Parsing in Parallel

Xiuming Huang & Louise Guthrie

Computing Research Laboratory
New Mexico State University
Las Cruces, NM, USA 88003

ABSTRACT

The paper is a description of a parallel model for natural language parsing, and a design for its implementation on the Hypercube multiprocessor. The parallel model is based on the Semantic Definite Clause Grammar formalism and integrates syntax and semantics through the communication of processes. The main processess, of which there are six, contain either purely syntactic or purely semantic information, giving the advantage of simple, transparent algorithms dedicated to only one aspect of parsing. Communication between processes is used to impose semantic constraints on the syntactic processes.

1. Introduction

This paper describes parallel model for natural language parsing and gives a design for its implementation. With the advent of parallel machines, it may be possible to view the relation of syntax and semantics in natural language parsing in a wholly new way. The approach is moving towards an application environment which is similar to the heterarchical sytem proposed by Winograd [Winograd 72]. Processes which control the syntactic aspects can be separated from those which control the semantic aspects in that the syntactic processes contain no semantic information themselves, but receive it by communication with the semantic processes, and vice versa. The advantage of this approach is that transparent algorithms can be written that are dedicated to only one aspect of parsing, while the desirable effects of integrating syntax and semantics can be achieved through the communication of processes. In our model we use this communication to enforce semantic constraints on the syntactic processors in order to avoid the combinatorial explosion of producing all legal syntactic possibilities.

Communication between the two components is then our focus in designing a parallel parser. There seem to be three obvious levels at which communication between syntax and semantics can take place: the word level, the phrase level, and the sentence level. We have chosen to consider communication at the phrase level rather than at either of the other two because it would be too early for the syntactic and semantic components to communicate at the word level (too little information is available at this level to help the parsing), and too late for them to communicate at the sentence level (too many syntactic parses might have already been produced). How the communication between the syntactic and semantic components takes place at the phrase level will be described in Section 3.

In Section 4, we design an implementation of this parallel model for a 6-Hypercube [Intel 85] multiprocessing system, which we will have available shortly. The 6-Hypercube has sixty-four identical processors (Intel 80286's with 512K bytes of memory) and no shared memory. Although each node in the Hypercube can eventually communicate with any other node, each processor can directly communicate with only six immediately neighboring nodes. We therefore intend to limit our message passing among processors to immediate neighbors whenever possible.

Like the work of Eiselt [Eiselt 85] on parallel inference processing, we have a perspicuous assignment of natural language processing modules to processors in the machine, although we are suggesting a parallel implementation of a parser with much more parallelism and with a clearer separation of syntax and semantics. The work on "massively parallel parsing" by Waltz and Pollack [Waltz 85] models various components of comprehension by activation and inhibition of nodes of a network. A practical application of their approach requires massively parallel processing, currently beyond the state of the art in multiprocessing.

We base our parallel model on the Semantic Definite Clause Grammars (SDCG) formalism of Huang [Huang 85]. The SDCG evolved from the Definite Clause Grammars of Pereira [Pereira et al 80] and is described below.

2. Semantic Definite Clause Grammars

The SDCG is currently implemented on a single processor machine where it is the parser for the XTRA (English Chinese Sentence Translator) machine translation system [Huang 85]. The XTRA is a prototype system now running under a C-prolog interpreter and has a wide coverage of English phenomena, even though its vocabulary is rather small (1000 entries). The SDCG uses the semantics of words and phrases to restrict the number of syntactic parses of a sentence to those which are semantically compatible.

A simplified version of the SDCG used in the XTRA system is as follows:

```
(1) sentence(s(Subj_Np,
          vp(v(Verb_sense),Obj_Np)) -->

    noun_phrase(Subj_Np),
       is_verb(Verb),
       subject_verb_match(Subj_Np,Verb,
                Verb_sense),
       noun_phrase(Obj_Np),
       verb_object_match(Verb_sense,Obj_Np).
```

The grammar says that an input string is a sentence with the structure *s(Subj_Np, vp(v(Verb_sense), Obj_Np))* if it is composed of *Subj_Np* which is a noun phrase, followed by *Verb* (a verb) whose one sense *Verb_sense* is semantically compatible with *Subj_Np*, followed by *Obj_Np* (a noun phrase) which is semantically compatible with *Verb_sense*.

The sub-grammar for parsing a noun phrase is as follows:

```
(2) noun_phrase(np(det(Det), adj(Adj_sense),
                n(Noun_sense))) -->

          determiner(Det),
          adjective(Adjective),
          noun(Noun),
          adj_noun_match(Adjective, Noun,
                   Adj_sense, Noun_sense).
```

The last predicate in the noun phrase subgrammar, 'adj_noun_match', tries to match Adjective and Noun to find a compatible pair of senses for the given Adjective and Noun to be combined. The predicates 'subj_verb_match' and 'verb_object_match' in the sentence grammar accomplish similar task. All those matches are based on the system of selectional restrictions proposed by [Katz & Fodor 63] and their codings are omitted here to save space. Later we will see how they function.

There is a syntactic lexicon in the SDCG of the following form:

determiner(the).

noun(coach,[coach1,coach2]).

noun(star,[star1,star2]).

adjective(tough,[tough1,tough2,tough3,tough4]).

verb(marry,[marry1,marry2]).

For instance, the syntactic entry for "coach" is a noun having two senses, labeled "coach1" and "coach2".

For each word sense, a semantic interpretation is given in the semantic dictionary:

sem(coach1,[head(thing)])*. (eg. 'a passenger coach')

sem(coach2,[head(man)]). (a trainer)

sem(star1,[head(thing)]). (a celestial object)

sem(star2,[head(man)]). ("a singing star", etc)

sem(tough1,[poss(thing)]). (modifies 'thing', as in "a tough materail")

sem(tough2,[poss(man)]). (modifies 'man', as in "a tough mountaineer")

sem(marry1,[subj(man),obj(man),head(do)]). ("John married Mary.")

sem(marry2,[subj(man),obj(thing),head(do)]). (eg. in "He married money.")

For example, "coach1" labels the sense of "coach" whereby it refers to a "thing". In parsing (3),

(3) The tough coach married a star*.

according to the grammar in (1) the system starts with the predicate 'noun_phrase', which is presented in (2). After it instantiates the variables *Det, Adjective* and *Noun* instantiated to "the", "tough" and "coach", it attempts to apply the predicate 'adj_noun_match', whose task it is to find the first pair of senses for the words "tough" and "coach", respectively, which are compatible with each other according to our selectional restrictions. Here the first pair found would be 'tough1 + coach1', because the semantic category of "coach1" ('thing') fits into the 'poss(thing)' slot of the word sense "tough1" (meaning that his adjectival sense is for modifying something whose semantic category is 'thing').

Now the parser is at the predicate 'is_verb' where it finds the verb "marry". It then tries to match *Subj_Np* ('tough1 + coach1') with a some sense of the "marry" but fails because both "marry1" and "marry2" prefer the subject to be of the semantic category 'man', which "coach1" cannot satisfy. The system backtracks, trying 'adj_noun_match' again and producing the next matching pair of senses for "the tough coach" ('tough2 + coach2'). When 'subj_verb_match' is tried again and it selects 'marry1' as the appropriate verb sense. The parser proceeds to analyse the rest of the sentence, employing "noun_phrase" to find the object noun phrase sense and "verb_obj_match" to see whether this noun phrase sense fits the particular verb sense. 'Star1' (a celestial object) is thus tried and rejected, and 'star2' (a celebrity) is accepted ('marry1' requires the object to be of the semantic category 'man'). A plausible reading of the sentence is thus gained ("The strict trainer married a celebrity.")

It is clear from the above description that in the SDCG syntax and semantics closely interact: syntax - semantics - syntax, etc. One class of predicate waits for the other to make a decision, then makes its own decision. How much backtracking must be done is unpredictable; the parse might only be completed after several routes have been tried and rejected.

---

* The semantic primitives such as 'thing', 'man', etc, are based on the primitive set suggested in [Wilks 75].

* Modified version of the "semantic garden path" sentence by [Charniak 83] ("The astronomer married the star.")

3. Parallel Parsing

The model consists of six processes which communicate to produce all the semantically compatible parses of a given sentence. Each process will be implemented as a tree of processors. The root node of the tree contains a queue of requests and allocates processors to the elements of the queue as they become available. For the purpose of this model it is sufficent to note that each process itself has the capability of processing several requests in parallel. We identify below each of the processes and describe the communication between them.

1) Sentence master - Controlling process which operates as a modified top down syntactic processor (modified in the sense that information from other processes influences its decisions).

2) Noun-phrase master (NP-master) - Given an arbitrary string, it identifies syntactically all possible initial noun phrases in the string. Through communication with the AN-master, it determines which of these are semantically acceptable.

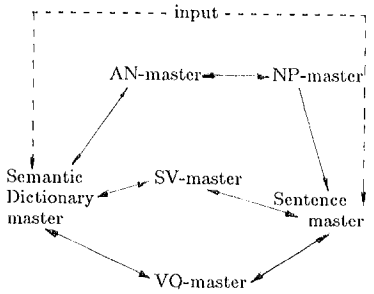
3) Semantic dictionary master - Contains the semantic dictionary and provides appropriate entries for the current input sentence to the other semantic processes.

4) Adjective-Noun master (AN-master) - Given an adjective and a noun, finds all possible pairs (adjective word sense, noun word sense) that are compatible.

5) Subject-Verb master (SV-master) - Given a word sense for a noun and a verb, finds all possible word senses for the verb that are compatible.

6) Verb-Object master (VO-master) - Given a word sense of a verb and a word sense of a noun, determines whether or not that verb sense-object noun sense pair is compatible.

The following diagram illustrates the processes and the communication between them.



Input is read simultaneously by the semantic dictionary, and the sentence master. The sentence master contains the syntactic dictionary and begins a top-down parse of the sentence guided by the definite clause grammar. Whenever a noun phrase is searched for, the noun phrase master is invoked to produce all possible initial noun phrases in the remainder (the unparsed portion) of the input string. After the main verb of any clause has been identified by the sentence master, the SV-master is invoked to produce all possible verb senses which are meaningful at this point in the parse. In the case that a transitive verb is found and a possible word sense for the object noun is determined, the VO-master is consulted as to whether or not the given verb word sense and object noun word sense are acceptable as a verb-object pair.

In communicating with the NP-master or SV-master, several possibilities may be returned to the sentence master, and the parse is continued for each of these possibilities in parallel.

The NP-master, which is also a syntactic process, finds all possible initial noun phrases which are meaningful by using its own syntactic information (in a top down manner) and by communicating with the AN-master for semantic information. This communication is similar to that of the sentence master with the SV-master. After determining an adjective which is followed by a noun, the NP-master invokes the AN-master to find all meaningful adjective-noun word sense pairs. Multiple adjectives which modify a noun are considered in parallel by the AN-master, which in this case, returns pairs which consist of a list of adjective word senses and a noun word sense. Whenever the NP-master recives a pair from the AN-master, it continues any work that it might have (such as finding prepositional phrases which modify the noun, e.g. 'the big boy in the park'). If several pairs are returned by the AN-master, the remainder of the parse is handled by the NP-master and is done in parallel when possible.

The sentence master produces all the parses of the sentence that have not been blocked. A parse may be blocked for any one of the following three reasons:

1) The syntactic category needed by the sentence master is not satisfied by any initial segment of the unparsed portion of the input.
2) The SV-master returns a negative response.
3) The VO-master returns a negative response.

We use the example in Section 2 ("The tough coach married a star.") to illustrate the above communication of processes and to exhibit a path which is blocked.

For simplicity, we write the SDCG used previously, without the arguments for the predicates involved. We also add an additional rule for noun_phrase and another entry in the semantic dictionary for the noun sense of 'tough', tough3 (as in "the tough never suffer"), to make the example interesting.

```
sentence --> noun_phrase, verb,
             subject_verb_match,
             noun_phrase, verb_object_match.

noun_phrase --> determiner, adjective, noun,
                adj_noun_match.

noun_phrase --> determiner, noun.

determiner --> [the].

determiner --> [].
```

The sentence master receives the input and in this case, immediately passes it to the NP-master and waits. The NP-master finds "The tough" and "The tough coach" as possible initial noun phrases in the string it was given. "The tough" (tough3) is returned immediately to the sentence master who begins searching for a verb. Simultaneously, the NP master sends the adjective noun pair, (tough, coach) to the AN-master. The AN-master returns (tough1, coach1) ("rugged vehicle") and (tough2, coach2) ("strict trainer"). Note that these are the same possibilities considered by backtracking in the example in Section 2. The NP-master returns these to the sentence master, who initiates the continuation of the parse for each of these possibilities. The sentence master, in the interim, found a verb (coach) for its first noun-phrase (the tough3) and request a subject-verb match from the SV-master. The SV-master returns coach3 (the verb sense of coach) and the sentence master continues with the remainder of the input string "married a star". Here, a noun_phrase is needed, and so once again the NP-master is invoked, and asked to find an initial noun phrase in the string. Since no noun phrase is found, this path is blocked. The path containing (tough1,coach1) will be blocked exactly as the description in Section 2. The path containing (tough2,coach2) will succeed and produce the correct parse for the sentence.

We now consider the function of the Semantic Dictionary master. While the sentence master is receiving its input and begins the processing described above, the semantic dictionary master simultaneously finds all possible word senses for each input word. The semantic dictionary contains an entry for each sense of a word. The structure of each entry reveals its syntactic category. Word senses corresponding to nouns contain only the semantic class to which the word sense belongs. For example, the semantic dictionary entry for the noun "name" (as in the girl's name) is given by:

sem(name1, [head(sign)]).

Adjective word senses contain the semantic class of the noun that it prefers to modify. The adjective "specific" has the following entry:

sem(specific1, [poss(sign)]).

Word senses corresponding to verbs are described with a structure which contains the class of the subject that is prefered by this verb, the class of the object prefered, and the semantic class of the verb itself. The verb "name" ("to name a dog") is represented as:

sem(name2, [subj(man), obj(man), head(make)]).

After finding all possible word senses for words in the input sentence, the semantic dictionary master sends these dictionary entries to the appropriate semantic processes. Verb entries are sent to the SV- and VO-masters, adjectives are sent to the AN-master, and nouns are sent to all three. These three process masters then contain a "cache" of the semantic dictionary entries relevant to the parsing of the present input sentence. The purpose of the "cache" is so that the semantic dictionary entry for any input word can be quickly found by the processes which use these entries.

4. The Design of the system

We describe the design of the implementation* of the parallel parsing model. Each of the six processes consists of a tree of processors. We label the root of each process tree with the name of the process that it represents. The design of the semantic processors and the noun-phrase master is independant of the implementation of the SDCG which is used. The design of the sentence master, however, is heavily dependant on the formal grammar used for the SDCG implementation as the parser for XTRA. The two syntactic processes above, the NP-master and the sentence master, have a significantly more complex design than those of the semantic processes so that different possible syntactic alternatives may be considered in parallel.

---

*Although the actual implementation has not begun, we hope to do so by summer 1986 when the Hypercube multiprocessor will have been ready for use.

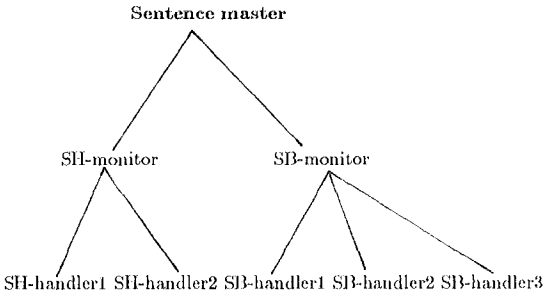
4.1. The sentence master

The design of the sentence master is based on the following production rules of the SDCG:

sentence --> sentence_body.
sentence --> sentence_head, sentence_body.

Intuitively, we can consider the sentence_head to be whatever appears before the sentence subject (it can be an empty string), and the sentence body to be the remainder of the sentence.

The sentence master, as illustrated below, can be thought of as the root of a tree which has two children which we will refer to as the sentence monitors: the sentence head monitor (SH-monitor) and the sentence body monitor (SB-monitor). Each sentence monitor is the root of a subtree of child processors (SH-handlers and SB-handlers) and acts as a monitor for these child processors. We later describe the sentence handlers in more detail.



The sentence master is the process which determines whether or not a string is a sentence. Any input to the sentence master is immediately given to both the SH-monitor and the SB-monitor to examine in parallel the possibilities that the sentence does and does not have a sentence head. The SH-monitors and the SB-monitors each put incoming requests from the sentence master in a queue and allocate the first available child processor to begin its work. In the case of a SH-handler, this work is to identify a possible sentence head, and in the case of an SB-handler, it is to see if the input string is a sentence body. The SH-handlers and SB-handlers monitor child processes which operate in parallel.

In the case that a sentence head is found by one of the SH-handlers, the result is returned to the sentence master via the SH-monitor. The remainder of the input is then given to the SB-monitor which allocates a free SB-handler to continue the parse of the remainder of the sentence. For example, consider the sentence:

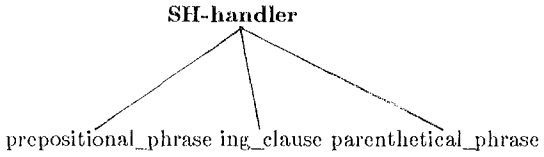
(4) Writing to John was difficult.

The sentence master gives the sentence to both the SH-handler and the SB-monitor which in turn give it to one of their children, say SH-handler1 and SB-handler1. Since the grammar for the SDCG indicates that an ing-clause is a possible sentence head, SH-handler1 will identify "writing to John" as a candidate sentence head. The remainder of the sentence "was difficult" is given to a new SB-handler, say SB-handler2 via the SH-monitor and the SB-monitor, to see if this is a possible sentence body. SB-handler2 fails and notifies SH-handler1 (via the SH- and SB-monitors). SH-handler1 and SB-handler2 become available for other processing and SB-handler1 succeeds in showing that "Writing to John was difficult" is a legal sentence body.

The SH-handlers and the SB-handlers are arrays of processors which implement the or-parallelism of Prolog for the predicates sentence_head and sentence_body respectively. Below is a simplified version of the grammar rules used in the SDCG for sentence_head.

sentence_head --> ing-clause.
sentence_head --> prepositional_phrase.
sentence_head --> adverbial phrase.

Based on these rules, each SH-handler monitors three child processors:



The SB-handlers monitor five processors which are again based on the SDCG. The function of these five child processes will vary depending on the type of the input sentence (declarative, interrogative, or imperative). We give here a simplified version of the sentence_body productions in the SDCG for a declarative sentence.

sentence_body --> subject_np, vp1.
sentence_body --> subject_np, vp2.
sentence_body --> inverted_sentence.

subject_np --> noun_phrase.
subject_np --> ing_clause.

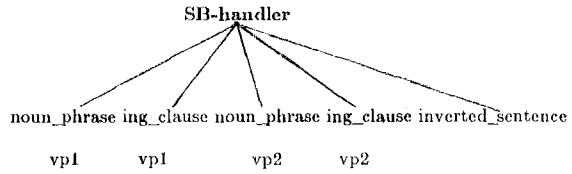
Here vp1 represents a complete verb phrase, like that in the sentence

(5) John didn't go to the park yesterday.

And vp2 represents an elliptical verb phrase, like "didn't" in

(6) No, John didn't.

An illustration of the SB-handlers in this case is given below.



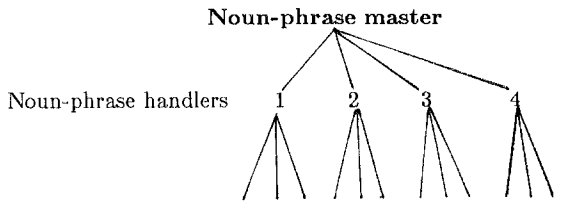
In Section 3 we indicated that the sentence master communicaters with the NP-master. Actually, each of the child processors of the sentence handlers sends a message to the NP-master, via the sentence master, whenever the DCG dictates that a noun phrase should be found next in the input string. The NP-master returns all semantically compatible noun phrases. Where there is more than one acceptable noun phrase, a message is sent to the requesting sentence handler who allocates one possible noun phrase to the waiting child processor and distributes the others to available child processors. Each child process of the sentence handlers communicates with the NP- , SV-, and VO-masters via the sentence master.

It is possible that one of the child processors of the sentence handlers needs to know whether or not some subclause is itself a sentence. For example, if one of the paths of, say, SB-handler1 does a recursive call to check whether or not the next phrase is a sentence (as in a parenthetical expression or a conjunctive sentence), a message is sent to the sentence master to take care of this request. The requesting processor waits.

Should each of the sentence handlers have a waiting child processor and the sentence master a request, we invoke a special processor, called the black-sheep processor, to grant the request, so that the requesting processes may continue. The black-sheep processor, functions precisely as the current single processor implementation of the SDCG and will only be used to avoid deadlock*.

4.2. The Noun-phrase master

Since noun phrases are the major building block of many substructures of a sentence, and since ambiguity often arises through determination of different noun phrases (eg. "The tough coach the young" and "The prime number consecutively"), the identification of noun phrases is an important place for parallelism in the parser. The NP-master can be thought of as the root of a tree of processors. It functions similarly to the sentence master. The noun-phrase master contains a queue of noun-phrase requests and allocates them to available noun-phrase handlers.



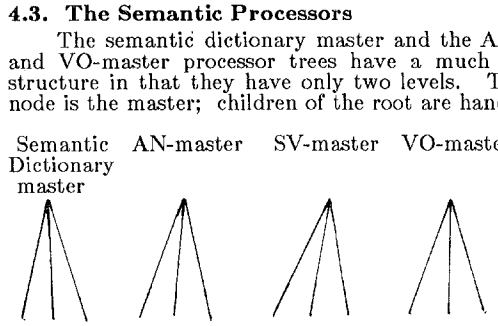
Each noun-phrase handler monitors three child processors. The child processors try to parse the next input phrase as a noun phrase with no adjectives, one adjective, and two or more adjectives respectively. For example, in parsing the phrase "the tough coach," two of the child processors would succeed (no adjectives and one adjective), these results are reported to the parent noun-phrase handler, and then sent to the sentence master via the NP-master. At this point, the waiting sentence processor (child of either one of the SB-handlers or one of the SH-handlers) continues with one of the possibilities and an available sibling processor is allocated by the sentence handler to continue the parse of the sentence using the other possible noun phrase.

In the case of a truly ambiguous sentence, all legal parses are eventually produced. The above example would produce two parses in the case of "The tough coach married people", but not in the case of "The tough coach the young."

Each of the child processes of the noun-phrase handler communicates with the AN-master via the noun-phrase handler.

4.3. The Semantic Processors

The semantic dictionary master and the AN-, SV- and VO-master processor trees have a much simpler structure in that they have only two levels. The root node is the master; children of the root are handlers.



The Semantic dictionary entries are divided among the semantic dictionary handlers. The Semantic dictionary master reads the input and passes the relevant semantic entries, which it obtains from its child processors, to the AN-, SV-, and VO-masters as described in Section 3.

The AN-master receives input which is in general a list of adjectives and a noun, from the noun-phrase handlers. It forms all possible pairs (adjective word sense, noun word sense) and allocates child processors to determine whether or not there is a semantic match. The pairs consisting of a list of adjective word senses, and a noun word sense which matches each of the adjective word senses in the list, are returned to the NP-master.

The SV-master and the VO-master receive input directly from the sentence processors. The input and output of these processes is exactly as described in Section 3. In both cases, the semantically compatible word sense pairs are determined in parallel.

5. Future work

The Computing Research Laboratory (CRL) has the use of Longman's LDOCE English dictionary, which is realistic in size, provides comprehensive syntactic information and also has its semantic entries both syntactically and semantically restricted, and limited to a 2000 word vocabulary. We plan to implement the Semantic Dictionary master by providing each of the semantic dictionary handlers with a portion of LDOCE.

After the initial implementation of the designed parallel parser, we would like to see how Wilksian Preference Semantics [Wilks 75, Wilks et al 85] can be realized in our parser in the sense that one or more readings (in the case of genuine ambiguity) can be selected by weighting the competing interpretations. We are also investigating a parallel parsing model which is driven by semantics, rather than syntax. We have in mind that the role of the sentence master in this case is purely semantic and that syntax is used only to help the segmentation of the input string. Comparison of the two systems would be of great interest to us. Eventually, we also want to consider the incorporation of pragmatics into the system.

6. Acknowledgements

We would like to thank the Natural Language group at the CRL, namely Yorick Wilks, Jerry Ball, Sharon Dorfman, David Farwell, Dan Fass, Chengming Guo, and Sylvia Candelaria de Ram, for their comments and suggestions. We also thank Ted Dunning for his many helpful discussions.

7. References

Charniak, E. (1983) "Passing markers: a theory of contextual influence in language comprehension," *Cognition Science,* 1983, 7, 171-190.

Eiselt, K.P. (1985) "A parallel-process model of online inference processing," *Proceedings of International Joint Conference on Artificial Intelligence,* Los Angles, CA.

Huang, X-M. (1985) "Machine translation in the SDCG formalism," *Proceedings of the Conf. on Theoretical and Methodological Issues in Machine Translation of Natural Languages,* Colgate University, New York.

---

*The system will never invoke the black-sheep processor unless the sentence contains four or more conjoined sentences or a parenthetical sentence with three or more conjuncts in it. In these cases the black-sheep processor is used only for the recursive calls which cannot be handled by the system.

Intel, Corp. (1985) *iPSC Data Sheet,* Intel Scientific Computers, Oregon.
Katz, J. & Fodor, J. (1963) "The structure of a semantic theory," *Language* 39, pp.170-210.
Pereira, F. & Warren, D. (1980) "Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks," *Artificial Intelligence,* 13:231-278.
Waltz, David L. & Pollack, Jordan B. (1985) "Massively parallel parsing: a strongly interactive model of natural language interpretaiton," mimeo.
Wilks, Y.A. (1975) "Preference semantics," Keenan (ed), *Formal Semantics of Natural Language,* Cambridge University Press, London.
Wilks, Y., Huang, X-M. and Fass, D. (1985) "Syntax, preference and right attachment," *Proceedings of IJCAI85,* UCLA, Los Angeles.
Winograd, T. (1972) *Understanding Natural Language,* Academic Press, New York.