and CREATE) become the key action operators in the system. Each acquired rule contains one (and only one) of these operators as its action part.

3. The Learning Module

Fig.1 shows the flowchart of the language acquisition system. After accepting an input sentence and its corresponding parse tree, the system initially sets the node stack to be empty, fills the cells in the lookahead buffer (the number of cells of the lookahead buffer is discussed in later sections), and finds the solution path (sequence of actions that transforms the input sentence to its corresponding parse tree) deterministically.

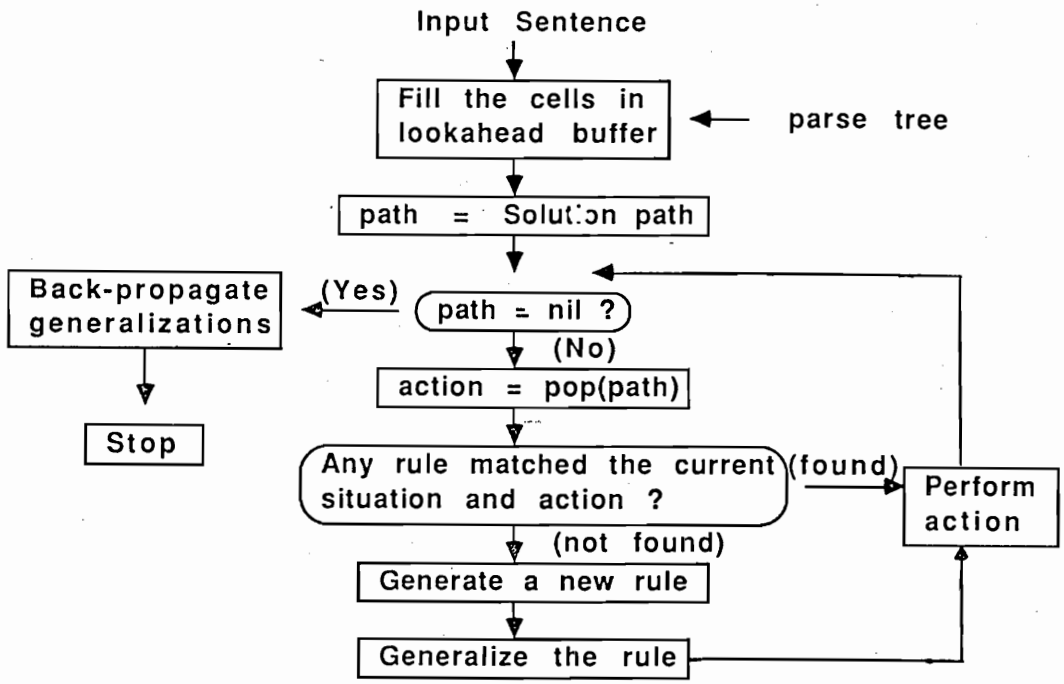

Fig. 1. The overview of the system.

Then, it iteratively pops up the first action in the solution path, tries to find a rule whose RHS (Right Hand Side) matches the action, and LHS (Left Hand Side) condition matches the current configuration of the node stack and the lookahead buffer. If a rule is found, it is fired, otherwise, a new rule is generated according to the current configuration of the node stack and the lookahead buffer (as the LHS of this rule) and the action (as the RHS of this rule). After trying to generalize this new rule, the action is performed, and the iterative process continues until all actions in the solution path are examined. At this time, the system will back-propagate the generalizations from the later acquired or fired rules to the previous acquired rules. Thus, the previous acquired rules are further generalized without causing over-generalizations.

3.1 The input

The semantic bootstraping hypothesis, which is employed in many language acquisition systems (Pinker [11] and Berwick [3]), states that when speaking to a child, parents refer to physical objects using nouns, and actions, which cause the change of the states, using verbs. Thus, although the child might not initially know what the grammatical categories (such as noun, verb, ... etc.) are in the target language, he or

she might also learn them according to the syntax-semantic correspondences. Berwick [3] uses the thematic representations for these correspondences, and in Anderson [1], they are encoded as associated networks which describe the scene of the input sentence. Thus, the trainer must provide not only the training sentences but also their corresponding semantics. In our language acquisition model, the additional information is expressed in the parse trees. For example, for the input sentence *I eat an apple,* the corresponding parse tree is also input to the system:

(S (NP (N I)) (VP (V eat) (NP (DET an) (N apple)))).

In fact, the parse tree provides the learner much information including the categories of words and the structures of phrases. The provision of the category information is also a simplification made by Wexler and Culicover [16]. And as Pinker [11] pointed out, it is possible to derive from the parse tree the lexical entries and the phrase structure rules that generate the input sentence. However, since the derived phrase structure rules are too general, they might sometimes generate syntactically and semantically illegal sentences. What the system wants to acquire are the critical features of words in parsing situations to construct the correct parsing rules without causing over-generalizations. The derivations of general phrase structural rules are not helpful in our problem domain.

Also, from the viewpoint of problem solving, the input sentence can be treated as the initial state, while its corresponding parse tree as the goal state. By properly defining parsing operators (ATTACH, DROP, ... etc.) and representing the parse tree, we can design an algorithm to deterministically detect the whole solution path (the solution path can transform the initial state to the goal state) without resorting to searching or prior domain knowledge. Thus, the given parse tree is not only necessary (in the sense that it represents the syntax-semantic correspondences), but also powerful for the language acquisition system.

3.2 The Determination of a Solution Path

We view language parsing as a problem-solving task which is to transform the input sentence (the initial state) to its corresponding parse tree (the goal state). The LEX system (Mitchell et. al. [10] and Utgoff [15]) formulates the solving of the calculus as a search problem. After a solution is found, specializations and generalizations are then conducted to enhance the performance of the problem solver. The similar idea is used in our language acquisition system. However, by using the input parse tree and the parsing device described above, we can easily determine a solution path without resorting to searching the entire solution space or relying on the high-level domain knowledge (such as the X-bar theory used in Berwick [3]). For example, consider the sentence *I know the beautiful girl* and its parse tree (S (NP (N I)) (VP (V know) (NP (DET the) (ADJ beautiful) (N girl))))). A corresponding solution path can be easily determined (recall the parsing device described in section 2.2)

((CREATE NP) (DROP) (CREATE S) (CREATE VP) (CREATE NP) (ATTACH) (ATTACH) (DROP) (ATTACH) (DROP) (ATTACH) (DROP)).

The determination algorithm scans the parse tree from left to right. When a right parenthesis is encountered, a DROP operator is output, and when a left parenthesis is encountered, a CREATE operator should be output. However, this output of CREATE operator is delayed until its first component is parsed and dropped (section 2.2). The algorithm is as follows:

DetectAction (P)

Input: the parse tree P which is represented as a list.

Output: the solution path for constructing P.

Algorithm:

1. Let C = Car(P); D = Cdr(P).
2. If C is S, NP, VP, PP, ADJP, or ADVP then

2.1. Let CC = (pop D).

2.2. DetectAction (CC).

2.3. Output the action CREATE C.

2.4. For each element CC in D do

2.4.1. DetectAction (CC).

2.4.2. Output the action ATTACH.

2.5. Output the action DROP.

3. Return.

It should be noted that, when the first element (C) of the input tree (P) is not a phrase such as an S (Sentence), NP (Noun Phrase) , VP (Verb Phrase), PP (Prepositional Phrase), ADJP (Adjective Phrase), or ADVP (Adverbial Phrase), no action is output for constructing this element. For example, in step 2.4.1, when CC is "(N I)", the recursive call "DetectAction (CC)" produces nothing, and after this call, an ATTACH action is output in step 2.4.2. The structure "(N I)" only gives category information 'N' to the word 'I'. Since it is not a phrase, no actions are needed to construct it. The parser simply attaches it to the constituent at the top of the node stack. It should also be noted that, by the definition of the action CREATE in section 2.2, when a constituent C is created (step 2.3), it will automatically attach its first son (the constituent CC in step 2.2).

3.3 Generalizations

There are three types of generalizations in the system: simple generalizations, generalization by asking questions, and generalization back-propagations. We will describe them subsequently in this section.

3.3.1 Simple Generalizations

When a new rule is generated, the system will try to generalize it to its largest extent according to the current concept description. Initially, the concept description

(the lexicon) is empty as shown in Fig.2.a. When training examples are provided, the concept description will accumulate the features of each word. For example, after the first sentence *I see the man* and its parse tree (S (NP (N I)) (VP (V see) (NP (DET the) (N man)))) are entered, there are 11 actions detected in its solution path, and thus 11 specific rules (one action for each rule) are acquired. Fig.2.b shows the current concept description.

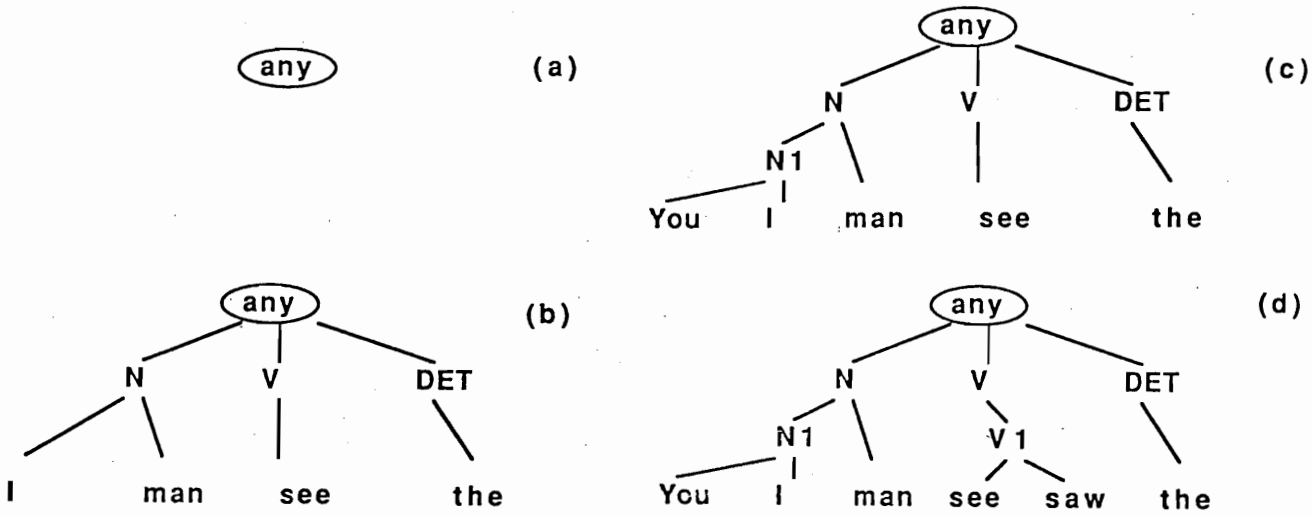

Fig. 2. The concept description.

Since the conception description currently has only specific information, no generalizations are possible in this case. When another sentence *You see the man* is entered, there is only one difference between the previously acquired rule in Fig.3.a and the newly generated rule in Fig.3.b. These two rules are deterministically generalized to the rule shown in Fig.3.c. Fig.2.c illustrates the current concept description (N1 is the acquired general node). Next, suppose the sentence *I saw the man* is entered, the generalizer will generalize from both the rule shown in Fig.3.c and the newly generated rule shown in Fig.3.d. Since there exists a difference (*see* and *saw*) and a more-general relationship (N1 is more general than I) between them, a question *You saw the man* will be asked to ensure the validity of the generalization shown in Fig.3.e. In this case, since this generated sentence is a valid one (confirmed by the trainer), the rule in

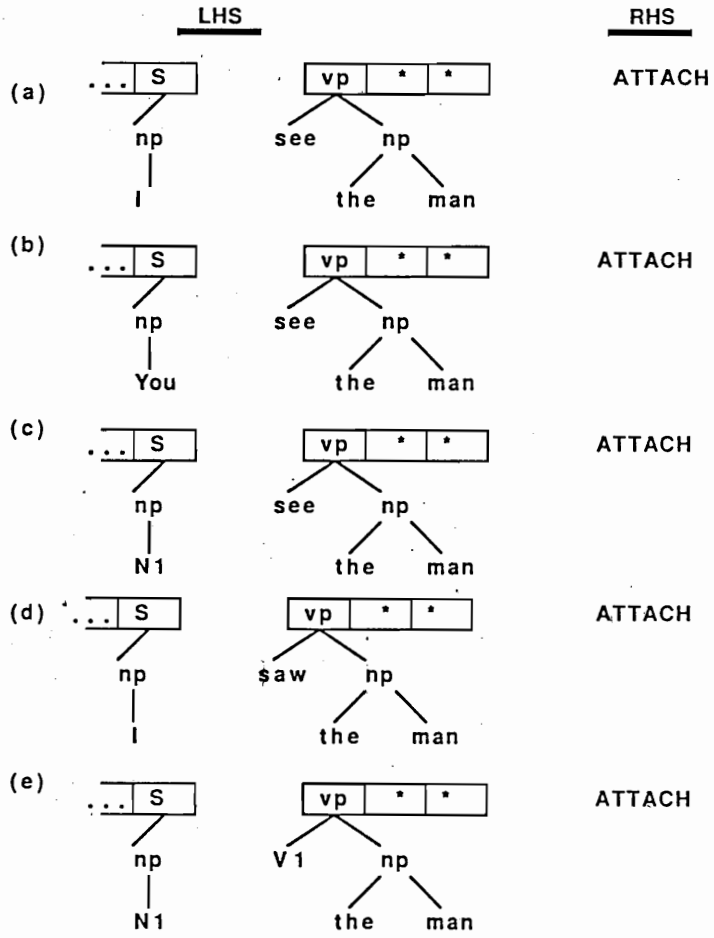

Fig. 3. Generalizations between two rules.

Fig.3.c and the rule in Fig.3.d are generalized to the rule in Fig.3.e. The conception description is also updated as shown in Fig.2.d (V1 is the newly acquired general node). The algorithm of this simple generalization mechanism is as follows:

SimpleGeneralization (R K)

Input: the new rule R and the activated rule packet K.

Output: the resulting rule after generalizing R.

Algorithm:

1. In the rule packet K, find a rule T which has the same action with R's, and there is only one difference between their LHSs.

   If found then

   2.1. If a more-general relationship is possible, ask a question to

justify the generalization.

2.1.1. If the answer is "Yes" then

2.1.1.1. Generalize R and T. Denote the resulting rule by P.

2.1.1.2. Remove T from the rule packet K.

2.1.1.3. Return SimpleGeneralization (P K).

2.1.2. Otherwise, go to step 1 to find another rule.

2.2. Otherwise,

2.2.1. Generalize rule R and T. Denote the resulting rule by P.

2.2.2. Remove rule T from the rule packet K.

2.2.3. Return SimpleGeneralization (P K).

3. Otherwise, return R.

The above simple generalizations have the following features:

1. Generalizations are possible only when the action parts of the two rules are identical and there is at most one difference between their LHSs. If there are more-general relationships between them, questions must be asked to justify the generalizations.
2. When the asked sentences are indeed syntactically invalid (such as *He eat an apple*) or semantically invalid (such as *The apple runs),* the generalizations will be prohibited. With the capability to ask questions, it shifts the burden of generating near-miss (Winston [17]) examples from the trainer to the learner.
3. If a rule is successfully generalized to a new rule, this new rule will be generalized again by the same simple generalization mechanism.
4. There can be no over-generalizations. Each generalization is carefully-justified.

3.3.2 Generalizations by Asking Questions

Another type of generalization is to climb up the existing concept description hierarchy by asking questions. It is performed based on a single rule. For example,

after the sentence *I give the man an apple* is processed, the concept description becomes a hierarchy as shown in Fig.4.a. Fig.4.b shows an acquired rule that can not be generalized by the above methods (since there are more than one differences between this new rule and the old ones).

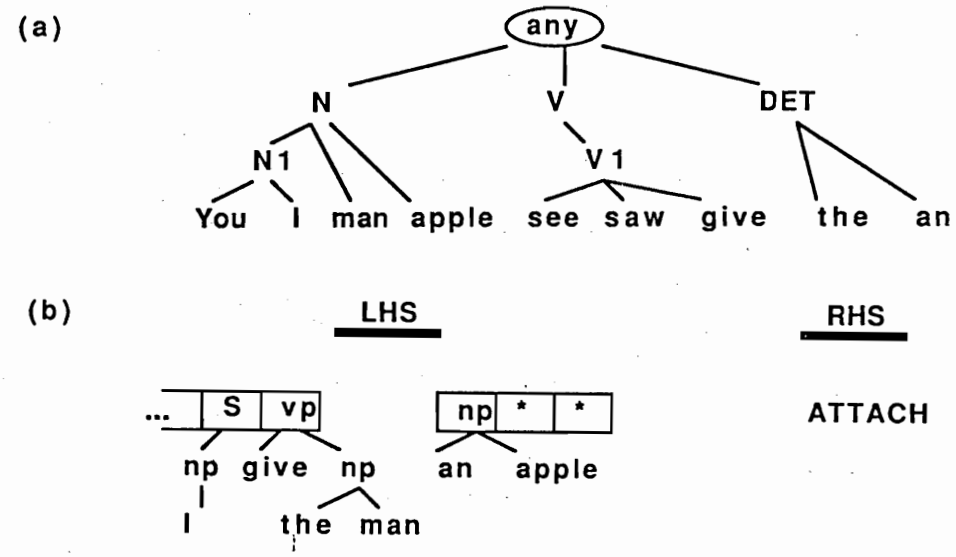

Fig. 4. Generalizations based on single rule.

The generalizer tries to generalize it by climbing the concept description hierarchy. Here, *see* and *saw* must be tested, and thus two questions are asked: *I see the man an apple* and *I saw the man an apple* which are all ungrammatical ones (since the verb *give* is dative, but the verb *see* and *saw* are not). The trainer answers "no" to the system, and thus, no generalizations are performed. The concept description remains unchanged.

Note that the learning system does not know what the newly generated nodes actually "mean". For example, syntactically, there may be Subject-Verb agreements between N1 and V1 in Fig.4.a, and semantically, the system can distinguish *apple* (eatable) from *desk* (un-eatable) by accepting *eat an apple* and not accepting *eat a desk.*

3.3.3 Generalization Back-Propagations

The above generalization by asking question mechanism, employed to climb the

concept hierarchy, is very powerful. However, if the generalizer is allowed to climb the concept hierarchy each time when a new rule is generated, too many questions will be asked. The last type of generalizations is introduced to release this difficulty. It is performed when all necessary rules for parsing the current input sentence are fired. The system will generalize the latest fired rule by climbing the concept hierarchy, and then back-propagate the results of this generalization to previously fired rules. Consider the foilowing example: *I give the man I like an apple.* Assume that the current concept description is shown in Fig.5.a.

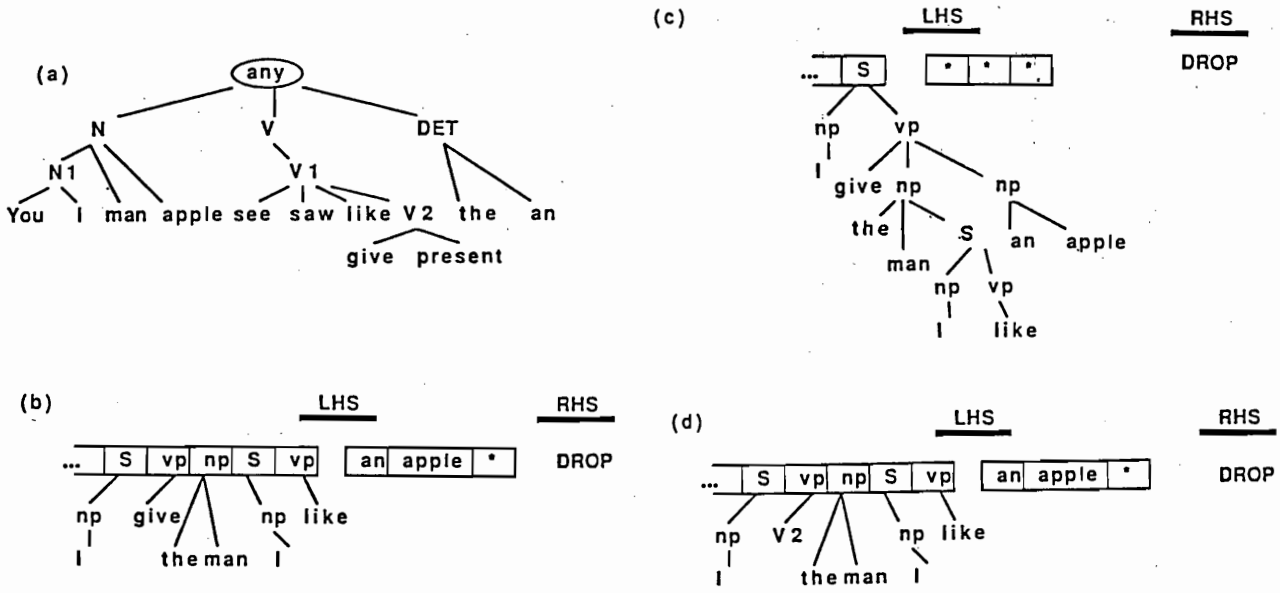

Fig. 5. Generalization Back-Propagations.

After processing this sentence, the set of fired rules for parsing this sentence will contain the rules shown in Fig.5.b and Fig.5.c. The rule shown in Fig.5.c is the last fired rule. The generalizer will try to generalize this rule by climbing the concept hierarchy. In this case, it try to climb the hierarchy from "give" to "V2". Thus, a question is asked: *I present the man I like an apple.* Since this sentence is syntactically and semantically valid (the trainer might answer "yes" to the system), the generalization from *give* to V2 is confirmed and back-propagated to the former one and the *give* in

Fig.5.b will be generalized to V2 which contains *give* and *present* as shown in Fig.5.a. Fig.5.d shows the result of this rule generalization by back-propagations.

Note that, by this method, all generalizations may be back-propagated to many rules. Since the generalizations of the last fired rules have been justified, the back-propagated generalizations will inherently be justified, and therefore will not cause over-generalizations. The number of differences between rules becomes irrelevant in this type of generalizations. This generalization mechanism is one kind of EBL. However, the generalizations are confirmed interactively by trainers (answering "yes" or "no" to the system), rather than justified by predefined domain knowledge. Thus, It is not limited by incomplete prior domain knowledge. Also, by this method, the number of questions asked by the system can be reduced. Thus, this is a very powerful method for conducting generalizations.

3.3.4 The Maintenance of The Concept Description

As described above, the concept description is simultaneously acquired while the system performs rule generalization. The system retrieves and updates the concept description very frequently. Therefore, as the concept description grows larger and larger, to efficiently maintain it becomes very important.

Currently, the concept description is represented as a hierarchy. When performing the generalization process, some concept nodes (such as "N1" and "V1" in Fig.5.a) might be created. The system keeps two information items for each node:

1. a list for recording its father nodes, and
2. a counter for recording how many times a node is referenced by rules.

The father list of a node is updated as follows:

> If a rule, which references a node T in its LHS, is generalized by the simple generalization mechanism, the corresponding new node F now referenced by the rule will be a father of the node T.

While the counter of a node is updated as follows:

> It is incremented by one when a rule, which does not reference the node before, references the node now. On the other hand, it is decremented by one when a rule, which references the node before, does not reference the node any more.

In fact, the counters of concept nodes is updated only when the system generalizes rules:

1. If the system performs simple generalization, two rules are generalized to a more general rule. All concept nodes referenced by these two rules are not referenced by them now, thus their counters should be decremented by one. While for the nodes referenced by the new rule, their counters should be incremented by one.
2. If the system climbs the concept hierarchy from a node M to a node N, it will back-propagate the generalization. Each time a rule is generalized by this back-propagation, the counter of the node M is decremented by one, while the counter of the node N is incremented by one.

If the counter of a node is equal to zero (i.e. no rule references it), this node can be removed, and all its children nodes become the sons of all its father nodes.

4. Implementation

The language acquisition system proposed in this paper is implemented in GCLISP on an PC386 computer. There are about 1000 lines of lisp codes.

For efficiency, acquired rules are classified into the following rule packets: Srule, Nrule, Vrule, Prule, and Orule packets. If the first cell of the node stack in the condition part of a rule is an S, this rule is classified into the Srule packet. If this cell is a VP, it is classified into the Vrule packet, ... etc. Otherwise (such as the node stack is NIL), it is classified into the Orule packet. When searching for appropriate rules to

fire, only those rules in the activated packets are examined.

4.1 A Simple Training Case

Table 1 shows a simple training case. Twelve training examples (12 pairs of input sentences and parse trees) are entered to the system.

Table 1. The training examples for illustrating the learning process.

```
s1:  (It is a book)
t1:  (S (NP (N It)) (VP (V is) (NP (DET a) (N book))))
s2:  (It is a desk)
t2:  (S (NP (N It)) (VP (V is) (NP (DET a) (N desk))))
s3:  (That is a pen)
t3:  (S (NP (N That)) (VP (V is) (NP (DET a) (N pen))))
s4:  (That is a chair)
t4:  (S (NP (N That)) (VP (V is) (NP (DET a) (N chair))))
s5:  (It is a pencil)
t5:  (S (NP (N It)) (VP (V is) (NP (DET a) (N pencil))))
s6:  (It is a pen)
t6:  (S (NP (N It)) (VP (V is) (NP (DET a) (N pen))))
s7:  (That is a desk)
t7:  (S (NP (N That)) (VP (V is) (NP (DET a) (N desk))))
s8:  (That is a pencil)
t8:  (S (NP (N That)) (VP (V is) (NP (DET a) (N pencil))))
s9:  (This is a pencil)
t9:  (S (NP (N This)) (VP (V is) (NP (DET a) (N pencil))))
s10: (This is a book)
t10: (S (NP (N This)) (VP (V is) (NP (DET a) (N book))))
p11: (This is a pen)
t11: (S (NP (N This)) (VP (V is) (NP (DET a) (N pen))))
s12: (This is a desk)
t12: (S (NP (N This)) (VP (V is) (NP (DET a) (N desk))))
```

The learner totally performs 1 time of climbing the concept hierarchy and back-propagating the generalization, and 4 times of simple generalization with asking questions. It successfully acquires 11 rules and classifies "THIS", "THAT", and "IT" into one category, and "PENCIL", "CHAIR", "PEN", "DESK", and "BOOK" into another category. Since the system is allowed to ask questions, it responses additional sentences which are confirmed by the trainer. In this case, the system has learned to parse 15 sentences.

It is interesting to note that:

1. If the learner is only allowed to perform simple generalization process without asking any questions, there are totally 20 rules acquired, and there are more concept nodes in the concept description.
2. If it is allowed to perform simple generalization process with asking questions, only 11 rules are acquired. And during processing these 12 training examples, training examples (s8, t8), (s10, t10), (s11, t11), and (s12, t12) actually contribute nothing to further generalization. That is, before processing them, the learner has already acquired capability to parse them. Thus, the learner could learn to parse 15 sentences from the given 8 (=12-4) training examples.
3. If it is also allowed to climb the concept hierarchy and back-propagate the generalization, training examples (s6 t6), (s7, t7), (s8, t8), (s10 t10), (s11, t11), and (s12, t12) would contribute nothing to further generalization. Thus, the learner actually learns to parse 15 sentences from the given 6 (=12-6) training examples.

4.2 A More Complex Training Case

We illustrate a more complex training case in Table 2 which is to show how the system learn proper PP-attachments (Prepositional phrases attachments). For processing these training examples, the system asked nine questions to perform simple generalizations, among which four were confirmed with "Yes", while five were denied with "No". Also, the system asked six questions to climb the concept hierarchy and to back-propagate generalizations, all of them are confirmed. There are totally 95 rules acquired.

After giving these 13 training examples, the system learned how to determine the proper attachment of the prepositional phrases (headed with *in, on,* and *with)* for the verbs *put, keep, want* and *see.* For verbs *put* and *keep,* if the prepositions are *in* or *on* (training sentence s1, s2, s3, and s4), the prepositional phrases should, although not

Table 2. More complex training examples (their corresponding parse trees are not listed).

s1: (I put the dog in the box)
s2: (I keep the dog on the box)
s3: (You put the dog in the house)
s4: (I put the dog on the table)
s5: (I want the dog in the house)
s6: (I want the dog on the house)
s7: (I see the dog with a telescope)
s8: (I see the dog with a bell)

necessary, be attached to the corresponding verb phrases (VP). On the other hand, if the verb is *want* (training sentence s5 and s6), it is better to attach these prepositional phrases to the corresponding noun phrases. For the verb *see* (training sentence s7 and s8), the way of attaching prepositional phrases depends on the object of this prepositional phrases. For example, in training sentence s7, the prepositional phrase *with a telescope* should be attached to the verb phrase *see the dog*. While for training sentence s8, the prepositional phrase *with a bell* should be attached to the noun phrase *the dog*.

4.3 A Chinese Training Case

In this training case, the following Chinese training sentences are entered to the system:

我 吃 一 個 蘋 果. (I eat an apple) (S1)
我 吃 一 個 梨 子. (I eat a pear) (S2)
我 吃 一 枝 冰 棒. (I eat a ice-rod) (S3)
我 有 一 個 蘋 果. (I have an apple) (S4)

我 吃 一 枝 甘 蔗. (I eat a cane) (S5)

他 吃 一 個 梨 子. (He eats a pear) (S6)

The system totally asked ten questions, and among which five questions are confirmed. After training with these sentences, the system has acquired 23 rules and successfully classified 蘋 果 (apple) and 梨 子 (pear) into a category whose associable unit is 個 , and 冰 棒 (ice-rod) and 甘 蔗 (cane) into a category whose associable unit is 枝 . Besides, it has learned that the subject 我 (I) and 他 (He) share the same verbs 吃 (eat, eats) and 有 (has, have). That is, unlike in English, these subjects and verbs can agree in Chinese. These associable relationships are acquired implicitly in parsing rules. Thus, in general, the system can be applied to not only English but also Chinese language acquisition situations.

5. Conclusion and Discussion

We have proposed a computationally plausible model for natural language acquisition by viewing natural language parsing as a problem solving task. The system can acquire parsing rules and classify lexicons simultaneously without saving training examples. The proposed language acquisition model has 5 major features including:

1. No negative examples are provided by trainers.
2. None of lexicon features are given. This language acquisition model does not require to define a concept hierarchy (the lexicons) priorly, rather, this hierarchy is built incrementally.
3. Since the parse trees are given as input, with the operators for constructing the parse trees, the solution path can be deterministically found.
4. Combining the mechanisms of the generalization by asking question and generalization back-propagation, the number of questions can be reduced, and the learning system is more flexible in the sense that it is not restricted by incomplete

prior domain knowledge.

5. Allowing the learner to ask questions can shifts the burden of generating negative examples from the trainer to the learner.

Since natural language acquisition involves more tasks than merely those in parsing, the parsing-driven generalization approach in this paper only emphasizes on parsing-related learning issues. Even this, there are still many future works remain to be explored:

1. This system has acquired lexicons and parsing rules from the input statement sentences. Other types of sentences (such as command sentences, wh-question sentences, ... etc.) are not yet considered.
2. Since the size of lexicons in a concept hierarchy might grow significantly, to effectively maintain it is an interesting and important problem.
3. Currently, the system cannot acquire the concept of inflections among lexicons. This is due to its incapability to distiguish the inflection relation between lexicons, for example, the lexicon "went" is an inflection of "go". To acquire this, features of inflections among lexicons should be provided.
4. There might be a trade-off in choosing the number of cells in the node stack and the lookahead buffer. Too many cells will cause too specific rules, while too few cells will cause many over-generalizations in rules. Although the Wait-And-See strategy is modified (section 2.2) to promote its ability for resolving ambiguities, the three-celled buffer might still be inadequate. In case this situation is encountered, other mechanisms such as suspension (Liu and Soo [5]) should be introduced.
5. Although the system is allowed to ask questions, the trainer might not answer it, and in this case, the learner can just give up the possible generalizations. How to ask smarter questions is one of the future extensions of our work, and in addition,

how to incorporate more sophisticated generalization back-propagation mechanisms into the learning system is also an interesting problem.

References

1. Anderson J. R., *A Theory of Language Acquisition Based on General Learning Principles,* IJCAI, 1981.
2. Berwick R. C., *Learning Word Meanings from Examples,* IJCAI, 1983.
3. Berwick R. C., *The Acquisition of Syntactic Knowledge,* The MIT Press, Cambridge, Massachusetts, London, England, 1985.
4. Ellman T., *Explanation-Based Learning: A Survey of Programs and Perspective,* The ACM Computing Surveys, Vol. 21, No. 2, June 1989.
5. Liu R. L. and Soo V. W., *Dealing with Ambiguities in English Conjunctions And Comparatives by A Deterministic Parser,* International Journal of Pattern Recognition and Artificial Intelligence, to appear.
6. Marcus M. P., *A Theory of Syntactic Recognition for Natural Language,* The MIT Press, Cambridge, Massachusetts, London, England, 1980.
7. Michalski R. S., Carbonell J. G., and Mitchell T. M. (Eds.), *Machine Learning: An Artificial Intelligence Approach,* Vol. 1, 1983, and Vol. 2, 1986.
8. Mitchell T. M., *Version Space: An Approach to Concept Learning,* Ph.D. diss., Standford University, 1978.
9. Mitchell T. M., Keller R. M., and Kedar-Cabelli S. T. *Explanation-Based Generalization: A Unifying View* Machine Learning 1:47-80, 1986.
10. Mitchell T. M., Utgoff P. E. and Banerji R. B., *Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics,* in Machine Learning: An Artificial Intelligence Approach, Vol. 1, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), 1983.

11. Pinker S., *Language Learnability and Language Development,* The Harvard University Press, Cambridge, Massachusetts, London, England, 1984.

12. Pustejovsky J. and Bergler S., *The Acquisition of Conceptual Structure for The Lexicon,* AAAI, 1987.

13. Sammut C. and Banerji R. B., *Learning Concepts by Asking Questions,* in Machine Learning: An Artificial Intelligence Approach, Vol. 2, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), 1986.

14. Selfridge M., *A Computer Model of Child Language Acquisition,* IJCAI, 1981.

15. Utgoff P. E., *Shift of Bias for Inductive Concept Learning,* in Machine Learning: An Artificial Intelligence Approach, Vol. 2, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), 1986.

16. Wexler K. and Culicover P. W., *Formal Principles of Language Acquisition,* The MIT Press, Cambridge, Massachusetts, London, England, 1980.

17. Winston P. H., *Learning Class Descriptions from Samples,* in Artificial Intelligence, 2nd edition, Addison-Wesley Publishing Company, 1984.

18. Yu Jing-Chang, *Causal Models and Learning From Experiments in Imperfect Domain Theory,* M.S. thesis, Institute of Computer Science, National Tsing Hua University, R.O.C., 1989.

19. Zernik U., *Learning Idioms --- With and Without Explanation,* IJCAI, 1987.

20. Zernik U., *Language Acquisition: Learning a Hierarchy of Phrases,* IJCAI, 1987.

21. Zernik U., *Lexicon Acquisition: Learning from Corpus by Capitalizing on Lexical Categories", IJCAI, 1989.*