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.

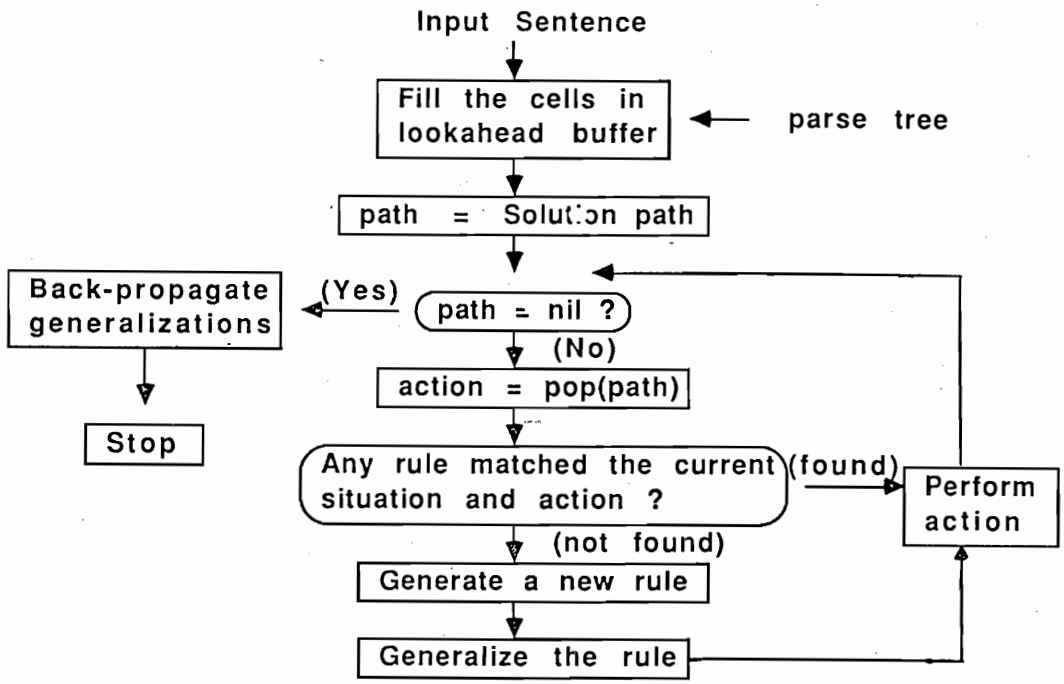

Fig. 1. The overview of the system.

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.

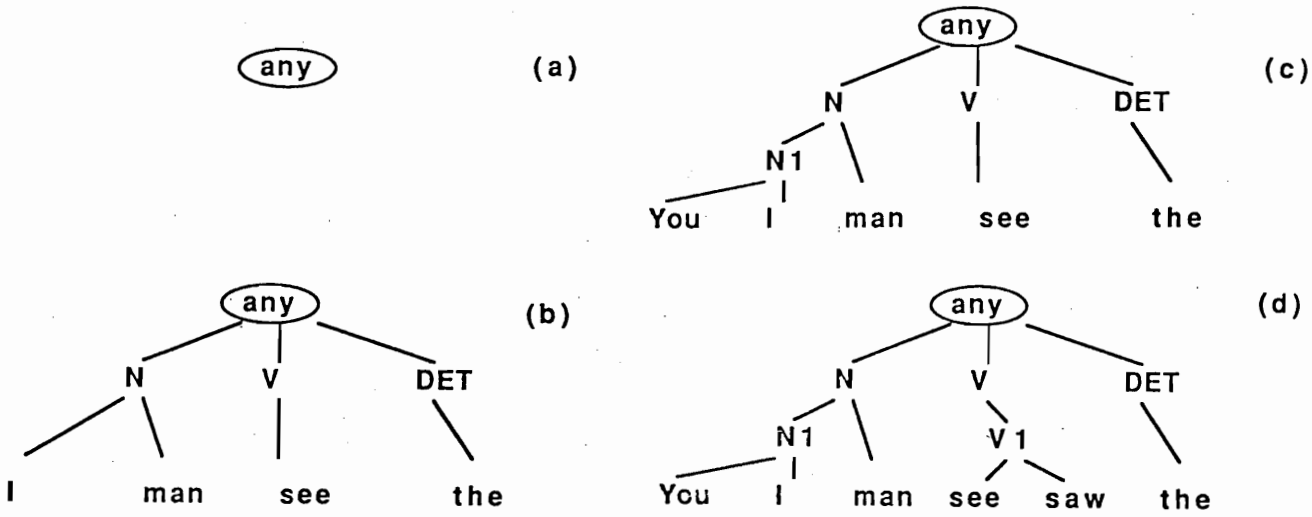

Fig. 2. The concept description.

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

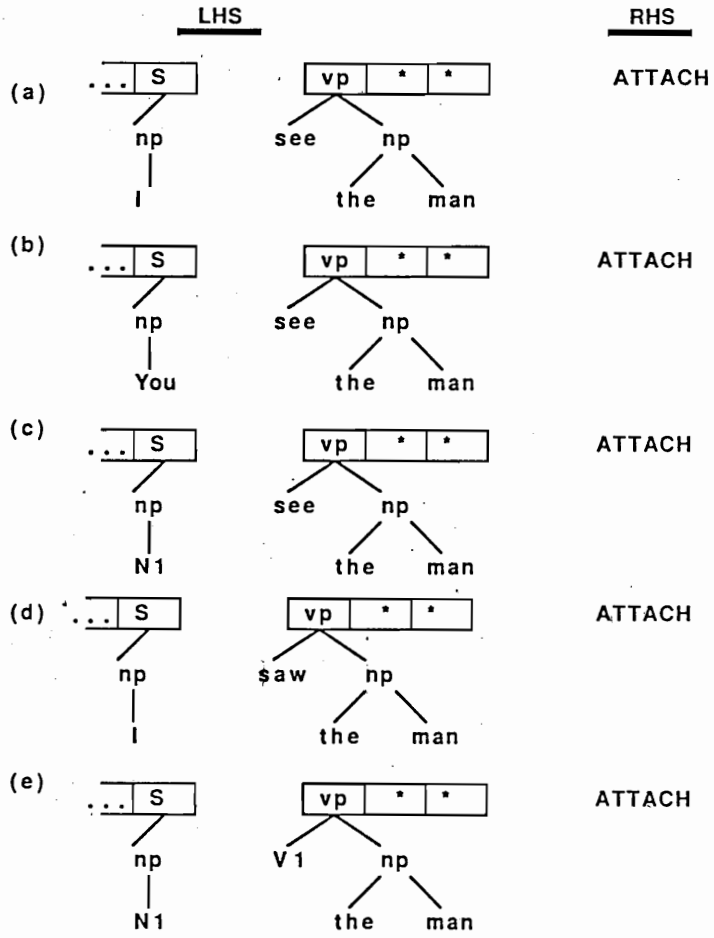

Fig. 3. Generalizations between two rules.

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).

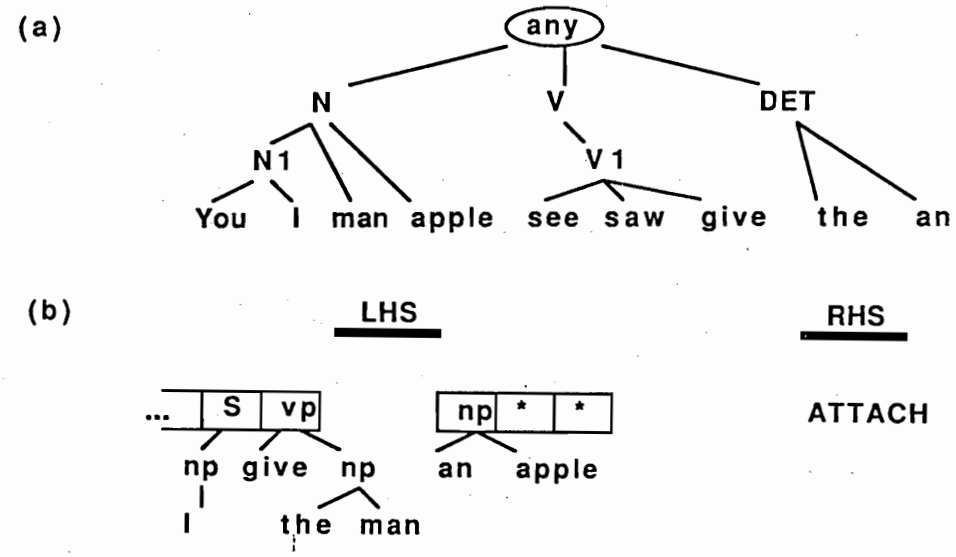

Fig. 4. Generalizations based on single rule.

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.

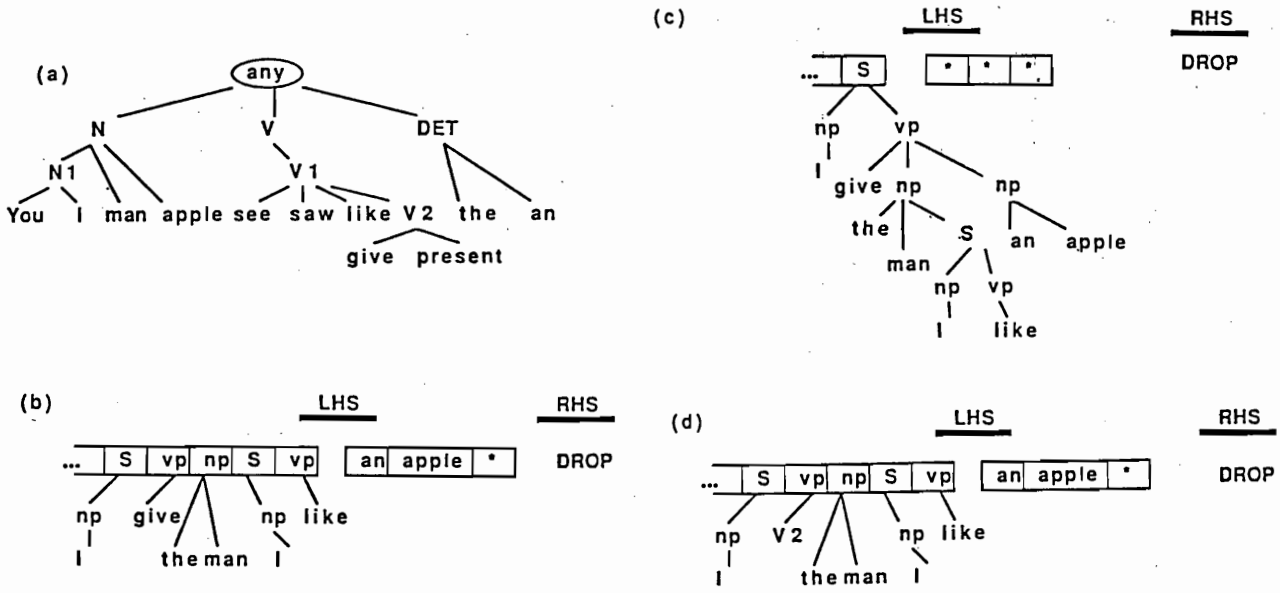

Fig. 5. Generalization Back-Propagations.

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.*