

High Performance Natural Language Processing on Semantic Network Array Processor*

Hiroaki Kitano,
Center for Machine Translation
Carnegie Mellon University
Pittsburgh, PA 15213 U.S.A.
roaki@cs.cmu.eduhiroaki.

Dan Moldovan, and Seungho Cha
Parallel Knowledge Processing Laboratory
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089-1115

Abstract

This paper describes a natural language processing system developed for the Semantic Network Array Processor (SNAP). The goal of our work is to develop a scalable and high-performance natural language processing system which utilizes the high degree of parallelism provided by the SNAP machine. We have implemented an experimental machine translation system as a central part of a real-time speech-to-speech dialogue translation system. It is a SNAP version of the Φ DMDIALOG speech-to-speech translation system. Memory-based natural language processing and syntactic constraint network model has been incorporated using parallel marker-passing which is directly supported from hardware level. Experimental results demonstrate that the parsing of a sentence is done in the order of milliseconds.

1 Introduction

In this paper, we will demonstrate that natural language processing speeds in the order of milliseconds is attainable by using a marker-propagation algorithm and a specialized parallel hardware.

The significance of the high-performance (or real-time) natural language processing is well known. Parsing sentences at the milliseconds speeds enables the realization of a speech recognition module capable of real-time speech understanding which eventually leads to the real-time simultaneous interpretation system. Also, the millisecond order performance enables the system to parse hundreds of sentences in a second, or over 3 million sentences per hour. This in turn makes possible bulk processing of text such as full-text retrieval, summarization, classification, translation, indexing and tagging.

In order to accomplish the high-performance natural language processing, we have designed a highly parallel machine called Semantic Network Array Processor (SNAP) [Moldovan and Lee, 1990] [Lee and Moldovan, 1990], and implemented an experimental machine translation system called DMSNAP using a parallel marker-passing scheme. DM-SNAP is a SNAP implementation of the Φ DMDIALOG speech-

to-speech dialogue translation system [Kitano, 1990a] [Kitano, 1991a], but with some modifications to meet hardware constraints. Despite its high performance, our system carries out sound syntactic and semantic analysis including lexical ambiguity, structural ambiguity, pronoun reference, control, unbounded dependency, and others.

In the next section, we describe briefly the SNAP architecture, then, describe design philosophy behind the DMSNAP followed by descriptions on implementation and linguistic processing. Finally, performance are presented.

2 SNAP Architecture

The Semantic Network Array Processor (SNAP) is a highly parallel array processor fully optimized for semantic network processing with marker-passing mechanism. In order to facilitate efficient propagation of markers and to ease development of applications, a set of marker propagation instructions has been microcoded. SNAP supports propagation of markers containing (1) bit-vectors, (2) address, and (3) numeric value. By limiting content of markers, significant reduction in cost and resource has been attained without undermining performance requirements for knowledge processing. Several AI applications such as natural language processing system, classification system [Kim and Moldovan, 1990], and rule-based system has been developed on SNAP.

The Architecture

SNAP consists of a processor array and an array controller (figure 1). The processor array has processing cells which contain the nodes and links of a semantic network. The SNAP array consists of 160 processing elements each of which consists of TMS320C30 DSP chip, local SRAM, etc. Each processing element stores 1024 nodes which act as virtual processors. They are interconnected via a modified hypercube network. The SNAP controller interfaces the SNAP array with a SUN 3/280 host and broadcasts instructions to control the operation of the array. The instructions for the array are distributed through a global bus by the controller. Propagation of markers and the execution of other instructions can be processed simultaneously.

Instruction Sets

A set of 30 high-level instructions specific to semantic network processing are implemented directly in hardware. These include associative search, marker setting and propagation, logical/arithmetic operations involving markers, create and

*This research has been funded by the National Science Foundation Grant No. MIP-9009109 and MIP-9009111.

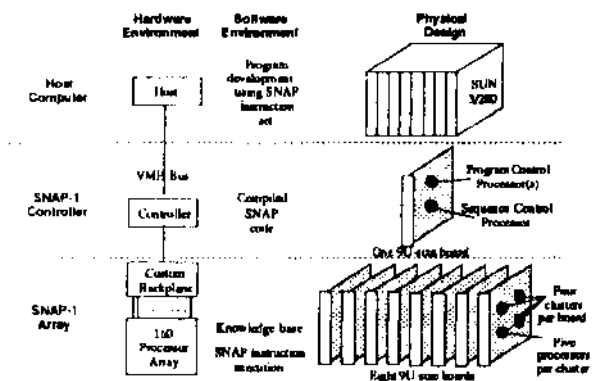


Figure 1: SNAP Architecture

delete nodes and relations, and collect a list of nodes with a certain marker set. Currently, the instruction set can be called from C language so that users can develop applications with an extended version of C language. From the programming level, SNAP provides data-parallel programming environment similar to C* of the Connection Machine [Thinking Machine Corp., 1989], but specialized for semantic network processing with marker passing.

Propagation Rules

Several marker propagation rules are provided to govern the movement of markers. Marker propagation rules enable us to implement guided, or constraint, marker passing as well as unguided marker passing. This is done by specifying type of links that markers can propagate. All markers in DMSNAP are guided markers, thus they are controlled by propagation rules. The following are some of the propagation rules of SNAP:

- *Seq*(r_1, r_2) : The *Seq* (sequence) propagation rule allows the marker to propagate through r_1 once then to r_2 .
- *Spread*(r_1, r_2) : The *Spread* propagation rule allows the marker to travel through a chain of r_1 links and then r_2 links.
- *Comb*(r_1, r_2) : The *Comb* (combine) propagation rule allows the marker to propagate to all r_1 and r_2 links without limitation.

Knowledge Representation on SNAP

SNAP provides four knowledge representation elements: *node*, *link*, *node color* and *link value*. These elements offer a wide range of knowledge representation schemes to be mapped on SNAP. On SNAP, a concept is represented by a node. A relation can be represented by either a node called relation node or a link between two nodes. The node color indicates the type of node. For example, when representing USC is in Los Angeles and CMU is in Pittsburgh, we may assign a relation node for in. The IN node is shared by the two facts. In order to prevent the wrong interpretations such as USC in Pittsburgh and CMU in Los Angeles, we assign IN#1 and IN#2 to two distinct IN relations, and group the two relation nodes by a node color IN. Each link has assigned to it a link value which indicates the strength of interconcepts relations. This link value supports probabilistic reasoning and connectionist-like

processing. These four basic elements allow SNAP to support virtually any kinds of graph-based knowledge representation formalisms such as KL-ONE [Brachman and Schmolze, 1985], Conceptual Graphs [Sowa, 1984], KOD1AK [Wilcnsky, 1987], etc.

3 Philosophy Behind DMSNAP

DMSNAP is a SNAP implementation of the Φ DMDIALOG speech-to-speech dialogue translation system. Naturally, it inherits basic ideas and mechanisms of the Φ DMDIALOG system such as memory-based approach to natural language processing and parallel marker-passing. Syntactic constraint network is introduced in DMSNAP whereas Φ DMDIALOG has been assuming unification operation to handle linguistic processing.

Memory-Based Natural Language Processing

Memory-based NLP is an idea of viewing NLP as a memory activity. For example, parsing is considered as a memory-search process which identifies similar cases in the past from the memory, and to provide interpretation based on the identified case. It can be considered as an application of Memory-Based Reasoning (MBR) [Stanfill and Waltz, 1986] and Case-Based Reasoning (CBR) [Riesbeck and Schank, 1989] to NLP. This view, however, counters to traditional idea to view NLP as an extensive rule application process to build up meaning representation. Some models have been proposed in this direction, such as Direct Memory Access Parsing (DMAP) [Riesbeck and Martin, 1985] and Φ DMDIALOG [Kitano, 1990a]. We consider that memory-based approach is superior than traditional approach. For arguments concerning superiority of the memory-based approach over the traditional approach, see [Nagao, 1984], [Riesbeck and Martin, 1985], [Kitano, 1990a], [Sumita and Iida, 1991], [Kitano and Higuchi, 1991a], and [Kitano and Higuchi, 1991b].

Parallel Marker-Passing

One other feature inherited from the Φ DMDIALOG is use of parallel marker-passing. In DMSNAP, however, a different approach has been taken with regard to the content of markers propagate through the network. Since Φ DMDIALOG has been designed and implemented on idealized simulation of massively parallel machines, markers carry feature structure (or graph) along with other information such as probabilistic measures, and unification or a similar heavy symbolic operations has been assumed at each processor element (PE). In the DMSNAP, content of the marker is restricted to (1) bit markers, (2) address markers, and (3) values¹. Propagation of feature structures and heavy symbolic operations at each PE, as seen in the original version of the Φ DMDIALOG, are not practical assumptions to make, at least, on current massively parallel machines due to processor power, memory capacity on each PE, and the communication bottleneck. Propagation of feature structures would impose serious hardware design problems since size of the message is unbounded

¹ We call a type of marker-passing which propagates feature structures (or graphs) an *Unbounded Message Passing*. A type of marker-passing which passes fix-length packets as seen in DMSNAP is a *Finite Message Passing*. This classification is derived from [Bliech, 1986]. With the classification in [Bliech, 1986], our model is close to the Activity Flow Network.

(unbounded message passing). Also, PEs capable of performing unification would be large in physical size which causes assembly problems when thousands of processors are to be assembled into one machine. Even with machines which overcome these problems, applications with a restricted message passing model would run much faster than applications with an unbounded message passing model. Thus, in DMSNAP, information propagated is restricted to bit markers, address markers, and values. These are readily supported by SNAP from hardware level.

Syntactic Constraint Network

Syntactic constraint network (SCN) is a new feature which has not been used in the previous works in memory-based NLP. SCN is used to handle syntactic phenomena without undermining benefits of memory-based approach. Although, unification has been the central operation in the recent syntactic theories such as LFG [Kaplan and Bresnan, 1982] and HPSG [Pollard and Sag, 1987], we prefer SCN over unification-based approach because unification is computationally expensive and it is not suitable for massively parallel implementation. Although there is a report on an unification algorithm on massively parallel machines [Kitano, 1991b], still it is computationally expensive, and takes up major part of computing time even on SNAP. In addition, there is a report that unification is not necessary the correct mechanism of enforcing agreement [Ingria, 1990]. Also, the classification-based approach [Kasper, 1989], which pre-compiles a hierarchy of feature structures in the form of a semantic network, can carry out similar task with less computational cost [Kim and Moldovan, 1990]. Finally, current unification hard-rejects failure which is not desirable from our point. We want the system to be robust enough that while recognizing minor syntactic violation, it keep processing to get meaning of the sentence.

in the syntactic constraint network model, all syntactic constraints are represented in the finite-state network consists of (!) nodes representing specific syntactic constraints (such as 3SGPRES), (2) nodes representing grammatical functions (such as SUBJ, OBJ, and OBJ2 for functional controller), and (3) syntactic constraint links which control state-transitions and the passing of information among them. Although, unification has been used to (1) enforce formal agreement, (2) percolate features, and (3) building up feature structure, we argue that these functions are attained by independent mechanism in our model. Formal agreement is enforced by activation and inhibition of nodes through active syntactic constraints. Percolation of feature is attained by passing of address through memory and syntactic constraint networks. It should be noted that not all features now being carried by unification grammar need to be carried around in order to make an interpretation of sentences. Our model only propagates necessary information to relevant nodes. Finally, instead of building up features, we distributively represent meaning of the sentence. When parsing is complete, we have a set of new nodes where each represents an instance of concept and links defines relation among them.

We are currently investigating whether our model is consistent with human language processing which has limited memory capacity [Gibson, 1990].

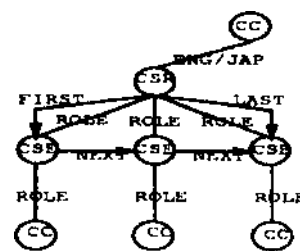


Figure 2: Concept Sequence on SNAP

4 Implementation of DMSNAP

DMSNAP consists of the memory network, syntactic constraint network, and markers to carry out inference. The memory network and the syntactic constraint network are compiled from a set of grammar rules written for DMSNAP. This section describes these components and a basic parsing algorithm to provide brief implementation aspects of the DMSNAP.

4.1 Memory Network on SNAP

The major types of knowledge required for language unslation in DMSNAP are: a lexicon, a concept type hierarchy, concept sequences, and syntactic constraints. Among them, the syntactic constraints are represented in the syntactic constraint network, and the rest of the knowledge is represented in the memory network. The memory network consists of various types of nodes such as concept sequence class (CSC), lexical item nodes (LEX), concept nodes (CC) and others. Nodes are connected by a number of different links such as concept abstraction links (ISA), expression links for both source language and target language (ENG and JPN), Role links (ROLE), constraint links (CONSTRAINT), contextual links (CONTEXT) and others.

A CSC captures ordering constraints of natural language, and it roughly corresponds to phrase structure rules. CSCs can be used to represent syntax and semantics of sentences at different levels of abstraction from instances of surface sequence to linguistically motivated grammar such as Lexical-Functional Grammar (LFG) [Kaplan and Bresnan, 1982]. As shown in figure 2, a CSC consists of a root node (CSR), clement nodes (CSE), a FIRST link, a LAST link, NEXT link(s) and ROLE links. A CSR is a representative node for the meaning of the entire CSC structure. CSRs are connected to their designated interlingual concepts by ENG or JPN. Each CSC has one or more CSEs linked to a CSR by ROLE links. The ordering constraints between two concept sequence element nodes arc represented by NEXT link. FIRST and LAST links in each CSC points to the first and last elements, respectively. Also, each CSE represents the relevant case role, and the case role has a selectional restriction. Since we want to avoid heavy symbolic operations during parsing, ROLE links and associated constraint links are used instead of performing type and value consistency check by unification. Therefore each CSE is used for both enforcing the ordering constraint and capturing semantic information.

Besides, concept instance nodes (CI) and concept sequence instance structures (CSI) are dynamically created during parsing. Each CI or CSI is connected to the associated CC or

CSC by INST link. CIs correspond to discourse entities proposed in [Webber, 1983]. Three additional links are used to facilitate pragmatic inferences. They are CONTEXT links, CONSTRAINT links and EQROLE links. A CONTEXT link is a path of contextual priming which is crucial in word sense disambiguation. When a word is activated during processing, the activation spreads through CONTEXT links and impose contextual priming to relevant concepts. A CONSTRAINT link denotes an antecedent/consequence relationship between two events or states, which is created between two CSRs. An EQROLE link denotes the necessary argument matching condition for testing an antecedent/consequence relationship, which is created between two CSEs in different CSCs.

4.2 Syntactic Constraint Network

DMSNAP has a syntactic constraint network (SCN) which captures various syntactic constraints such as agreement, control, etc. Syntactic constraint network consists of syntactic constraint nodes (SC nodes), syntactic function nodes (SF nodes), and syntactic constraint links (SC-links). SC nodes represents syntactic constraints such as 3rd Singular Present (3SgPrCs) and Reflexive Pronoun (Ref). These nodes simply get bit markers to indicate whether these syntactic constraints are active or not, and send bit markers to show which lexical items are legal candidate for the next word. SF nodes represents grammatical functions such as functional controllers. SF nodes generally get an address marker and a bit marker. The address marker carries point to the CI nodes which should be bound to a case-frame in dislocated place in the network, and a bit marker shows whether the specific grammatical function node should be activated. When both a bit marker and an address marker exist at a certain SF node, the address marker is further propagated through SC-links to send information which is necessary to carry out interpretation of sentences involving control and unbounded dependency.

4.3 Markers

The processing of natural language on a marker-propagation architecture requires the creation and movement of markers on the memory network. The following types of markers are used:

A-MARKERS indicate activation of nodes. They propagate through ISA links upward, carry a pointer to the source of activation and a cost measure.

P-MARKERS indicate the next possible nodes to be activated. They are initially placed on the first element nodes of the CSCs, and move through NEXT link where they collide with A-MARKERS at the element nodes.

G-MARKERS indicate activation of nodes in the target language. They carry pointers to the lexical node to be lexicalized, and propagate through ISA links upward.

V-MARKERS indicate current state of the verbalization. When a V-MARKER collides with the G-MARKER, the surface string (which is specified by the pointer in the G-MARKER) is verbalized.

C-MARKERS indicate contextual priming. Nodes with C-MARKERS are contextually primed. A C-MARKER moves from the designated contextual root node to other contextually relevant nodes through contextual links.

SC-MARKERS indicate active syntax constraints, and primed and/or inhibited nodes by currently active syntactic constraints. It also carries pointer to specific nodes.

There are some other markers used for control process and timing, they are not described here. These five markers are sufficient to understand the central part of the algorithm in this paper.

4.4 DMSNAP Parsing Algorithm

Overall flow of the algorithm implemented on SNAP consists of the following steps:

1. Activate a lexical node
2. *Push* an A-Marker through ISA link; Pass SC-Markers through SC-link.
3. When the A-Marker collide with a P-Marker on CSE, the P Marker is passed through NEXT link once. However, if the CSE was the last element of the CSC, then the CSC is accepted and an A-Marker is passed up through ISA link from CSR of the accepted CSC.
4. When the P-Marker passed through the NEXT link in step 3, then a copy of the P Marker is passed down through inverse ISA link to make top-down prediction.
5. Pass SC-Markers from active nodes to activate and/or inhibit syntactic constraints, and to percolate pointers to the specific CL
6. Repeat 1 through 5 until all words are read.
7. Compute total cost for each hypothesis.
8. Select Lowest Cost Hypothesis.
9. Remove other hypotheses.

This parsing algorithm is similar to the shift-reduce parser except that our algorithms handles ambiguities, parallel processing of each hypothesis, and top-down predictions of possible next input symbol. The generation algorithm implemented on SNAP is a version of the lexically guided bottom-up algorithm which is described in [Kitano, 1990b].

5 Linguistic Processing in D M S N A P

We will explain how DMSNAP carries out linguistic analysis using two sets of examples:

```

|   Example I
|si John wanted to attend UC AI-91.
s2  He is at the conference.
s3  He said that the quality of the paper is superb.  |
1   Example II
s4  Dan planned to develop a parallel processing
    computer.
s5  Eric built a SNAP simulator.
s6  Juntae found bugs in the simulator.
s7  Dan tried to persuade Eric to help Juntae modify
    the simulator.
s8  Juntae solved a problem with the simulator.
|s9 It was the bug that Juntae mentioned.  |

```

The examples contain various linguistic phenomena such as: lexical ambiguity, structural ambiguity, referencing (pronoun reference, definite noun reference, etc), control, and unbounded dependencies. It should be noted that each example consists of a set of sentences (not a single sentence isolated

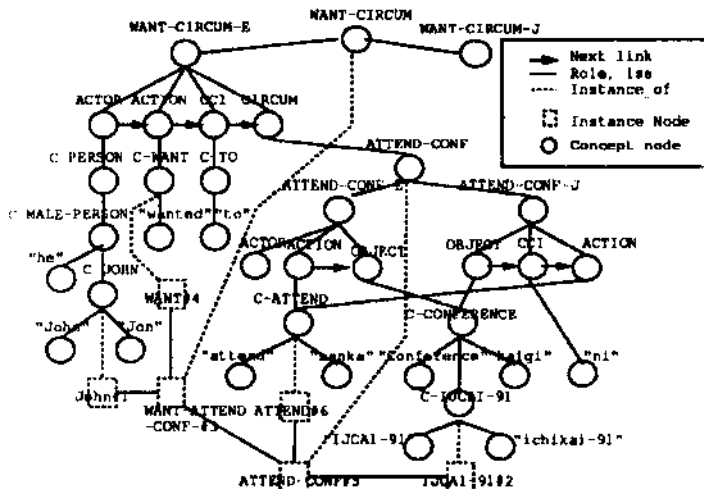


Figure 3: Part of Memory Network

from the context) in order to demonstrate contextual processing capability of the DMSNAP. These sentences are not all the sentences which DMSNAP can handle. Currently, DMSNAP handles substantial portion of the ATR's conference registration domain (vocabulary 450 words, 329 sentences) and sentences from other corpora.

5.1 Basic Parsing and Generation - Translation

The essence of DMSNAP parsing and generation algorithm is described using sentence *s1*. A part of memory network involved in this explanation is shown in figure 3. C- denotes concepts; and "... " denotes surface string in the lexical node. Notice that only a part of the memory network is shown and no part of the syntactic constraint network is shown here. Also, the following explanation does not describe activity of the syntactic constraint network part. This will be described in a relevant part later.

Initially, the first CSE in every *esc* on the memory network gets a P-MARKER. This P-MARKER is passed down ISA links. The CCs receiving a P-MARKER are C-PERSON and c-ATTEND. Also the closed class lexical items (CCI) in the target language propagate G-MARKER upward ISA links.

Upon processing the first word 'John' in the sentence *s1*, C-JOHN is activated so that C-JOHN gets an A-MARKER and a CI JOHN#1 is created under C-JOHN. At this point, the corresponding Japanese lexical item is searched for, and JON is found. A G-MARKER is created on JON. The A-MARKER and G-MARKER propagate up through ISA links (activating C-MALE-PERSON and C-PERSON in sequence) and, then, ROLE links. When an A-MARKER collides with a P-MARKER at a CSE, the associated case role *is* bound with the source of the A-MARKER and the prediction is updated by passing P-MARKER to the next CSE. This P-MARKER is passed down ISA links. In this memory network, the ACTOR roles of concept sequences WANT-CIRCUM-E is bound to JOHN#1 pointed by the A-MARKER. This is made possible in the SNAP architecture which allows markers to carry address as well as bit-vectors and values, where many other marker-passing machines such as NETL [Fahlman, 1979] and IXM2 [Higuchi et. al., 1991] only allow bit-vectors to be passed around. Also, G-MARKERS

are placed on the ACTOR role CSE of WANT-CIRCUM-J. The G-MARKER points to the Japanese lexical item 'jon'.

After processing 'wanted' and 'to', a P-MARKER is passed to CIRCUM and, then, to ATTEND-CONF. At this point, a source language (English) expression for the concept ATTEND-CONF is searched for and ATTEND-CONF-E is found. The first CSE of ATTEND-CONF-E gets a P-MARKER. After processing 'Attend' and 'IJCAI-91', ATTEND-CONF-E becomes fully recognized² so that a CSI having CIs is created under ATTEND-CONF-E. Then the associated concept ATTEND-CONF is activated. An A-MARKER is passed up from ATTEND-CONF to the last element of the CSR WANT-CIRCUM-E. As the result, the CSR WANT-CIRCUM-E and its CC WANT-CIRCUM are activated in sequence. Therefore the parsing result is represented by the activated CC WANT-CIRCUM and the associated CSI. Also, upon processing 'IJCAI-91' the concept C-CONFERENCE is activated and then C-MARKERS are passed to nodes connected to C-CONFERENCE by CONTEXT links. This is an operation for contextual priming.

When the parsing is done, a V-MARKER is passed to the target language (Japanese) expression WANT-CIRCUM-J from WANT-CIRCUM, and, then, to the first CSE of WANT-CIRCUM-J. Since the first CSE has a G-MARKER pointing to JON, 'jon' becomes the first word in the translated Japanese sentence and then the V-MARKER is passed to the next CSE. See [Kitano, 1990b] for the details of generation process. This operation is repeated for all CSEs in the CSC. Finally, the Japanese sentence *t1* is constructed for the English sentence *s1*.

With this algorithm, the first set of sentences (*s1*, *s2* and *s3*) is translated into Japanese:

- t1 Jon ha ichikai-91 ni sanku shilakatta.
- t2 Kare ha kaigi ni iro.
- t3 Kare ha ronbun no shitsu ga subarashii to itla.

5.2 Anaphora

Anaphoric reference is resolved by searching for discourse entity as represented by CIs under a specific type of concept node. Sentence *s2* contains anaphora problems due to 'He' and 'the conference'. When processing 'He' DMSNAP searches for any CIs under the concept C-MALE-PERSON and its subclass concepts such as C-JOHN. In the current discourse, JOHN#1 is found under C-JOHN. JOHN#1 and UCAI-91#2 are created when the *s1* is parsed. An A-MARKER pointing to JOHN#1 propagates up through ISA links. Likewise, IJCAI-91 #2 is found for C-CONFERENCE. In this sentence, there is only one discourse entity (ci in our model) as a candidate for each anaphoric reference, thus a simple instance search over the typed hierarchy network suffices. However, when there are multiple candidates, we use the centering theory by introducing forward-looking center (Cf), backward-looking center (Cb), etc [Brennan et. al., 1986]. Also, incorporating the notion of the *focus* is straightforward [Sidner, 1979].

5.3 Lexical Ambiguity

DMSNAP is capable of resolving this lexical ambiguity through use of *contextual priming* using the contextual marker (C-Marker) [Tomabechi, 1987] and the cost-based disambiguation [Kitano et. al., 1989]. Sentence *s3* contains a

²fully recognized means that the CSC can be reduced, in the shift-reduce parser's expression.

word sense ambiguity in the interpretation of the word 'paper' as either a technical document or a sheet of paper. Upon reading 'paper', C-THESIS and C-PAPER are activated. At this time, C-THESIS has a C-MARKER. The C-MARKER comes from activation of c-UCAI-91 and C-CONFERENCE, in previous sentences, which has contextual links connecting concepts relevant to academic conference such as C-THESIS. The meaning hypothesis containing C-THESIS costs less than the one with C-PAPER so that it is selected as the best hypothesis.

5.4 Control

Control is handled using the syntactic constraint network. Sentence s7 is an example of sentence *involving functional control* [Bresnan, 1982]. In s7, both *subject control* and *object control* exist - the subject of 'persuade*' should be the subject of 'tried' (*subject control*), and the subject of 'help' should be the object of 'persuade*' (*object control*). In this case, CSCs for *infinitival complement* has CSE without NEXT link. Such an CSE represents *missing subject*. There are SUBJ, OBJ, and OBJ2 nodes (these are *functional controller*) in the syntactic constraints network each of which store pointer to the CI node for possible controllee. Syntactic constraint links from each lexical items of the verb determine which functional controller is active. Activated functional controller propagate a pointer to the CI node to unbound subject nodes of cscs for infinitival complements. Basically, one set of nodes for functional controller handles deeply nested cases due to *functional locality*.

Take an example from s7, when processing 'Dan*', a pointer to an instance of 'Dan' which is C-DAN#1 is passed to SUBJ node of functional controller. Then, when processing 'tried', a SC-Marker propagates from the lexical node of 'tried*' to SUBJ through SC-link, and the SUBJ node is being activated. Then, the pointer to C-DAN#1 in the SUBJ node propagate to SUBJ role node (or ACTOR node) of the CSC for infinitival complement. After processing 'to' the csc for infinitival complement is predicted. Temporal bindings take place in each predicted CSC. When processing 'persuade', however, OBJ gets activated since 'persuade*' enforces *object control*, not *subject control*. Thus, after 'Eric' is processed, a pointer to an instance of 'Eric*' propagate in to the already active OBJ node, and then propagate to SUBJ role node (or ACTOR role node) of each CSC for infinitival complement. This way, DMSNAP performs control.

5.5 Structural Ambiguity

Structural ambiguity is resolved by the cost-based ambiguity resolution method [Kitano et. al., 1989]. The cost-based ambiguity resolution takes into account various psycholinguistic studies such as [Crain and Steedman, 1985] and [Ford et. al., 1981]. Sentence s8 contains a structural ambiguity in the PP-attachment. It can be interpreted either:

[s *juntae* [_{VP} *solved* [_{NP} *the problem*] [_{PP} *with* [_{NP} *the simulator*]]],
or

[s *juntae* [_{VP} *solved* [_{NP} *the problem* [_{PP} *with* [_{NP} *the simulator*]]]].

In this case, two hypotheses are activated at the end of the parse. Then, DMSNAP computes the cost of each hypothesis. Factors involved are contextual priming, lexical preference, existence of discourse entity, and consistency with world knowledge. In this example, the consistency with the world knowledge plays central role. The world knowledge

is a set of knowledge of common sense and knowledge obtained from understanding previous sentences. To resolve ambiguity in this example, the DMSNAP checks if there is a problem in the simulator. Constraint checks are performed by bit-marker propagation through CONSTRAINT links and EQROLE links. Since there is a CI which packages instances of ERROR and SNAP-SIMULATOR then the constraint is satisfied and the second interpretation incurs no cost from constraint check. However, there is no CI which packages instances of JUNTAE and SNAP-SIMULATOR. Therefore the first interpretation incurs a cost of constraint violation (15 in our current implementation). Thus DMSNAP is able to interpret the structural ambiguity in favor of the second interpretation.

5.6 Unbounded Dependency

There are two ways to handle sentences with unbounded dependency. The first approach is straightforward memory-based approach which simply store a set of CSCs involves unbounded dependency. A large set of CSCs would have to be prepared for this, but its simplicity minimized computational requirements. Alternatively, we can employ somewhat linguistic treatment of this phenomena within our framework. The syntactic constraint network has a node representing TOPIC and FOCUS which usually bound to the displaced phrase. An address of CI for the displaced phrase (such as 'the bug' in the example s9) is propagated to the TOPIC or FOCUS nodes in the syntactic constraint network. Further propagation of the address of the CI is controlled by activation of nodes along the syntactic constraint network. The network virtually encodes a finite-state transition equivalent to {COMPIXCOMP}*GF-COMP [Kaplan and Zaenen, 1989] where GF-COMP denotes grammatical functions other than COMP. The address of the CI bound to TOPIC or FOCUS can propagate through the path based on the activation patterns of the syntactic constraint network, and the activation patterns are essentially controlled by markers flow from the memory network. When the CSC is accepted and there is a case-role not bound to any CI (OBJECT in the example), the CSE for the case-role bound with the CI propagated from the syntactic constraint network.

6 Performance

DMSNAP complete parsing in the order of milliseconds. While actual SNAP hardware is now being assembled and to be fully operational by May 1991, this section provides performance estimation with precise simulation of the SNAP machine. Simulations of the DMSNAP algorithm have been performed on a SUN 3/280 using the SNAP simulator which has been developed at USC [Lin and Moldovan, 1990]. The simulator is implemented in both SUN Common LISP and C, and simulates the SNAP machine at the processor level. The LISP version of the simulators also provides information about the number of SNAP clock cycles required to perform the simulation.

There are two versions of DMSNAP, one written in LISP and one in C. The high-level languages only take care of the process flow control, and the actual processing is done with SNAP instructions. The performance data summarized in Table 2 was obtained with the first version of DMSNAP written in LISP. Furthermore, with a clock speed of 10 MHz, these execution times are in the order of 1 millisecond. These and

Sentence	Length (words)	Number of machine cycle	Time at 10 MHz (msec)
s2: He is at ...	4	6500	0.65
s3: He said that ...	10	15000	1.50
s5: Eric build ...	5	5500	0.55
s6: Juniae found ...	6	10500	1.05
s8: Juniae solved ...	7	16500	1.65

Table 1: Execution times for DmSNAP

other simulation results verify the operation of the algorithm and indicate that typical runtimes is on the order of milliseconds per sentence.

The size of the memory network for example 11 is far larger than that of example 1, yet we see no notable increase in the processing time. This is due to the use of a guided marker-passing which constraints propagation paths of markers. Our analysis of the algorithm shows that parsing time grow only to sublinear to the size of the network.

7 Conclusion

In this paper, we have demonstrated that high-performance natural language processing with parsing speeds in the order of milliseconds is achievable without making substantial compromise in linguistic analysis. To the contrary, our model is superior to other traditional natural language processing models in several aspects, particularly, in contextual processing.

The D M S N A P is based on the idea of memory-based model of natural language processing. The D M S N A P is a variation of the O D M D I A L O G speech-to-speech dialog translation system. We use the parallel marker-passing scheme to perform parsing, generation, and inferencing. The syntactic constraint network was introduced to handle linguistically complex phenomena without undermining benefits of the memory-based approach.

Not only the D M S N A P exhibits high-performance natural language processing, but also demonstrates capabilities to carry out linguistically sound parsing particularly on contextual processing. The use of the memory network to distributively represent knowledge and modify it to reflect new states of the mental model is an effective way to handle such phenomena as pronoun reference and control.

In summary, we demonstrated that the model presented in this paper is a promising approach to high-performance natural language processing with highly contextual and linguistics sound processing. We hope to extend this work to the real-world domains in the near-future. We are convinced that millisecond performance opens new possibilities for natural language processing.

References

[Blelloch, 1986] Blelloch, G. E., "CIS: A Massively Parallel Concurrent Rule-Based System," *Proceeding of AAAI-86*, 1986.

[Brachman and Schmolz, 1985] Brachman, R. J. and Schmolz, J. G., "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science**, 17:1-216, August 1985.

[Brennan et al., 1986] Brennan, S., Friedman, M. and Pollard, C. "A Centering Approach to Pronouns" *Proceedings of the ACL-86*, 1986.

[Bresnan, 1982] Bresnan, J., "Control and Complementation," *The Mental Representation of Grammatical Relations*, The MIT Press, 1982.

[Crain and Steedman, 1985] Crain, S. and Steedman, M., "On not being let up with the garden path: the use of context by the psychological syntax processor." In *Natural Language Parsing*, Dowty, D., Karttunen, L., and Zwicky, A. (Eds.), Cambridge University Press, 1985.

[Fahlman, 1979] Fahlman, S., *NETL: A System for Representing and Using Real-World Knowledge*, The MIT Press, 1979.

[Ford et al., 1981] Ford, M., Bresnan, J. and Kaplan, R. "A Competence-Based Theory of Syntactic Closure," Bresnan, J. (Ed.) *The Mental Representation of Grammatical Relations*. The MIT Press, 1981.

[Gibson, 1990] Gibson, T., "Memory Capacity and Sentence Processing," *Proceedings of ACL-90*, 1990.

[Higuchi et al., 1991] Higuchi, T., Kiuno, H., Handa, K., Furuya, T., Takahashi, H., and Kokubu, A., "IXM2: A Parallel Associative Processor for Knowledge Processing," *Proceedings of AAAI-91*, 1991.

[Ungria, 1990] Ungria, R., "The Limit of Unification," *Proceedings of ACL-90*, 1990.

[Kaplan and Bresnan, 1982] Kaplan, R. and Bresnan, J., "Lexical-Functional Grammar: A Formal System for Grammatical Representation," Bresnan, J. (Ed.), *The Mental Representation of Grammatical Relations*, Cambridge, MA: MIT Press, 1982.

[Kaplan and Zaenen, 1989] Kaplan, R. and Zaenen, A., "Long-distance Dependencies, Constituent Structure, and Functional Uncertainty," 1989.

[Kasper, 1989] Kasper, R., "Unification and Classification: An Experiment in Information-Based Parsing," *Proceedings of the International Workshop on Parsing Technologies*, Pittsburgh, 1989.

[Kim and Moldovan, 1990] Kim, J. and Moldovan, D., "Parallel Classification for Knowledge Representation on SNAP" *Proceedings of the 1990 International Conference on Parallel Processing*, 1990.

[Kiuno, 1991a] Kiuno, H., "ODMDialog: An Experimental Speech-to-Speech Dialogue Translation System," *IEEE Computer*, June, 1991.

[Kiuno, 1991b] Kiuno, H., "Unification Algorithm for Massively Parallel Computers," *Proceedings of the International Workshop on Parsing Technologies*, Cancun, 1991.

[Kiuno and Higuchi, 1991 a] Kiuno, H. and Higuchi, T., "Massively Parallel Memory-Based Parsing," *Proceedings of IJCAI-91*, 1991.

[Kiuno and Higuchi, 1991 b] Kiuno, H. and Higuchi, T., "High Performance Memory Based Translation on IXM2 Massively Parallel Associative Memory Processor," *Proceedings of AAAI-91*, 1991.

[Kiuno, 1990a] Kitano, H., "DmDialog: A Speech-to-Speech Dialogue Translation System," *Machine Translation*, 5, 1990.

[Kitano, 1990b] Kitano, H., "Parallel Incremental Sentence Production for a Model of Simultaneous Interpretation," Dale, R., Mellish, C., and Zock, M. (Eds.) *Current Research in Natural Language Generation*, Academic Press, London, 1990.

[Kiuno et al., 1989] Kiuno, H., Tomabechi, H., and Levin, L., "Ambiguity Resolution in DmTrans Plus," *Proceedings of the European Chapter of the Association of Computational Linguistics*, 1989.

[Lee and Moldovan, 1990] Lee, W. and Moldovan, D., "The Design of a Marker Passing Architecture for Knowledge Processing," *Proceedings of AAAI-90*, 1990.

[Lin and Moldovan, 1990] Lin, C. and Moldovan, D., "SNAP: Simulator Results", Technical Report PKPL 90-5, University of Southern California, Department of EE-Systems, 1990.

[Moldovan and Lee, 1990] Moldovan, D., Lee, W., and Lin, C., "SNAP: A Marker Propagation Architecture for Knowledge Processing", Technical Report PKPL 90-1, University of Southern California, Department of EE-Systems, 1990.

[Nagao, 1984] Nagao, M., "A Framework of a Mechanical Translation between Japanese and English by Analogy Principle," *Artificial and Human Intelligence*, Elithom, A. and Banerji, R. (Eds.), Elsevier Science Publishers, B.V. 1984.

[Pollard and Sag, 1987] Pollard, C. and Sag, I., *Information-Based Syntax and Semantics. Vol. I: Fundamentals*, CSLI Lecture Note Series, Chicago University Press, 1987.

[Quillian, 1968] Quillian, M. R. "Semantic Memory," *Semantic Information Processing*, Minsky, M. (Ed.), 216-270, The MIT Press, Cambridge, MA, 1968.

[Riesbeck and Martin, 1985] Riesbeck, C. and Martin, C., "Direct Memory Access Parsing", Yale University Report 354, 1985.

[Riesbeck and Schank, 1989] Riesbeck, C. and Schank, R., *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, 1989.

[Sidner, 1979] Sidner, C., *Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse*, Ph. D. Thesis, Artificial Intelligence Lab., M.I.T., 1979.

[Sowa, 1984] Sowa, J. P., *Conceptual Structures*, Reading, Addison Wesley, 1984.

[Sunfill and Waltz, 1986] Sunfill, C. and Waltz, D., "Toward Memory-Based Reasoning," *Communication of the ACM*, 1986.

[Sumita and Iida, 1991] Sumita, E., and Iida, H., "Experiments and Prospects of Example-Based Machine Translation," *Proceedings of ACL-91*, 1991.

[Thinking Machine Corp., 1989] Thinking Machine Corp., *Model CM-2 Technical Summary*, Technical Report TR-89-1, 1989.

[Tomabechi, 1987] Tomabechi, H., "Direct Memory Access Translation." *Proceedings of the UCAI-87*, 1987.

[Wiensky, 1987] Wiensky, R., "Same Problems and Proposals for Knowledge Representation", Technical Report UCBCSD 87/351, University of California, Berkeley, Computer Science Division, 1987.

[Webber, 1983] Webber, B., "So What Can We Talk About Now?" Brady, M. and Berwick, R. (Eds) *Computational Models of Discourse*, The MIT Press, 1983.