CONstrained MARKER PASSING*

YEONG-HO YU
ROBERT F. SIMMONS

APRIL 1988       AI88-76

*Support for this research was provided by the Army Research Office under grant number ARO DAAG29-84-K-0060.
Constrained Marker Passing

Yeong-Ho Yu
Robert F. Simmons

Artificial Intelligence Laboratory
The University of Texas at Austin
Austin, Texas 78712

25 April 1988

ABSTRACT

Marker Passing is an inference mechanism used in semantic network representations to find all paths between any pair of network nodes. Theoretically, each node in the network can be assumed to be a simple processor that activates each node/processor to which it connects, and the procedure can be seen to be accomplished as a parallel search that succeeds each time the initial pair of nodes are connected by some path of activations. Unfortunately, most such paths are irrelevant and meaningless, so a filtering process is required to select those paths that make relevant and sensible inferential connections. Although previous research in the area has appreciated the potential value of parallel marker-passing, the filtering has usually been a serial process, and the resulting system if implemented in a parallel machine would find filtering a bottleneck that would sharply limit the parallel processing gains. This paper presents a method for accomplishing the filtering as an integral aspect of the parallel marker passing thus removing the expensive bottleneck of serial filtering.

Support for this research was provided by the Army Research Office, under grant number ARO DAAG29-84-K-0060.
1. Introduction

A way to look at a Natural Language Processing (NLP) system is to check out how it makes inferences - that is, relations between entities in a given text and several kinds of knowledge in the system. Currently, various inference mechanisms are used in NLP, while new ones are being developed. However, the decisions of how to make inferences and how to represent knowledge bases (KB’s) are closely related, and a designer of a NLP system has to think about them at the same time. Although there are many possible combinations of representation schemes and inference mechanisms, that of the Semantic Network representation [Simmons 84] and Marker Passing is the one which has got much intention from several researchers recently [Norvig 87] [Eiselt 87] [Charniak 83, 86] [Hendler 88].

The major advantage of marker passing is its genuine parallelism which may allow us to implement it in a parallel hardware. Since a general trend of NLP systems is to incorporate more and more knowledge, generating inferences is getting to be crucial in the performances of the systems and a substantial speed up\(^1\) may decide if a NLP system is computationally viable or not.

Also, the introduction of parallelism may solve a dilemma which most knowledge-based AI systems implemented on serial machines suffer from: adding a new piece of knowledge would degrade the performance of the system unlike the cases of human beings, unless a sophisticated indexing scheme [Schank 82] is used. In parallel computations, it is possible to assign more processors to the new piece of knowledge and prevent the whole system from slowing down.

Above all, the most important merit of applying parallelism to NLP problems may be that it gives us a new perspective to look at the problems. That is, we may be able to develop several parallel algorithms which may hardly be conceived if we think about the problems in terms of serial processing.

Another advantage of marker passing is its compatibility with the semantic network representation, which has been widely used in NLP since its introduction. The mechanism of marker passing assumes a network of processors which are assigned to concepts if a semantic network is implemented on it. And, the other advantage is the simplicity of its local processing job, which allows many small-scale processors with a small memory to be able to do the job.

When the basic idea of the marker passing mechanism was introduced in NLP two decades ago by Quillian [Quillian 68] without the concept of parallelism, it did not attract much attention from researchers in NLP at that time. However, recently there has been a revived enthusiasm of it mainly due to the developments of hardware architectures [Fahlman 79][Hillis 85] which make the mechanism as a realizable one in parallel machines instead of a simulated one in serial machines. Most researches, which use the marker passing mechanism, assume parallel algorithms, although they are simulated in serial machines.

Since they use the mechanism as a passive inference machine in which inferences are generated for the nodes which are activated by another process and handed over back to the

\(^1\)The maximum speed up by using \(N\) processors is \(N\). If \(N\) is very large, for example 64000, then the speed up may be very substantial.
activating process for some filtering-out processing (Figure 1), it is possible to lose the gain from the parallel implementation of marker passing because of the serial processing in the other part.

Also, since most of those researches do not address any requirements on parallel hardwares, they leave a question of whether those simulated mechanisms can be implemented on parallel hardwares and show the desired performances.

This research points out common hidden problems in parallel marker passing mechanisms in those NLP systems, and suggests a solution. It also specifies requirements of a parallel hardware which may support the solution\textsuperscript{2}.

2. Problems in Marker Passing Mechanisms

In the paradigm of marker passing, a knowledge base is represented as a network in which a node represents a concept while a link represents a relation between two concepts\textsuperscript{3}. The mechanism used in most researches on marker passing is so called Generation-and-Evaluation method of which steps are described below and in Figure 2.

1. **Path Generation**: To find a relation between two concepts, the system casts markers from the nodes to all the neighbor nodes which pass the markers to their neighbor nodes again until the markers collide at some nodes. Currently, most researches assume implementing this step in a parallel process.

2. **Path Evaluation**: The path, which is the conjunction of two paths which the markers travel through until they collide, will be evaluated to find out what it means. Currently, most researches do this step in a serial process.

\textsuperscript{2}This does not mean that there is such a hardware available nor that this research aims at the design of such a hardware.

\textsuperscript{3}From now on, we use concept and node, and relation and link interchangeably.
Figure 2 Marker Passing

Even though this is a very simple mechanism, it guarantees that it will find a path between two concepts, if any, and if there are enough processors to implement the whole network, the time required to find a path between two concepts is proportional to the length of the path. However, two facts below make this mechanism impractical in a real NLP system.

- A NLP system usually does not know which pairs of concepts in a text should be related. In a real system, we do not want to try to find relations among all possible pairs of concepts in a text. This partially accounts for the first and the third problems below.

- Markers are passed to all the neighbors instead of some selected ones. Therefore, there will be lots of marker movements and collisions. This partially accounts for the first problem.

Therefore, in the systems where the marker passing process is controlled by another process, this process has to initiate markers from all the concepts in the input to find out all the possible inferences. This simple-minded approach is used in those researches mentioned above, and generates the following problems.

2.1. Problem 1: Meaningless Paths

As described earlier, each path generated from a collision must be evaluated to become a legitimate inference. If it is not (Figure 3), it will be dismissed. One of the drawbacks of the generation-and-evaluation method is that so many meaningless paths are generated due to the unguided collisions and propagations of markers in all directions. For instance, in an example of [Norvig 87], there are over 230 paths generated while only 7 of them are meaningful. This is a typical statistic and causes a serious problem in a real implementation because usually a serial process is used to filter out all the meaningless paths. Therefore, the serial process will be the bottleneck of the whole system and the gain by the parallel processing of path generation will be lost by their serial evaluation.
It may be suggested then that each local processor should filter out paths generated in it by itself instead of handing them over to the serial process. However, this will increase the requirements of processing power of each local processor, and also the problem stated below prevents each path from being evaluated by itself.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figures/figure4.png}
\caption{Competing Paths}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figures/figure5.png}
\caption{An Unwanted Path}
\end{figure}

\subsection{2.2. Problem 2: Competing Paths}

Although it is guaranteed that an inference path will be found, if any, through marker passing, it is not guaranteed that only one will be found. Actually, all the possible inference paths which are competing with each other between two concepts will be found (Figure 4). Thus, another task of the path evaluation is to find out which of multiple paths between two concepts is the right one (or the best one). Therefore, several paths should be compared rather than evaluated one by one. In current systems, finding the best path is mainly based on heuristics such as \textit{shortest path preference} and \textit{explain-more preference} which do not guarantee either the right answer or only one answer all the time.

\subsection{2.3. Problem 3: Unwanted Paths}

In Figure 5, there is a connection(\textit{ae - affected entity}) between two concepts("saw" and "hill") which is meaningful but not desired. This problem comes from the fact that, before processing the input, the system does not know which pairs of concepts it is seeking to relate. This problem is solved by other systems while resolving competing paths. That is, they dismiss the relation between "saw" and "hill" in favor of the one between "saw" and "man".
2.4. Problem 4: Ambiguities

In general, the major task of NLP is to resolve various kinds of ambiguities such as word sense ambiguities, syntactic ambiguities, specific vs. general rules, and anaphora ambiguities. Any of those ambiguities will generate multiple competing paths in the marker passing paradigm. Because some of the ambiguities in an input may be hard or unsolvable ones in a given context even to a human reader, it is very risky to depend on heuristics to find the best one as most systems do.

In this paper, we do not address all these problems, but only a part of them. The mechanism discussed in this paper is a part of a bigger research which is trying to address all the problems above [Yu 88]. The main concern of this paper is to prevent unnecessary paths from being generated at all instead of generating them and discarding them later. The next section discusses the proposed mechanism in detail.

3. Constraining Marker Passing Activities

When we talk about meaningful paths, we presuppose that we know which is meaningful and which is not. And, in any NLP system, there are certain kinds of inferences the system is looking for. Therefore, it is possible to define the inference paths⁴. For instance, [Norvig 87] defines the patterns of inference paths, his system looks for, in regular expression forms and uses them in the path evaluation step to dismiss meaningless paths.

The existence of evaluation processes in the other NLP systems with parallel marker passing implies that, in those systems, meaningful paths are defined in some way. However, those definitions of meaningful paths are used in the evaluation processes separated from the marker passing process (Figure 1).

In our system, we shift the computation of the evaluation process into the marker passing mechanism (Figure 6). That is, we define the patterns of inference paths by restricting the movements of markers to generate only well-patterned paths.

Each type of inference is given a separate type of marker which is allowed to travel through only some pattern(s) of that inference. This makes it possible to generate only allowed types of inferences⁵. But, there are several other things to consider such as how to record partial paths travelled through so far, and with which markers to collide. Therefore, a definition of a marker is more sophisticated and is shown in Figure 7.

Then, the definition of a marker is compiled into related nodes in a network such that the paths it may take are recorded in the nodes which may be a part of the paths. That is, a node has a list of marker types it can accept (in the case that the node is the end of a path) or pass (in the case that the node is in the middle of a path). If any marker, not in the list, reaches the node, the marker will be terminated. In the case of accepting a marker, the node checks if the marker may

---

⁴It does not mean that defined inferences are all the possible inferences any NLP system can make, but they are certainly the ones a NLP system aims at.

⁵This does not mean that all the generated paths will survive in the final output after all, but the number of paths which will be generated will be reduced greatly.
Definition of a Marker

Path: regular expression(s).
Collision Conditions: types of markers, conditions.
   Actions: actions to be taken when a collision occurs.

Data Structure of a Marker

Type: shows what inference(s) it is assigned to.
ID-#: represents its own identity.
Origin: the origin it is initiated from.
State Variable: the value of a partial path it has taken so far.
Destination (optional): the destination it is heading to.
Variable(s) (optional): readable and writable variable(s) for general purposes.

Figure 7. Definition and Data Structure of a Marker

collide with any markers which have arrived so far. If so, the node will take actions specified in the definition. The reactions, which a node may take when a marker arrives, are depicted in Figure 8.

While an implementation of a node is suggested in the next section, the data structure of a marker is given in Figure 7. An important design detail here is that, instead of recording an
actual partial path a marker has travelled so far, it changes the value of State Variable. That is, a numeric value is assigned to a partial path of an inference.

Figure 8. Reactions of a Node

![Diagram of reactions of a node]

Inference Path:   (inst)* (isa)* cat  
State Variable:    0 1 2 3

Figure 9. State Variable and Partial Paths

For example, in Figure 9, an inference path is defined by a regular expression in which there are four distinct states: the initial state, the state after passing some number of inst relations, the state after passing some number of isa relations, and the final state. Those states are represented with numbers 0 through 3 and kept in the state variable. Then, by checking the number, a processor can know what kind of path a marker has travelled through without explicitly recording it.

4. Hardware Requirements and Implementation

There are several parallel processors available in the market, but our research does not want to assume any specific hardware. Instead, we want to describe the requirements of an ideal hardware for our marker passing paradigm. Since we have not found any hardware which fulfills all these requirements, our research is also based on simulations of the ideal machine. Nevertheless, our simulator is carefully designed to fulfill all the hardware requirements to make sure that we can easily implement it whenever the ideal hardware is available. The requirements are as below.

- A large number of small scale processors with moderate size of memory, each of
which can execute its own instructions on its own data - Fine-grained Multiple Instructions Multiple Data machine.

- A central processor which can access all the local processors.
- A network which is flexible enough that a connection between two local processors may be created and/or destroyed dynamically\(^6\).

Even though the simulation of a hardware with these requirements is possible, the ultimate goal of our research is to build up a NLP system in a parallel machine. Therefore, as a possible target machine we picked an existing system - the Connection Machine- which is close to our ideal machine, and designed a possible implementation of our marker passing mechanism on it.

The machine is different from our ideal machine in the fact that it is a SIMD (Single Instruction Multiple Data) machine in which the central processor executes an instruction which effects all the local processors at the same time. Therefore, in this proposed implementation, the instructions, which should be taken by a node when it receives a marker, are normalized to a simple operation - looking up a table. Otherwise, the system has to slice the time of the central processor to execute the instructions of each local processor. Instead of its own instructions, a node simply has a table of entries of markers which it can accept or pass as in Figure 10. When a marker arrives at a node, the node checks the table to see if it is in the table, then checks the conditions. If every condition is fulfilled, it will do the actions specified. Therefore, the time needed to transmit and process a marker is the sum of the time needed to transmit the longest marker and the time needed to process a marker which happens to take the longest time.

<table>
<thead>
<tr>
<th>in-marker</th>
<th>condition</th>
<th>action</th>
<th>out-marker</th>
</tr>
</thead>
<tbody>
<tr>
<td>types of markers</td>
<td>conditions to be checked</td>
<td>actions to be taken</td>
<td>markers to be sent, if any</td>
</tr>
<tr>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
</tbody>
</table>

Figure 10. Table for a Node

5. An Example

Because even a simple example of marker passing takes a lot of space, only a part of an example will be shown here. The details of it may be found in [Yu 88] which proposes a three-level system. All the levels are designed with the constrained marker passing mechanisms. In this paper, we show an example at the first level - syntax level. The sentence to be processed is "John had a bike".

\(^6\)The reason for the dynamic linkage is not clear in this paper, but is explained in [Yu 88].
\[
\begin{align*}
\n\rightarrow & \quad \rightarrow N N \\
\n\rightarrow & \quad V (N) (P)^* \\
N & \quad \rightarrow (\text{det}) N \\
N & \quad \rightarrow N (P)^* \\
\text{P} & \quad \rightarrow \text{P} \\
P & \quad \rightarrow \text{prep} N \\
\end{align*}
\]

(a) A Grammar

(b) A Network Representation

Figure 11. Grammar for Example

In the syntax level, we use X-bar grammar, which has been shown to be general through several researches in linguistics. Figure 11(a) shows the grammar used in this example while Figure 11(b) is the network representation of the grammar\(^7\). In X-bar grammars, there are two possible syntactic relations between two constituents: \textit{specify} (or \textit{pre-modify}), and \textit{modify} (or \textit{post-modify}). Therefore, the task of the first level is to find out all the possible syntactic relations of these kinds.

At the start, the central processor creates nodes for words in a sentence and connects them to corresponding lexicon nodes in the network. Also, in order to represent the ordering of the words, it creates a \textit{precede} relation between consecutive words. The result is Figure 12(a).

For each syntactic relation - that is, \textit{specify} or \textit{modify} - there are two types of markers which are initiated from new nodes of words: \textit{proposer} and \textit{acceptor}. In this example, we show those of \textit{specify} of which definitions are given in Figure 13. The markers of \textit{modify} are very similar to those of \textit{specify} and not given in this paper. Using the definitions, we will show how a \textit{specify} relation between \textit{John} and \textit{had} is generated in our paradigm.

Figure 12(b) shows how markers from those two words collide at a node. Then, as specified in the definitions of markers, two markers of CR(Collision Report) type are generated and passed as in Figure 12(c). After those markers reach destinations, the nodes ask the central processor to create a new node between them. Then, the final result would be Figure 12(d).

This example only shows a flavor of our developing system based on the constrained marker passing. The details of it may be found in [Yu 88]. One final comment in this example is that, a \textit{proposer} marker or \textit{acceptor} marker may be passed to several neighbors at any point,

\(^7\)The network representation is compiled automatically from the rule representation by a program.
Figure 12. Syntax Processing of a Sentence
Proposer for Specify

*Path*: (*modify)* precede (specify)*

(Collision Condition: \(\text{Origin(Proposer)} = \text{Origin(Acceptor)}\))

Action: Send a CR-marker with \(\text{Destination(CR)} = \text{Origin(Proposer)}\)

Acceptor for Specify

*Path*: (inst)* (isa)* specify (*isa)* (*inst)*

(Collision Condition: \(\text{Origin(Acceptor)} = \text{Origin(Proposer)}\))

Action: Send a CR-marker with \(\text{Destination(RC)} = \text{Origin(Proposer)}\)

CR (Collision Report) for Specify

*Path*: (*specify)* *precede (modify)*

(Collision: None)

Figure 13. Definitions of Markers

but will be terminated as soon as it reaches a node which is not a part of the defined inference paths. For example, an acceptor may be passed from N to Mod in Figure 12(b), but will be terminated because it is not included in the definition of the marker. In the figures, we show only the relevant movement of markers.

6. Summary

The paradigm of marker passing is revived with the introduction of new parallel hardware. The parallel implementation of it may greatly speed up the inference process which is crucial in knowledge-based NLP systems. However, most researches based on this paradigm use the generation-and-evaluation technique which puts much burden on the process of filtering out meaningless paths. Since the filtering-out process is done in a serial process in the other researches, the gain of parallel path generation will be lost. Also, they don't specify the hardware requirements for their parallel implementations and, therefore, implementations of them on parallel hardwares may introduce several unforeseen problems.

In this research, we use so called the constrained marker passing mechanism in which markers are defined with possible inference paths and passed only through those allowed paths. This distributes the task of path evaluation over the local processors instead of to a single processor, and removes a possible bottleneck of the whole system. In other words, we shift the computation of filtering out inference paths from a serial process into the parallel marker passing mechanism.

This research is a part of a more comprehensive study in which other features are mixed with this mechanism to promote the local processing as much as possible while the job of a
central processor is kept minimal. Still, we are simulating the mechanism even though it is based on the requirements of an ideal machine for our mechanism. We also study a possible implementation of this mechanism in an existing parallel hardware.

References


