From: AAAI-90 Proceedings. Copyright ©1990, AAAI (www.aaai.org). All rights reserved.

Truly Parallel Understanding of Text*

Yeong-Ho Yu and **Robert F. Simmons**

Artificial Intelligence Laboratory
The University of Texas at Austin
Austin, Texas 78712
yu@cs.utexas.edu, simmons@cs.utexas.edu

Abstract

Understanding a text requires two basic tasks: making inferences at several levels of knowledge and composing a global interpretation of the given text from those various types of inferences. Since making inferences at each level demands an extensive computations, there have been several attempts to use parallel inference mechanisms such as parallel marker passing (PMP) to increase the productivity of the inference mechanism. Such a mechanism, when used with many local processors, is capable of making inferences in parallel. However, it often poses a large burden on the task of composing the global interpretation by producing a number of meaningless inferences which should be filtered out. Therefore, the increased productivity of the inference mechanism causes the slow down of the task of forming the global interpretation and makes it the bottleneck of the whole system. Our system, TRUE, effectively solves this problem with the *Constrained Marker Passing* mechanism. The new mechanism not only allows the system to make necessary inferences in parallel, but also provides a way to compose the global interpretation in parallel. Therefore, the system is truly parallel, and does not suffer from any single bottleneck.

Introduction

Understanding a text requires making inferences from several levels of knowledge such as syntax, semantics, and pragmatics (Allen 1987). Since the computation needed for this tends to be huge,there has been a trend toward utilizing parallel inference mechanisms such as *parallel marker passing* (PMP) to reduce the computational burden (Norvig 1987, Charniak 1986, Eiselt 1985, Riesbeck & Martin 1985).

Such a mechanism, when used with many local processors, is capable of making inferences on different parts of a sentence in parallel and finding all possible inferences in a short time. For example, in a sentence like "*John saw a man on a hill*," some local processors may be used to make the necessary inferences between *John* and *saw*, while other local processors are making inferences between *saw* and *a man*.

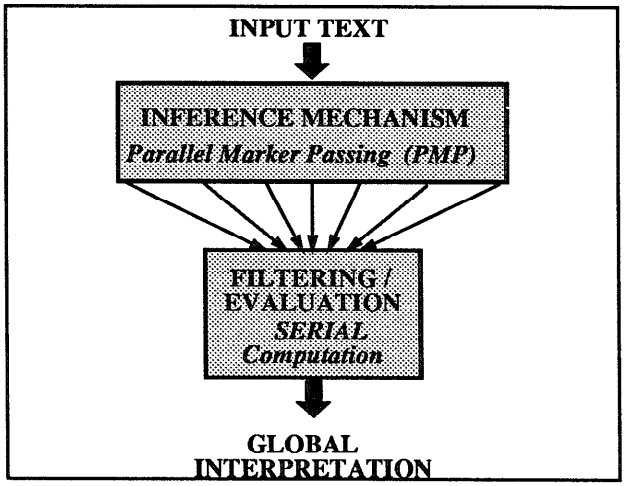

Figure 1: A Bottleneck in PMP

Moreover, the mechanism allows the system to pursue all possible inferences at the same time if there is any ambiguity in the given text. For instance, in the previous sentence, there is a PP-attachment ambiguity of "*on a hill.*" In this case, such a system may make all possible inferences and decide later which one is correct (or better) in the given context.

Since there are many types of possible parallelism — inter-level parallelism, inter-phrasal parallelism, and inter-sentential parallelism— the potential gain from PMP is very large.

However, making the inferences is not the only task in a text understanding system. As a matter fact, it is only a part of it. The other part is to construct a global interpretation of the given text from those inferences. Unfortunately, the parallel inference mechanisms used in most previous works generate a huge number of inferences, most of which are *meaningless*. For instance, (Norvig 1987) had to generate over 230 inferences to understand a three-sentence text, even though only seven of them were meaningful.

*This work is sponsored by the Army Research Office under contract DAAG29-84-K-0060.

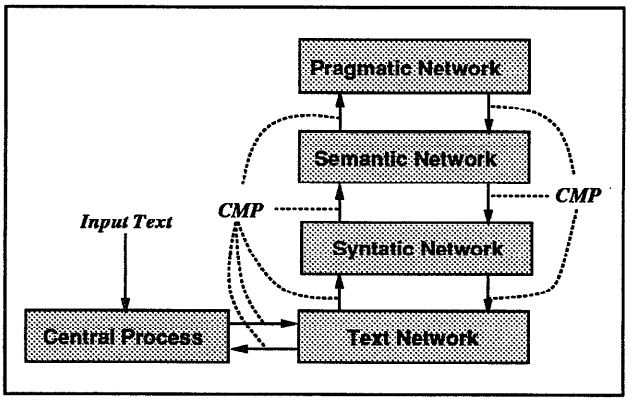

Figure 2: TRUE Architecture

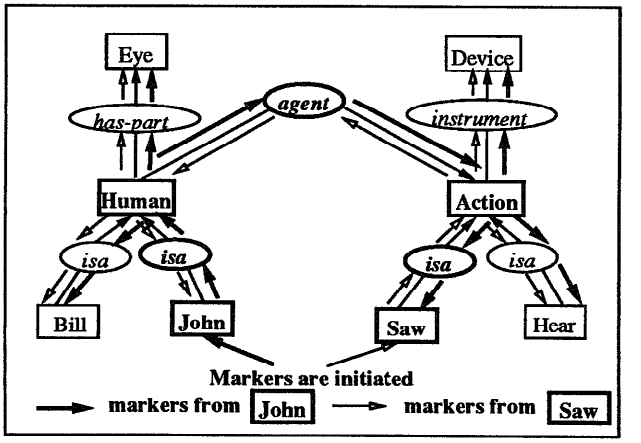

Figure 3: Parallel Marker Passing

Therefore, a *filtering/evaluation step* is needed to remove those meaningless inferences. Also, to find the best interpretation in the presence of ambiguities, the filtering/evaluation step often has to compare several inferences and has to be implemented as a serial process like Figure 1. From the system's viewpoint, the filtering step tends to be the bottleneck of the whole system, and negates the benefits of PMP.

Our system, TRUE[1], solves the problem in two ways. First, it reduces the number of total inferences by pruning meaningless inferences as soon as possible and by utilizing the interactions of three levels of knowledge so that only necessary inferences are made. Secondly, TRUE distributes the filtering step over local processors so that there is no single bottleneck.

The basic mechanism behind all of these is a parallel mechanism called *Constrained Marker Passing*, which is the subject of the next section.

Constrained Marker Passing

Figure 2 shows the architecture of TRUE with its three levels of knowledge: *syntax, semantics,* and *pragmatics.* All three levels are homogeneous in their representation and processing mechanisms: they each use a network of connected nodes[2] to represent their knowledge and all use a parallel mechanism called *Constrained Marker Passing* (CMP) in making inferences.

The result of the whole computation is a *text network* that represents an understanding of the given text in the form of a connected, coherent network of semantic concepts. Initially, the *text network* is empty. When a new sentence is read, the system makes nodes for all possible concepts for each word in the sentence[3], while connecting them to the syntactic category nodes and semantic superclass nodes in the knowledge network as well as to the nodes for the words in the sentence[4]. Managing the text network and creating new nodes for newly-inferenced relations among concepts in the text network are the *only* tasks required of the *central process*; all other computations are localized in the nodes.

Like all the other parallel marker passing mechanisms, CMP finds an inference by detecting the collision of two markers which have traveled through the connected nodes of the knowledge base. However, it is different from other parallel marker passing mechanisms in that it has *programmable* markers. While other PMP mechanisms utilize only one or two types of markers which are passed to all the neighbors indiscriminately and collide with other markers anywhere as in Figure 3, in CMP, there are many types of markers each of which is *constrained* to travel only through pre-defined patterns of paths in levels of the networks and to interact only with pre-specified markers at pre-specified locations as in Figure 4.

The number of markers and the definitions of them are decided by the programmer. For the given task of making some specific types of inferences— inference paths in the network— the programmer decides how such paths may be found and defines markers for them. For instance, in Figure 4, to find a case (or semantic) relation, a programmer defines new types of markers which ascend *isa* hierarchies and collide with each other at a case relation.

To help the programmer to define markers easily and efficiently, TRUE provides *Marker Definition Language* (MDL).

[1]It stands for True Readers Understand Efficiently.

[2]Even though *concept, node,* and *local processor* are three different entities, they are used interchangeably in this paper. The actual distinction is that a *concept* is represented as a *node* in our network representation while a *node* is implemented by a *local processor* in our parallel implementation.

[3]Lexical ambiguities will cause multiple concept nodes for words, and they will be resolved in the same way as other types of ambiguities are resolved.

[4]See Figure 7 for an example of the text network.

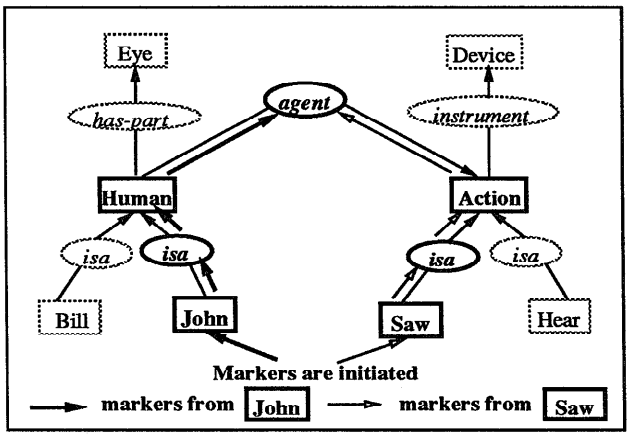

Figure 4: Constrained Marker Passing

Marker Definitions and MDL

A marker definition consists of the four fields: *marker type, authorized paths, special actions,* and *collision specifications.* Marker type represents the identification symbol for the marker while authorized paths specify the pattern of paths which the marker may travel. These paths are described in the form of regular expressions. Special actions are actions to be taken in the middle of the paths. Collision specifications define the details of marker collision such as *with what marker to collide, where, when,* and *what actions to be taken after a collision.*

Currently, MDL is a LISP function and all the values of the fields are given as arguments of the function. Figure 5 shows the definition of the marker used in Figure 4. Note that the patterns are given as regular expressions and may be used to find an infinite number of different paths.

Marker Processing Tables

Even though it is easier for the programmer to envision a marker as an active entity which travels the network for itself, and to define the definition for it, in the actual implementation , a marker is a passive entity passed around by the local processors (nodes). Therefore, it is the responsibility of the nodes to process the markers appropriately.

The instructions of how to handle arriving markers are given as a table called the *Marker Processing Table* (MPT). This table has four fields: *marker type, conditions, actions,* and *out-going markers,* and may have several entries for different types of markers. The algorithm for processing markers at a node is shown in Figure 6. A marker is terminated when there is no MPT entry for its type or when the conditions of the entry fail. Otherwise, the specified actions are taken, and new markers are passed to the appropriate neighbors.

marker type CS(case-search)
A marker for finding a case relation.

authorized path inst·cpt·(isa·cpt)*·case-rel

It travels through an instance-of link to a concept node, then, a chain of isa links and concept nodes to a case relation node.

special actions None.

This one has none. However, other types of markers have actions like getting some values from the nodes in the middle of the path.

collision specifics
with: CS
where: at the end of the path
when: always
actions: send NC to CP
send ARG to INST

It collides only with another CS at the end of a path of the defined pattern under any condition. As a result, a NC marker will be sent to CP and a ARG marker to INST neighbors.

Figure 5: A Marker Definition

All the nodes use the same algorithm, but have different MPT's which are *compiled* from the marker definitions. Therefore, in the example of Figure 4 and Figure 5, the MPT's of all those nodes (i.e., *isa, concept,* and *agent*) of the authorized paths have entries for that type of marker while others don't. This algorithm provides an efficient way for automatic terminations of markers as soon as they deviate from the authorized paths.

Advantages of CMP

The most important advantage of CMP is that, through careful programming, the system only generates the necessary inferences without making meaningless ones. Secondly, since the markers are terminated as early as possible, the marker traffic is very light compared with other mechanisms. Since the overall performance of a parallel system often depends on the communication time among local processors, this light traffic may improve the overall system performance substantially.

Finally, the programmability of markers and MDL provides the programmer with a way to define markers for *fine-grained interactions* among nodes at different levels. As it will be shown in the following sections, through these interactions between levels, TRUE not only generates the necessary inferences, but also finds the best inference in the presence of ambiguities.

Three Levels and Their Interactions

In this section, the summaries of how several kinds of inferences are made in those three levels and how

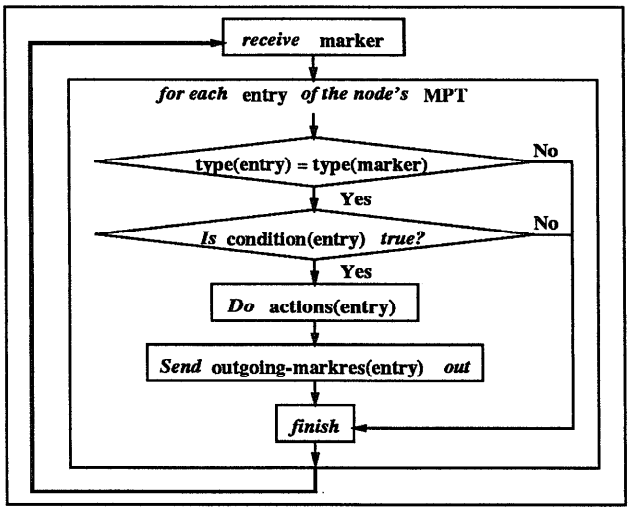

Figure 6: Algorithm for Local Processors

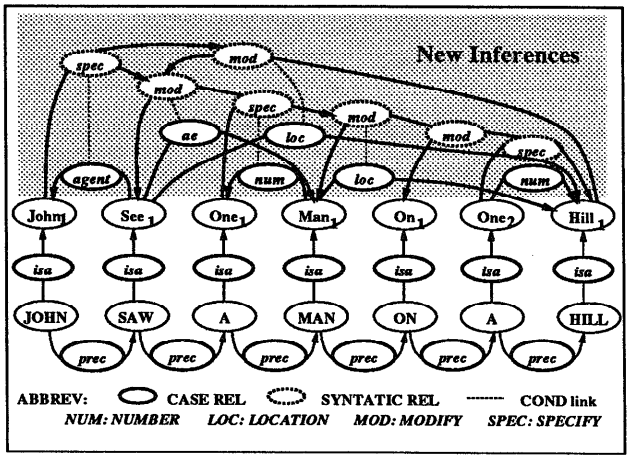

Figure 7: A Sample Text Network

interactions between levels minimize the number of inferences generated are presented. More details can be found in (Yu & Simmons 90).

Each level makes different types of inferences. The syntax level looks for syntactic relations such as *specify* and *modify* which have their origins in $\overline{X}$ grammar. Roughly speaking, they are equivalent to the left and right branches in conventional parse trees. When the initial text network is instantiated with the given sentence, all the concept nodes in the network send out markers for syntactic relations. The markers travel through the syntax network which has the knowledge of English grammar such as "an NP *specifies* a VP" and "a PP *modifies* an NP or a VP." As a result, this network will find all possible syntactic relations in the text.

The semantic level searches for case relations among concepts in the text such as *agent* between *saw* and *John* or *affected-entity*(ae) between *saw* and *a man* in the sentence "John saw a man on a hill". The search for a case relation is initiated when a syntactic relation is found in the syntax level. For instance, in the previous sentence, the new syntax relations between *John – saw, saw – a man, a – hill*, etc. initiate the searches for case relations between those concepts resulting in the case relations, *agent, affected-entity,* and *number* respectively. The rationale behind this *bottom-up* interaction is to prevent the system from finding a syntactically-incorrect case relation such as *affected-entity* between *saw* and *a hill*, and to reduce the amount of computation in the semantic level with the help from the syntax level.

In TRUE, this interaction is stated in the definition of markers for the syntactic relations. That is, the definitions of the markers for syntactic relations have *collision specifications*, a part of which tells the node to fire markers for case relations when a collision between two syntactic markers occurs.

Figure 7 shows the text network after the syntax and semantic levels have made all possible inferences for the sentence "John saw a man on a hill." Note that there are two possible syntactic relations coming out from "on a hill" due to the PP-attachment ambiguity. The dotted lines— a graphical abbreviation for *cond* links– connect the related syntactic and case relations.

The pragmatic level tries to find the coherence relations between event and state concepts (usually verbs)(Alterman 1982) in the given text. For instance, there is a coherence relation(*antecedent*) between a concept *give* and *have— to give something to somebody, one has to have it first.* The knowledge about this coherence relation can be stated as a rule as below:

If *agent*(ownership) = *agent*(ownership-transfer) **and**
affected-entity(ownership)
= *affected-entity*(ownership-transfer)
then ownership is an *antecedent* of ownership-transfer.

This rule requires some argument matchings for the coherence relation to be instantiated. This is to prevent a false coherence relation from being inferred. For instance, it is wrong to infer an *antecedent* relation between **had** and **gave** in the sentences like "John had an old bike. His father gave him a new bike."

In the same way that the syntax level activates the semantic level, the inferences made at the semantic level activate the search for coherence relations at the pragmatic level. When a case relation is instantiated as a node in the text network, it initiates markers which travel through the part of the pragmatic network which represents the argument matching conditions. If all the conditions of a coherence relation are met through the collisions of markers from the arguments, then the coherence relation is instantiated in the text network.

Through this guidance from a low level to a high level, TRUE accomplishes the task of minimizing the

number of inferences without loosing any valid ones.

There is another type of inference which has not been described so far. It is that of finding possible referents for pronouns in the sentence. This is also done through separate types of markers which are activated as soon as the sentence is instantiated in the text network.

Distributed Filtering

There are two reasons for filtering the inferences in other systems: to eliminate the meaningless inferences, and to find the best inference out of several inferences generated from an ambiguity and form a globally coherent interpretation for the given text. In TRUE, there are no meaningless inferences and subsequently no need for such elimination. However, TRUE still generates multiple inferences for ambiguous parts of the text, and has to find the best one among them.

For instance, in a three-sentence text, "John had a bike. Bill wanted it. He gave it to him.", there are ambiguities in the referents for pronouns *he* and *him*. In the beginning, markers for pronoun referents find both *John* and *Bill* as possible candidates for both pronouns.

To resolve this kind of ambiguity, the system has to depend on the inferences made at the higher levels. For instance, the inference of a coherence relation between *have* and *give* makes *John* as the preferable referent for *He* since the knowledge says that the *agents* of *have* and *give* have to be the same to have such a coherence relation.

In reality, there are many kinds of ambiguities and this constitutes the biggest problem in understanding a text. There have been two approaches in finding the best out of multiple inferences: *centralized* and *distributed*.

The centralized approaches like (Charniak 1986, Norvig 1987) gather all inferences at a central process and compare them by applying several heuristics. This approach, even though it is easier to implement, has an apparent disadvantage over the distributed approach: *the central process* may turn out to be the bottleneck of the system.

Therefore, TRUE takes a distributed approach similar to those of (Waltz & Pollack 1985, Howells 1988). In these *local connectionist* systems, each node has a real value called its *activation level* and links between nodes have *weights*. The activation level of a node is computed from the sum of the products of the activation levels of neighbors and the weights of the links between itself and the neighbors. In this scheme, a negative weight between two nodes indicates that they are *mutually exclusive* while a positive weight indicates that they are *mutually inclusive*. After a number of cycles, the activation levels of all nodes may converge to stable values. Then, the nodes with high activation levels are declared as winners.

However, this approach also has a disadvantage. That is, the network should be pre-wired with appropriate weights. In TRUE, the text network grows as a new sentence comes in and as new inferences are made. So, it has no way to decide the proper weights in advance and has to use another method to compute the activation levels.

Instead of re-calculating its activation level periodically, a node in TRUE changes its activation level only when an *activation-change* marker arrives either from a node which is connected to it through a *cond* link, or from its competitor. In the first case, the activation level is increased while, in the second case, it is decreased[5]. Note that a node is connected to another through a *cond* link only when two nodes are at different levels and they are related. For example, a case relation is connected to a syntactic relation through a *cond* only when the search for the case relation was initiated from the syntactic relation.

Therefore, when a marker initiated from a low level succeeds in finding a relation in a higher level, the new node for the high level relation sends out an *activation-change* marker which will increase the activation level of the node for the low level relation. This *top-down* interaction may be interpreted as the principle that, when an inference made at a low level results in an inference at a high level, the inference at the low level is *rewarded* through an increase of its activation level.

When the activation level of a node changes, it sends out *activation-change* markers to its competitors and the nodes which are connected with it through *cond* links so that they may adjust their activation levels.

Figure 8 shows a simplified[6] final text network of the three-sentence text mentioned above. In the figure, those concepts and relations which are involved in the ambiguities are drawn with bold lines, while irrelevant ones are drawn with shaded lines. Two shaded boxes emphasize two ambiguities resulting from two prosnouns — *He* and *him*. Among those ambiguous relations, the winners —those that are included in the global interpretation— are drawn with the white background.

In the beginning, the ambiguity of *He* causes two *referent* relations (*refer-3, refer-4*) to be generated, which in turn enable two *agent* relations (*agent-3, agent-4*) to be generated and connected to them through *cond* relations (dotted lines). When an *antecedent* relation is inferenced between **had** and **gave**, the argument matchings increase the activation level of *agent-3*, which in turn increases that of *refer-3*. Then, they decrease the activation levels of their competitors— *agent-4* and *refer-4* respectively.

Discussion

Parallelizing a part of a system does not necessarily improve the overall system performance. In some cases, it

[5]The amount of change depends on several factors. Refer (Yu & Simmons 90) for more details.

[6]For the sake of readability, the syntactic relations, *isa* relations, and some other relations are not shown.

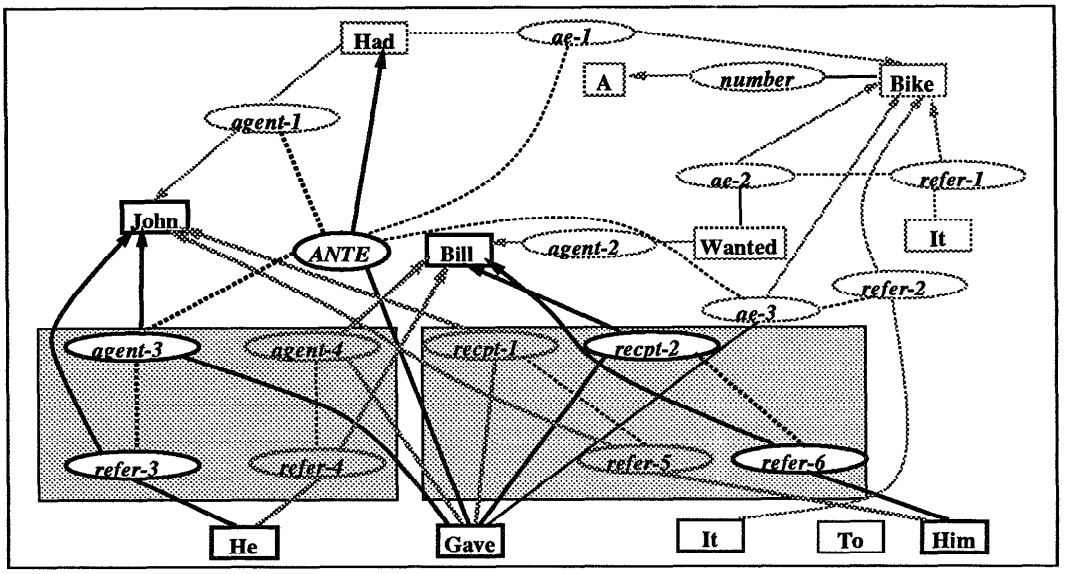

Figure 8: A Simplified Final Text Network

may worsen the system performance by putting a huge burden on a single bottleneck. Unfortunately, this is the case for most systems based on parallel marker passing. Their major bottleneck has been the filtering step which slows down the system greatly due to the huge number of meaningless or ambiguous inferences.

TRUE, on the other hand, does not generate any meaningless inferences and finds the best inference among ambiguous ones efficiently. This is accomplished through Constrained Marker Passing, the bi-directional interactions between three levels of knowledge, and the distributed filtering step.

There are other systems which share some common characteristics with TRUE. For instance, (Sumida, Dyer, & Flower 1989) uses another type of constrained marker passing which is similar to CMP. However, CMP is more general and powerful and provides a programmer with MDL and its compiler so that the programmer may define new types of markers freely without affecting existing ones.

(Norvig 1987, Eiselt 1985) use the technique of pruning meaningless inferences paths. But, the pruning is done at a centralized process and tends to be the bottleneck of the systems.

Currently, TRUE is implemented in a Symbolics LISP machine on top of a simulator for CMP. All the knowledge base and marker definitions are written in a format easy for a programmer and compiled into a knowledge network and a set of MPT's. Currently, there are about thirty types of markers in TRUE. The implementation of TRUE in the Connection Machine is under way.

References

Allen, J. 1987. *Natural Language Understanding,* Benjamin/ Bummings Publishing Co.

Alterman R.E.. A System of Seven Coherence Relations for Hierarchically Organizing event concepts in text. Tech Report, TR-209, AI Lab, Univ of Texas.

Charniak, E. 1986. A Neat Theory of Marker Passing. In Proceedings of the Fifth National Conference on Artificial Intelligence. Menlo Park, Calif.: AAAI.

Eiselt, K.P. 1985. A Parallel-Process Model of On-Line Inference Processing. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence. Menlo Park, Calif.: IJCAI, Inc.

Howells, T. 1988. Vital: A Connectionist Parser. In Proceedings of the Tenth Annual Conference of the Cognitive Science Society

P. Norvig. 1987. Unified Theory of Inference for Text Understanding. Ph.D. diss., EECS, UC Berkeley.

Riesbeck, C.K., and Martin, C.E. 1985. Direct Memory Access Parsing, Technical Report YALEU-DCS-RR 354, Dept of Computer Science, Yale University.

Sumida, R.A.; Dyer, M.G.; and Flowers, M. 1988. Integrating Marker Passing and Connectionism for Handling Conceptual and Structural Ambiguities. In Proceedings of the Tenth Annual Conference of the Cognitive Science Society.

Waltz, D.L., and Pollack, J.B. 1985. Massively Parallel Processing: A Strongly Interactive Model of Natural Language Interpretation. *Cognitive Science*, 9(1).

Yu, Y., and Simmons, R.F. 1990. Bi-Directional Interactions in a Parallel Text Understanding System, Tech Report, AI90-131, AI Lab, University of Texas.