Parsing English Conjunctions And Comparatives

Using The Wait-And-See Strategy

Rey-Long Liu and Von-Wun Soo

Institute of Computer Science

National Tsing-Hua University

ABSTRACT

The major problems in parsing conjunction and comparative English sentences are ambiguities of the scoping and the ellipsis. For a correct parsing, the parser must use not only the syntax but also the semantic information of these sentences. However, as Chiang et. al. [1] pointed out, the semantic information of these sentences can only be obtained after these sentences have been parsed. It is also the reason why a syntax-directed parsing strategy without collecting adequate semantics of input sentences needs to backtrack each time when it makes incorrect assumptions during parsing.

The Wait-And-See strategy, introduced by Marcus [2], is based on the "determinism hypothesis" which claims that the natural language can be parsed by a computationally simple mechanism without backtracking. In this paper, we show a method using the Wait-And-See strategy to parse conjunctions and comparatives simultaneously. In order to enhance the efficiency and correctness of the parser, several mechanisms such as bottom-up preparsing, suspension, and pattern matching are implemented. The bottom-up preparsing looks up the dictionary and recognizes isolated sentence fragments which can be determined without ambiguities. Suspension allows the parser to suspend temporally at ambiguous points and continue to parse the rest of the sentence until it obtains necessary information to resolve the ambiguities. Pattern matching uses the concept of symmetry to detect missing components (the ellipses) in the two conjuncted or compared sentence fragments.

1. Introduction

When parsing sentences with conjunction and/or comparative words, it is possible to make incorrect assumptions at some decision points. Ambiguities of scoping and ellipsis are the major problems in parsing conjunctions and comparatives. Scoping problems occur when a parser has no adequate information to detect the boundaries of constituents, while ellipsis problems occur when a parser has no adequate information to determine the missing components.

For solving the scoping ambiguities, Kosy [6] proposed a Wait-And-See strategy to parse conjunctions deterministically. Rules are written separately to handle the detection of the boundaries of constituents (segmentation rules) and the valid attachment of constituents (recombination rules) respectively. Segmentation operations are separated from and always proceed the recombination operations. This parser can parse many complex sentences efficiently. However, it has difficulty when parsing sentence:

John gives Mary the pen that I give you and Bob gives the man who smiles in the classroom an apple.

In order to detect the boundary of the NP *the man who smiles in the classroom,* it needs to use the recombination operation to "recombine" the clause *who smiles in the classroom.* However, this type of interleaving operations is not allowed in their parsing method. Thus when the recombination operation proceeds, it will not have adequate information to determine the scope of the conjunction word *and.*

For solving the ellipsis ambiguities, Huang [8] presented an algorithm to resolve the ambiguities of ellipses including Gapping, Right Node Raising, Reduced Conjunctions. However, the scoping ambiguities remained unaddressed. Kwasny [3] treated conjunctions as ellipses (ungrammatical forms) and handled them with a pattern matching method. When a conjunction word is seen, patterns are generated dynamically from already identified elements and matched against the remaining segments of an input sentence. This treatment reduces the size of grammar rules and handles the ellipsis ambiguity problem very well. However, the scoping ambiguity problem still remains unsolved. For example, when parsing sentence:

John gives Mary the pen that I give you and Bob gives Jane a pen., a simple pattern matching can not determine which grammatical constituents are actually conjuncted by the conjunction.

Huang [8], Ryan [9], and Chiang et. al. [1] had analyzed many sentences of different conjunction and comparative types. Chiang et. al. [1] also implemented an ATN parser for parsing such kind of sentences. Their parser requires preparsing the basic terms (including noun phrases, verbs, conjunction words, and prepositions) which reduces the reconstruction of basic terms when backtracking. And while parsing basic terms, the parser collects semantic information of the sentence for later construction. Thus, the efficiency is promoted. However, as they pointed out, there is still one drawback in their parser, --- it cannot deal with the sentence which has both conjunction and comparative words. This is because the ATNs for conjunction and the ATNs for comparatives are written independently. We must write other ATNs to handle a sentence with both conjunction and comparative words. However, this could cause too much overhead.

For solving scoping and ellipsis problems and parsing conjunctions and comparatives simultaneously and deterministically, we implemented an efficient parser based on the Wait-And-See strategy.

2. The Wait-And-See Strategy

The Wait-And-See strategy, introduced by Marcus [2], is based on the "determinism hypothesis" which says that a natural language can be parsed by a computationally simple mechanism without backtracking.

A Wait-And-See Parser (WASP) has a production system architecture, whose grammar and parsing heuristics are expressed in terms of rules which are composed of condition and action parts. Two major data structures, defined by Marcus [2], are:

1. active node stack: a pushdown stack of incomplete constituents,

2. lookahead buffer: a small constituent buffer containing constituents which are complete, but whose higher grammatical function is as yet uncertain.

In general, the rules in a WASP are partitioned into rule packets. Each rule packet contains rules which are particularly for the configuration of the top of the node stack. For example, if the top of the node stack is a VP, the corresponding rule packet for the VP is activated. However, the selection of which rule to fire may depend on the contents of the lookahead buffer and the node stack. Readers who are not familiar with the Wait-And-See strategy are referred to a chapter in Allen's book [10].

Since a WASP partitions its knowledge base into independent parts, it has the merits of modularity. We can extend easily to handle more complex type of sentences, and introduce heuristics for each part of knowledge individually to take care of different types of sentences. However, there are still some tasks to be made to improve efficiency --- including bottom-up preparsing, suspension, and pattern matching which are to be discussed in detail in section 3, section 4, and section 5 respectively.

3. Bottom-up Preparsing

According to Winston [7], a WASP requires preparsing the NPs in the original input sentence. In general, simple NPs can be preparsed deterministically, but not a complex NP.

We introduce the bottom-up feature of parsing to promote the efficiency of the parser. In fact, the bottom-up preparsing looks up the dictionary and performs a simple type of pattern matching to recognize isolated sentence fragments which can be determined without any ambiguities.

There are four types of grammatical constituents to be preparsed:

a. word types:

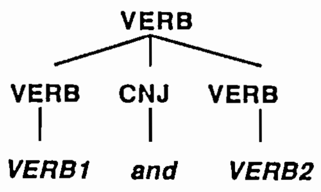
e. g. VERB, NOUN, PREPOSITION, ..., etc.

b. simple NPs:

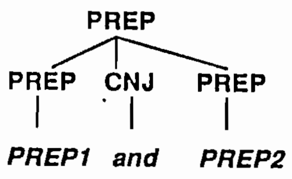
e. g. [DET] (ADJ)* [NOUN].

c. simple conjunctions of words with the same types:

e. g. For a pattern like "VERB1 and VERB2" where VERB1 and VERB2 share the same verb type, we treat it as a VERB and the following tree is constructed:



Similarly, for the pattern like "PREP1 and PREP2" where both PREP1 and PREP2 are prepositions, we combine two conjuncted prepositions into one without any ambiguities:



d. idioms

e. g. "take care of" may be treated as a VERB.

For example, if the input sentence is:

I meet and take care of the patient at and through the night.,

the result after bottom-up preparsing will be:

((NP I)
(VERB (VERB meet) (CNJ and) (VERB (take care of))
(PREP (PREP at) (CNJ and) (PREP through))

(NP (DET the)(NP night)))

Preparsing obtains a lot of important information for our WASP. This will contribute greatly to a correct parsing.

4. Suspension

Hayes et. al. [4] used the concept of parsing suspension to the problem of interjection, restart, and implicit termination in spoken and written languages. The main purpose of its parsing suspension is to provide a flexible way to ignore the input mismatch. In our problem domain, the suspension used here is quite different from that in Hayes et. al. [4]. In order to parse a sentence deterministically without backtracking, a simple lookahead (lookaheading simple words) might not be sufficient. What a parser needs to "lookahead" may be grammatical constituents (e.g. VPs, PPs,... etc) which could only be obtained by "parsing". The parsing suspension mechanisms will be suitable for not only the conjunction and comparative sentences but also for cases where a grammatical constituent lookahead is needed (such as the trace assignment problem mentioned in Cheung [5]). Three types of suspensions are implemented in our WASP:

1. Suspension for scoping ambiguity.
2. Suspension for ellipses before the conjunction words.
3. suspension for pattern formation and for subsequent pattern matching. The first two types of suspensions are discussed in this section and the third type is discussed in the next section.

For parsing conjunctions, ambiguous point might occur in two conjuncted NPs. There are two reasons for suspending the binding of the conjuncted NPs. The first one is that an NP may have two roles in a sentence --- either the subject or the object, but never both. For example, consider the example:

The pen that I give you and Bob gives Jane costs five dollars. (1)

When a parser encounters *you and Bob,* it does not yet have adequate information to determine the role of the NP *Bob* presumably the parser scans the sentence from left to right. Fig.1 shows the parse tree of this sentence.

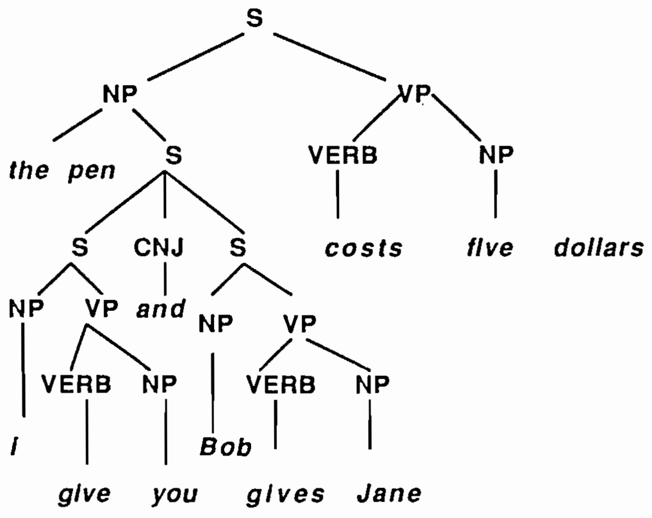


Fig. 1. The parse tree of the sentence:
"The pen that I give you and Bob gives Jane costs five dollars.""

The second reason is that even if the role of an NP is determined, the binding may be still ambiguous. Consider the following examples:

John gives Mary the pen that I give you and Bob gives Jane a pen. (2)

and

John gives Mary the pen that I give you and Bob gives Jane. (3)

Although the NP *Bob* in both sentence is a subject, the presence of the NP *a pen* determines the binding of two conjuncted sentences. In sentence (2) the sentence *Bob gives Jane a pen* should be conjuncted with the major sentence *John gives Mary the pen ...*, while in sentence (3), the sentence *Bob gives Jane* is conjuncted with *I give* *you,* and then the whole conjuncted sentence will serve as a clause. The parse tree for sentence (2) and sentence (3) are shown in Fig.2 and Fig.3 respectively.

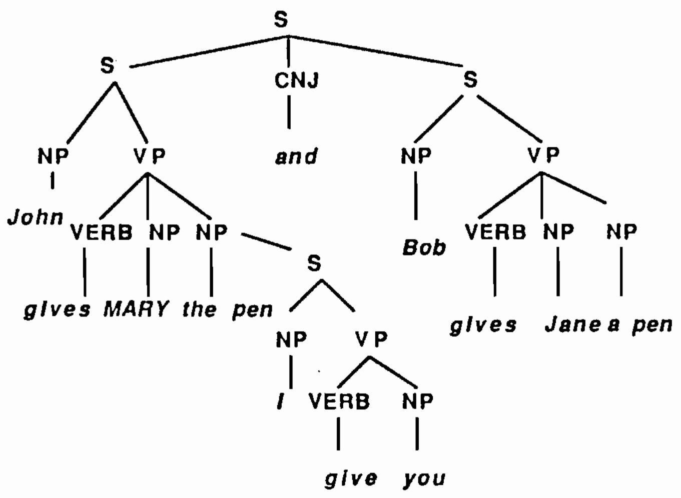


Fig. 2. The parse tree of the sentence:
"John gives Mary the pen that I give you and Bob gives Jane a pen."

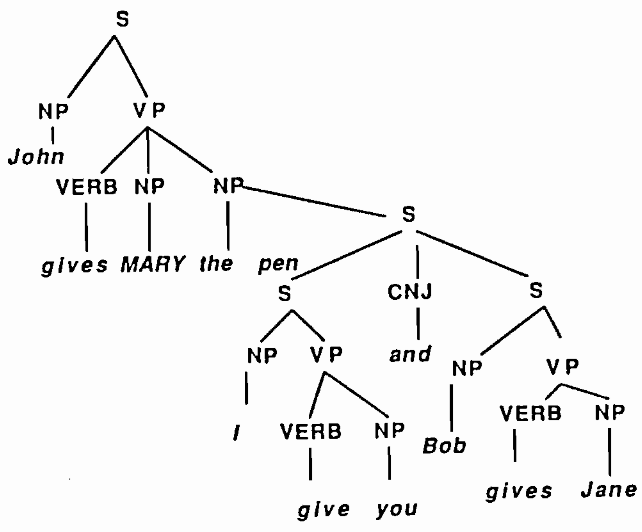
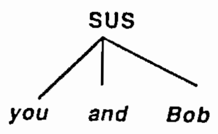


Fig. 3. The parse tree for the sentence:
"John gives Mary the pen that I give you and Bob gives Jane.".

Thus, a parser must collect adequate information to determine the roles of these NPs and ways of binding conjuncted grammatical constituents. It might be necessary for a parser to lookahead. However, what it needs to lookahead may be a grammatical constituent (e.g. a VP, S, ...) rather than words. So, it is necessary to suspend the parsing in order to lookahead for a needed grammatical constituent. Consider this example:

John gives Mary the pen that I give you and Bob gives the man who smiles.

When a parser encounters the conjuncted NPs --- *you and Bob,* it is necessary to determine the grammatical role (subject or object) of the NP *Bob* and the way of binding. In order to make a correct decision, it is necessary to collect more information from the input following *Bob*. So our WASP pushes a suspension node (SUS) containing the ambiguous part *you and Bob* onto the node stack:



The parsing will continue from the word immediately following *Bob,* i.e. the verb *gives*. After getting the grammatical constituent (in this case it is a VP) following the suspension node, our WASP may have a clear view about the sentence structure to make a correct binding for NPs in the suspension node. In this case, the NP *Bob* should be a subject of a sentence which is conjuncted with the sentence *I give you.* And this solves the scoping ambiguity problem of the conjunction. A complete parse tree is shown in Fig.4.

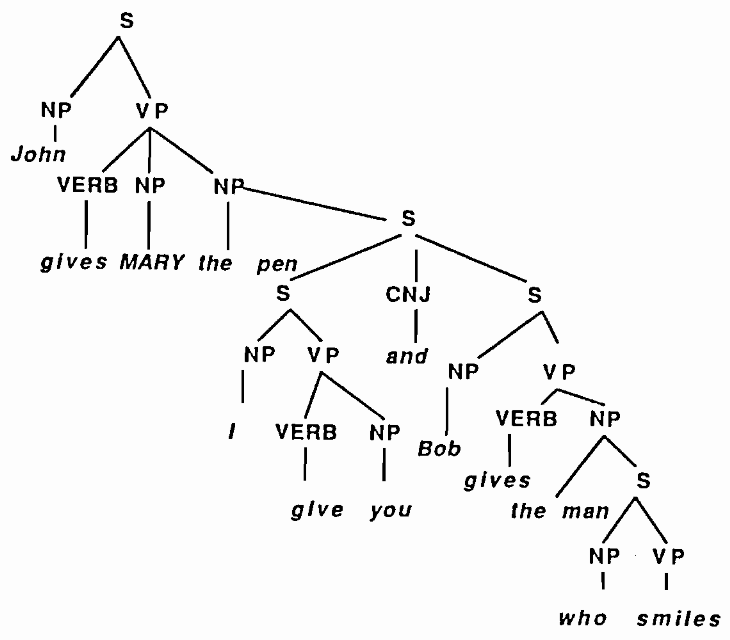


Fig. 4. The parse tree for the sentence:
"John gives Mary the pen that I give you and Bob gives the man who smiles".

The second type of suspension is used to solve the ambiguity problem of the ellipses which occurs before the conjunction word. The missing constituents might be found only when the constituents after the conjunction word have been parsed. Thus a suspension is introduced here. Consider the example:

The man kicked and the woman played the ball.

Since the verb *kicked* is transitive, there must be a missing NP before the conjunction word *and.* The parser suspends this ambiguity here and continues to parse the components after the conjunction word. After parsing the constituent after the conjunction word (in this case, it is an S) the suspension is resumed, and the missed component (in this case, it is the NP *the ball)* can be found and copied.

It should be noted that our parser acts in a one-pass and backtrack-free manner regardless the introduced suspension mechanism. And since there is no work done in vain during parsing, the way of parsing is very efficient.

5. Pattern matching

The "symmetric property" of conjunctions and comparatives is an important feature that can be used to parse these sentences. The symmetric property means that any two conjuncted or compared constituents (NPs, PPs, VPs, or S) will have similar syntactical structures. Thus when handling ellipses in these sentences, the syntactical patterns of these two constituents may be compared (matched) to determine the ellipses. This is a basic approach for parsing conjunctions and/or comparatives.

Consider the example:

I ate an apple and John a hotdog.

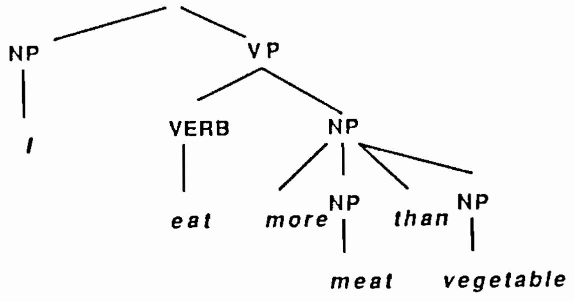
By comparing the syntactical structures before and after *and,* the parser can easily find the ellipses in this sentence, and treat this sentence as:

I ate an apple and John ate *a hotdog.*

When there is an incomplete syntactical structure (e.g. a VP which is lack of an object, a PP without an NP,... etc.) and a conjunction or a comparative word, the pattern matching is necessary to "fill the gap" of these syntactical structure. For example:

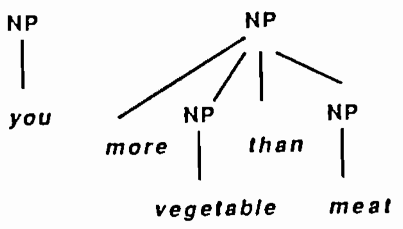
I eat more meat than vegetable and you more vegetable than meat.

When the parsing process proceeds to the conjunction word *and,* a parser has parsed a complete sentence *I eat more meat than vegetable,* and will have the following partial parse tree:



(P1)

However, the pattern following the conjunction word *and* is an incomplete one:



(P2)

These two patterns P1 and P2 must be matched and combined to get the whole complete sentence. The parse tree is shown in Fig.5.

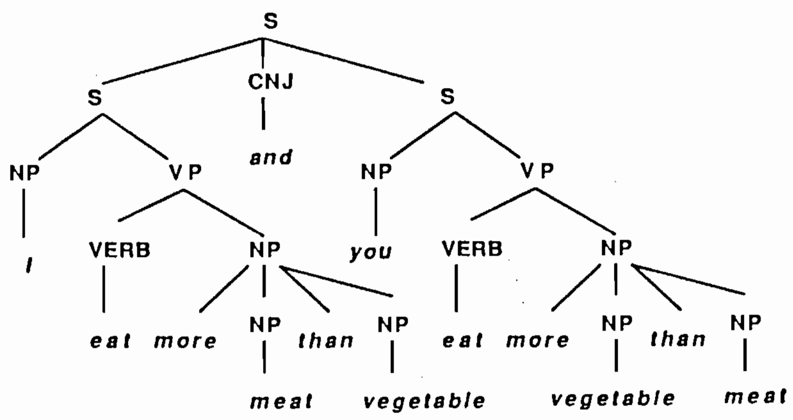


Fig. 5. The parse tree for sentence: *"I eat more meat than vegetable and you more vegetable than meat."*

The question is: *when and how can a parser form the patterns?* It is obvious that only when patterns are parsed, can a WASP perform pattern matching to solve the ellipsis ambiguity problem. This means that the parser should lookahead in a way similar to the suspension action mentioned in the above section. There are three rules for constructing patterns:

a. If the current node is a suspension node (SUS) and the next input token is a simple NP (directly obtained from preparsing), try to extend the NP to its largest scope, and then attach it to the suspension node.

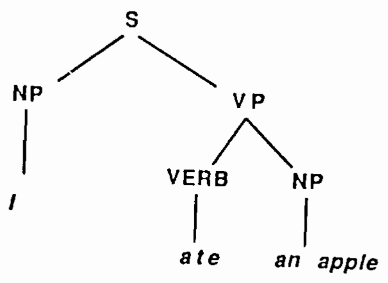
b. If the current node is a suspension node (SUS) and the next input token is a PREP, try to build a complete PP, and then attach it to the suspension node.

c. If the current node is a suspension node (SUS) and the next input token is VERB, the pattern is now formed in the SUS, and the pattern matching is followed.

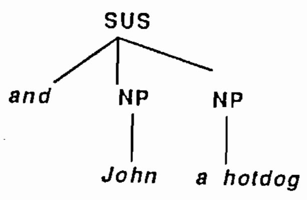
For example, consider the sentence mentioned above:

I ate an apple and John a hotdog.,

the partial parse tree before suspension node is:



and the suspension node is:



Thus, a pattern matching is needed and the verb *ate* is copied. Fig.6 shows the complete parse tree.

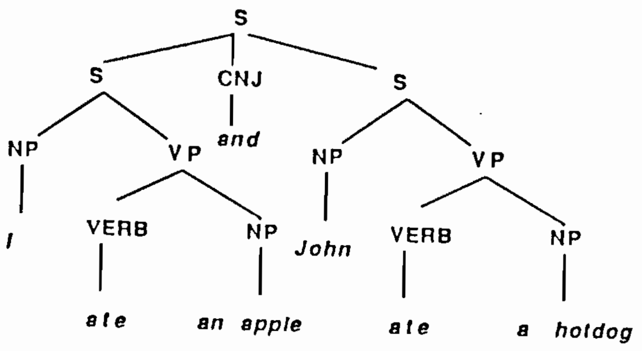


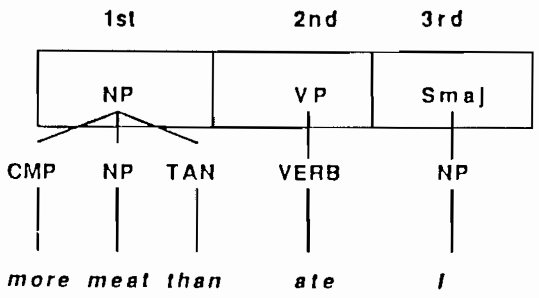
Fig. 6. The parse tree for sentence:
"I ate an apple and John, a hotdog."

Consider a more complex example with both conjunction and comparative words:

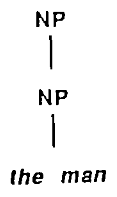
I ate more meat than the man who gave Mary a pen and John a hotdog.

Our WASP will proceed the following steps:

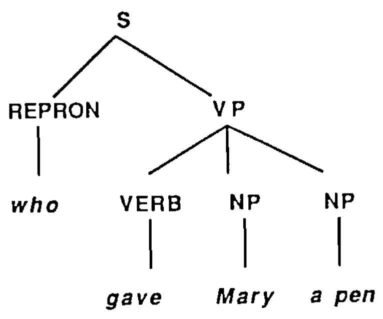
a. When comparative word *more* is encountered, by lookaheading the NP *meat,* our parser concludes that there is a larger NP consisting of comparative words. Thus it tries to build the larger NP. The node stack looks like:



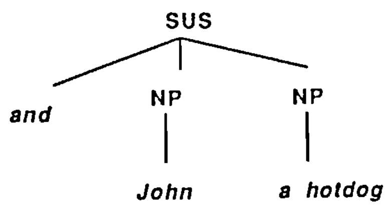
b. When the NP *the man* and the relative pronoun *who* are encountered, Our WASP tries to build a new NP. An NP is pushed, and the top of the node stack is:



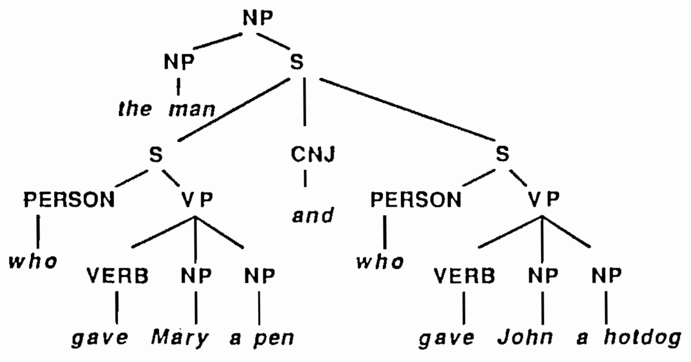
c. When the conjunction word *and* is encountered, the top of the node stack is:



And a suspension node should be constructed as before:



d. Then pattern matching is needed, and the *gave* is copied. And a complete clause is constructed:



e. After the NP *the man who gave ...* is parsed, it can be matched either with *meat* or with *I*. Since *I* and *the man who gave ...* have the same word type --- PERSON, it is better to match these two NPs. Thus, our parser will successfully parse this sentence. The complete parse tree is:

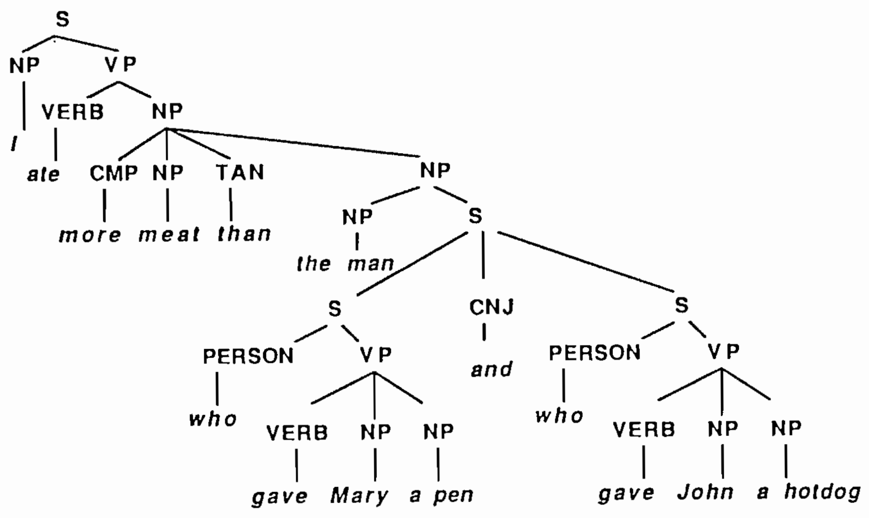


Fig. 7. The parse tree for sentence:
"I ate more meat than the man who gave Mary a pen and John a hotdog."

6. Implementation

The design of our parser has taken into consideration of the sentences which allow comparatives and conjunctions to appear simultaneously at any grammatical constituents. To pay for this capability, additional rules are needed. However, the effort is relative minute.

Our system is currently implemented in LISP and runs under the GCLISP interpreter system on a PC386. There are currently about 100 rules in the rule packets. In Appendix, we illustrate up to 42 sentences to test different patterns of comparative and conjunction sentences. The run time for each sentence is also recorded. Almost all the sentences can be successfully parsed within 300 msec. However, the ambiguities of the attachment of the prepositional phrases can sometimes cause problems. Most of the cases, we found, require more semantic information than actually assumed in our implementation.

Future extension of our work requires a sound and complete dictionary, a better preparsing mechanism to take care of a variety of idioms. How to incorporate more semantic features into the system to guide correct parsing is also an important direction of our research.

7. Conclusion

For parsing English sentences with comparatives and conjunctions, we are concerned with the efficiency and extensibility of a parser. Therefore, we adopt the Wait-and-See strategy to eliminate the backtracking that is a key factor affecting the efficiency. In addition, we introduce such mechanisms as preparsing, suspension, and pattern matching to further promote the power of the parser. The bottom-up preparsing promotes the efficiency by simplifying the subsequent tasks of parsing; the parsing suspension allows to collect information for guiding a backtrack-free parsing and resolves the scoping ambiguities; and the pattern matching resolves the ellipsis ambiguities in the conjunctions and comparatives. Since our WASP is designed in a highly modular and uniform manner, its extensibility is high.

8. Acknowledgement

The authors would like to acknowledge valuable comments from anonymous referees.

9. References

1. Whay-Loong Chiang, Jiunn-Jen Chen, and I-peng Lin, *Generalized Augmented Transition Network for English-Chinese Machine Translation System,* ICS, 1988.
2. Mitchell Marcus, *A Computational Account of Some Constraints on Language,* Theoretical Issues in Natural Language Proceeding-2 D. Waltz, ed., 236-246,Urbana-Champaign, Association for Computational Linguistics, 1978.
3. Stan C. Kwasny and Norman K. Sondheimer, *Relaxation Techniques for Parsing Grammatically Ill-formed Input in Natural Language Understanding System,* American Journal of Computational Linguistics, Volume 7, Number 2, 1988.
4. Philip J. Hayes and George V. Mouradian, *Flexible Parsing,* American Journal of Computational Linguistics, Volume 7, Number 4, 1981.
5. B. Cheung and K. P. Chow, *Universal Feature Instantiation Principles And Wait-And-See Strategy,* ICS, 1988.
6. Donald W. Kosy, *Parsing Conjunctions Deterministically,* Proceedings of the 24th ACL Conference, 78-83, 1986.

7. Patrick H. Winston, *Artificial Intelligence,* 2nd edition, Addison-Wesley Publishing Company, 1984.

8. Xiuming Huang, *Dealing With Conjunctions In A Machine Translation Environment,* Proceedings of COLING 84, Stanford, 243-246.

9. Karen Ryan, *Corepresentational Grammar and Parsing English Comparatives,* Proceedings of the 19th ACL conference, Stanford, 13-18, 1981.

10. James Allen, *Natural Language Understanding,* Chapter 6, The Benjamin/Cummings Publishing Company, Inc., 1987.

Appendix: Table of test sentences successfully parsed.

A. SENTENCE WITH CONJUNCTIONS:	Run Time (sec)
Part 1. SENTENCES WITH SCOPING PROBLEMS	
The story that John told Mary and Bob give the man who was crying a hint.	0.27
The story that John told Mary and Bob told you is a good story.	0.27
The story that John told Mary and Bob is a good story.	0.22
Henry repeated the story that John told Mary and Bob told you.	0.27
Henry repeated the story that John told Mary and Bob told John his opinion.	0.28
The pen that I give you and Bob gives Jane costs five dollars.	0.28
John gives Mary the pen that I give you and Bob gives Jane a pen.	0.27
John gives Mary the pen that I give you and Bob gives Jane.	0.28
John gives Mary the pen that I give you and Bob gives the man who smiles.	0.33
I ate meat and vegetable in the store.	0.16
I played a football and John ate the dinner.	0.17
I give the man who gives Mary and Bob a paper a hint.	0.22
Part 2. SENTENCES WITH ELLIPSIS PROBLEMS	
The man kicked and the woman played the ball.	0.22
John drove the car through and completely demolished a window.	0.22
John played tennis and Jack football.	0.16
I give Mary an apple and John a hotdog.	0.17
I ate an apple and John a hotdog.	0.16
I ate and kicked and the man who are crying ate an apple.	0.27
I give Mary an apple and John a hotdog and an apple.	0.16
I give Mary an apple and John a hotdog and an apple is eaten.	0.22
I played the ball in the store and tennis in the school.	0.22
I ate the dinner slowly and Mary quickly.	0.16
The man kicked the child and ate the dinner.	0.16
I played a football and John ate the dinner.	0.17
I gave the pen to Mary and John to Bob.	0.22
Bob gave the pen to Mary in the store and John in the school.	0.27
Bob gave the pen to Mary in the store and John to Bob in the school.	0.28
I gave the pen to Mary and the apple to Bob.	0.22
The man who gave John an apple and Mary a hotdog kicked the ball.	0.22
B. SENTENCES WITH COMPARATIVES:	
John reads more than most students.	0.16
You run faster than I.	0.11
John has learned more words than Jane.	0.16
John eats more meat than vegetable.	0.17
John reads more than most students do.	0.16
Taller people than I gave the apples to Mary.	0.16
John ate more apple than Mary gave him.	0.16
I give the man taller than you an apple.	0.17
C. SENTENCES WITH BOTH CONJUNCTIONS AND COMPARATIVES:	
John and Bob run faster than Mary and Jane.	0.16
I ate more vegetable and fruit than meat and hotdog.	0.16
John reads more than most students who are crying and I.	0.22
John eats more meat than vegetable and Jane more vegetable than meat.	0.27
I ate more meat than the man who gives Mary a pen and John a hotdog.	0.27