[International Workshop on Fundamental Research for the Future Generation of Natural Language Processing (FGNLP), July 1992, Manchester]

# Utility of Massively Parallel Computing Platform in Natural Language Processing*

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

July 23, 1992

**Abstract**

This paper demonstrates the utility of the Semantic Network Array Processor (SNAP) as a massively parallel platform for high performance and large-scale natural language processing systems. SNAP is an experimental massively parallel machine which is dedicated to, but not limited to, the natural language processing using semantic networks. In designing the SNAP, we have investigated various natural language processing systems and theories to determine the scope of the hardware support and a set of micro-coded instructions to be provided. As a result, SNAP employs an extended marker-passing model and a dynamically modifiable network model. A set of primitive instructions is micro-coded to directly support a parallel marker-passing, bit-operations, numeric operations, network modifications, and other essential functions for natural language processing. This paper demonstrates the utility of SNAP for various paradigms of natural language processing. We have discovered that the SNAP provides milliseconds or microseconds performance on several important applications such as the memory-based parsing and translation, classification-based parsing, and VLKB search. Also, we argue that there are numerous opportunities in the NLP community to take advantages of the computational power of the SNAP.

**Keywords:** Massively Parallel Computing, Memory Based Translation, VLKB, Classification based Parsing

*This research is supported by the National Science Foundation under grant MIP-9009111, and conducted as a part of IMPACT (International Consortium for Massively Parallel Advanced Computing Technologies). A version of this paper appears in COLING-92.

# 1 Introduction

In order to accomplish the high-performance natural language processing, we have designed a highly parallel machine called Semantic Network Array Processor (SNAP) [Lee and Moldovan, 1990]. The goal of our project is to develop and test the validity of the massively parallel machine for high performance and large-scale natural language processing. Thus, the architecture of the SNAP was determined reflecting extensive analysis of basic operations essential to the natural language processing. As a result of the investigation, we have decided to employ an extended marker-passing model and a dynamically modifiable network. Also, a set of primitive instructions is micro-coded to directly support essential operations in natural language systems.

Several approach can be taken to use SNAP as a platform for natural language processing systems. We can fully implement NLP system on SNAP, or we can speed up existing systems by implementing computationally expensive parts on SNAP. We have implemented some of these approaches on SNAP, and obtained extremely high performance (order of milliseconds for given tasks).

In this paper, we describe the design philosophy and architecture of SNAP, and present several approaches toward high performance natural language processing systems on SNAP.

# 2 SNAP Architecture

## 2.1 Design Philosophy of SNAP

The Semantic Network Array Processor (SNAP) is a highly parallel array processor fully optimized for semantic network processing with a marker-passing mechanism. The fundamental design decisions are (1) a semantic network as a knowledge representation scheme, and (2) parallel marker-passing as an inference mechanism.

First, the use of a semantic network as a representation scheme can be justified from the fact that most of the representation schemes of current AI and NLP theories (such as frame, feature structure, sort hierarchy, systemic choice network, neural network, etc.) can be mapped onto semantic networks. Also, there are numbers of systems and models which directly use semantic networks [Sowa, 1991].

Second, the use of marker-passing can be justified from several aspects. Obviously, there are many AI and NLP models which use some form of marker-passing as the central computing principle. For example, there are significant number of research being done on word-sense disambiguation as seen in [Waltz and Pollack, 1985], [Hendler, 1988], [Hirst, 1986], [Charniak, 1983], [Tomabechi, 1987], etc. All of them assume passing of markers or values among nodes interconnected via some types of links. There are studies to handle syntactic constraints using some type of networks which can be mapped onto semantic networks. Recent studies on the Classification-Based Parsing [Kasper, 1989] and the Systemic Choice Network [Carpenter and Pollard, 1991] assume hierarchical networks to represent various linguistic constraints, and the search on these networks can be done by marker-passing. Also, there are more radical approaches to implement entire natural language systems using parallel marker-passing as seen in [Norvig, 1986], [Riesbeck and Martin, 1985], [Tomabechi, 1987], and [Kitano, 1991]. There are, however, differences in types of information carried in each marker-passing model. We will describe our design decisions later.

As reported in [Evett, et al., 1990], however, serial machines are not suitable for such processing because it causes performance degradation as size of semantic network increases.

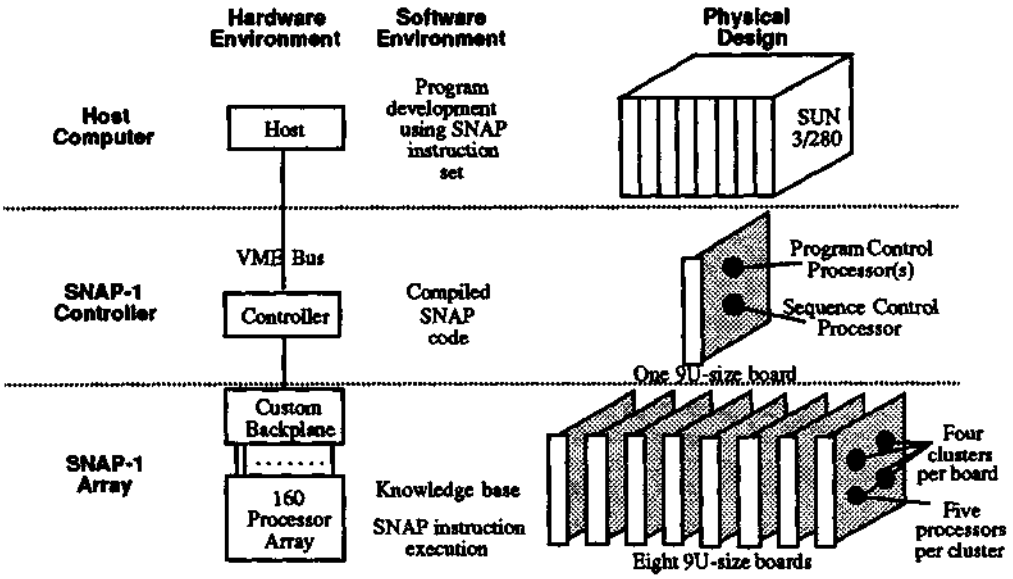


Figure 1: SNAP-1 Architecture

There axe clear needs for highly parallel machines. The rest of this section provides a brief overview of the SNAP architecture.

## 2.2 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 a TMS320C30 DSP chip, local SRAM, etc. Each processing elements 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.

## 2.3 Parallel Marker-Passing

In the SNAP, content of the marker are: (1) bit-vector, (2) address, and (3) numeric value (integer or floating point). In SNAP, the size of the marker is fixed. According to the classification in [Blelloch, 1986], our model is a kind of *Finite Message Passing.* There are types of marker-, or message-, passing that propagates feature structures (or graphs), which are called *Unbounded Message Passing.* Although we have extended our marker-passing model from the traditional bit marker-passing to the complex marker-passing which carries bits, address, and numeric values, we decided not to carry unbounded messages. This is because propagation of feature structures and heavy symbolic operations at each PE 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 the size of the message is unbounded, which means that the designer can not be sure if the local memory size is sufficient or not until the machine actually runs some applications. Also, PEs capable of performing operations to manipulate these messages (such as unification) would

be large in physical size which causes assembly problems when thousands of processors are to be assembled into one machine. Since we decide not to support unbounded message passing, we decide to support functionalities attained by the unbounded message passing by other means such as sophisticated marker control rules, dynamic network modifications, etc.

## 2.4 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 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 Machines Corp., 1989], but specialized for semantic network processing with marker passing.

Particularly important is the marker propagation rules. Several marker propagation rules are provided to govern the movement of markers. Marker propagation rules enables us to implement guided, or constraint, marker passing as well as unguided marker passing. This is done by specifying the type of links that markers can propagate. The following are some of the propagation rules of SNAP:

- *Seq(r1,r2) :* The *Seq* (sequence) propagation rule allows the marker to propagate through *r1* once then to *r2.*
- *Spread(r1,r2) :* The *Spread* propagation rule allows the marker to travel through a chain of *r1* links and then *r2* links.
- *Comb(r1,r2) :* The *Comb* (combine) propagation rule allows the marker to propagate to all *r1* and *r2* links without limitation.

## 2.5 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 kind of graph-based knowledge representation formalisms such as KL-ONE [Brachman and Schmolze, 1985], Conceptual Graphs [Sowa, 1984], KODIAK [Wilensky, 1987], etc.

# 3 The 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 has been proposed in this direction, such as Direct Memory Access Parsing (DMAP) [Riesbeck and Martin, 1985] and ΦDMDIALOG [Kitano, 1991]. For arguments concerning superiority of the memory-based approach over the traditional approach, see [Nagao, 1984], [Riesbeck and Martin, 1985], and [Sumita and Iida, 1991].

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

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.

**Memory Network on SNAP** The major types of knowledge required for language translation 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 part of the memory network is shown in Figure 2.

**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: (1) A-Markers indicate activation of nodes. They propagate through ISA links upward, carry a pointer to the source of activation and a cost measure, (2) 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, (3) 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, (4) 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, (5) 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, and (6) 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.

The parsing algorithm is similar to the shift-reduce parser except that our algorithms handle 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, 1990]. Details of the algorithm is described in [Kitano et al., 1991b].

DMSNAP can handle various linguistic phenomena such as: lexical ambiguity, structural ambiguity, referencing (pronoun reference, definite noun reference, etc), control, and unbounded dependencies. Linguistically complex phenomena are handled using the syntactic constraint network (SCN). The SCN enables the DMSNAP to process sentences involving unbounded dependencies, controls without passing feature structures. Details of the SCN is described in [Kitano et al., 1991b]. One notable feature of DMSNAP is its capability to parse and translate sentences in context. In other words, DMSNAP can store results of previous sentences and resolve various levels of ambiguities using the contextual information. Examples of sentences which DMSNAP can handle is shown below. It should be noted that each example consists of a set of sentences (not a single sentence isolated from the context) in order to demonstrate the contextual processing capability of the DMSNAP.

| | Example I |
|---|---|
| s1 | John wanted to attend Coling-92. |
| s2 | He is at the conference. |
| s3 | He said that the quality of the paper is superb. |
| | **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. |

These sentences in examples axe not all the sentences which DMSNAP can handle. Currently, DMSNAP handles a substantial portion of the ATR conference registration domain (vocabulary 450 words, 329 sentences) and sentences from other corpora.

The following are examples of translation into Japanese generated by the DMSNAP for the first set of sentences (s1, s*2* and s3):

t1 Jon ha koringu-92 ni sanka shitakatta.
t2 Kare ha kaigi ni iru.
t3 Kare ha ronbun no shitsu ga subarashii to itta.

DMSNAP completes the parsing in the order of milliseconds. Table 1 shows parsing time for some of the example sentences.

# 4 Classification-Based Parsing

Classification-Based Parsing is a new parsing model proposed in [Kasper, 1989]. In the classification-based parsing, feature structures are indexed in the hierarchical network, and an unifiability of two feature structures are tested by searching the Most Specific Subsumer (MSS). The unification, a computationally expensive operation which is the computational bottleneck of many parsing systems, is replaced by search in the lattice of pre-indexed feature structures.

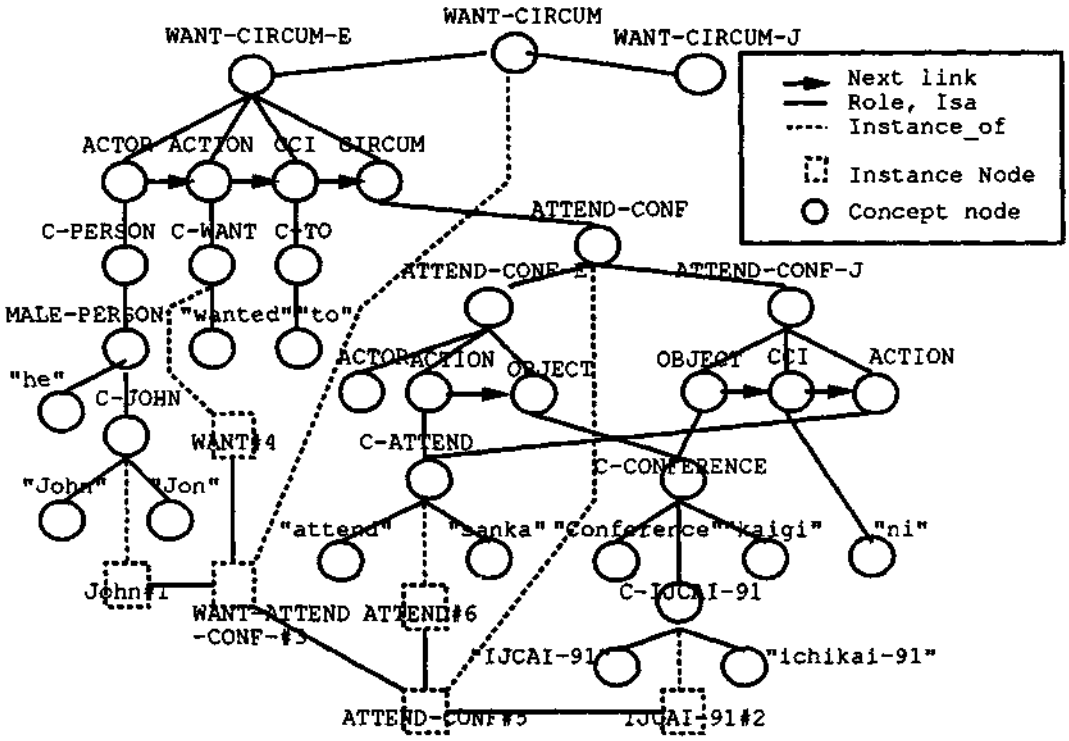


Figure 2: Part of Memory Network

| Sentence | Length (words) | Time at 10 MHz (msec) |
|---|---|---|
| s2: He is at ... | 4 | 0.65 |
| s3: He said that ... | 10 | 1.50 |
| s5: Eric build ... | 5 | 0.55 |
| s6: Juntae found ... | 6 | 1.05 |
| s8: Juntae solved ... | 7 | 1.65 |

Table 1: Execution times for DMSNAP

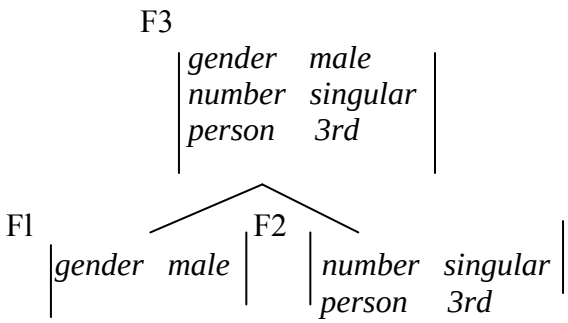


Figure 3: A part of a simple example of classification lattice

For example, in Figure 3, the feature structure F3 is a result of successful unification of the feature structure F1 and F2 (F3 = F1 ∪ F2). All feature structures are pre-indexed in a lattice so that the unification is replaced by an intersection search in the lattice with complex indexing. To carry out a search, first we set distinct markers on each feature structures F1 and F2. For example, set marker M1 on F1, and M2 on F2. Then, markers Ml and M2 propagate upward in the lattice. Ml and M2 first co-exist at F3. The most simple program (without disjunctions and conjunctions handling) for this operation follows:

```
set_marker(Ml,f1);
set_marker(M2,f2);
propagate(M1,M1,UP,UP,SPREAD);
propagate(M2,M2,OP,UP,SPREAD);
marker_and(Ml,M2,M3);
propagate (M3, m_tmp, UP, UP, SPREAD);
cond_clear_marker(m_tmp,M3);
collect_nodes(M3);
```

Of course, nodes for each feature structure may need to be searched from a set of features, instead of direct marking. In such a case, a set of markers will be propagated from each node representing each feature, and takes disjunction and conjunction at all nodes representing a feature structure root. This operation can be data-parallel.

There are several motivations to use classification-based parsing, some of which are described in [Kasper, 1989]. The efficiency consideration is one of the major reasons for using classification-based parsing. Since over 80% of parsing time has been consumed on unification operations, replacing unification by a faster and functionally equivalent method would substantially benefit the overall performance of the system. The classification-based parsing is efficient because (1) it maximizes structure sharing, (2) it utilizes indexing dependencies, and (3) it avoids redundant computations. However, these advantages of the classification-based parsing can not be fully obtained if the model was implemented on the serial machine. This is because a search on complex index lattice would be computationally expensive for serial machines. Actually, the time-complexity of the sequential classification algorithm is $O(Mn^2)$, and that of the retrieval algorithm is $O(R_{ave}logM)$, where $M$ is a number of concepts, $n$ is an average number of property links per concept, $R_{ave}$ *is* an average number of roleset relations for one concept. We can, however, circumvent this problem by using SNAP. Theoretically, time-complexity of the classification on SNAP is $O(log_{Fout}M)$, and that of the parallel retrieval is $O(F_{in}D_{ave} + Rave)$, where $F_{out}$ is an average fan-out (average number of subconcepts for one concept), $F_{in}$ is an average fan-in (average number of superconcept for one concept), and $D_{ave}$ is an average depth of the concept hierarchy [Kim and Moldovan, 1990].

In our model, possible feature structures are pre-computed and indexed using our classification algorithms. While a large set of feature structures need to be stored and indexed, SNAP provides sufficiently large memory/processor space to load an entire feature structure lattice. It is analogous to the idea behind the memory-based parsing which pre-expand all possible syntactic/semantic structures. Here again, we see the conversion of time-complexity into space-complexity.

Figure 4 shows performance of retrieval of classification lattice with varying fan-out and size. The clock cycle is 10 MHz. It demonstrates that we can attain micro-seconds response for each search. Given the fact that the fastest unification algorithm, even on the parallel machines, takes over few milliseconds per unification, the performance obtained in our experiment promises a significant improvement in parsing speed for many of the unification-based parsers by replacing unification by classification-based approach.

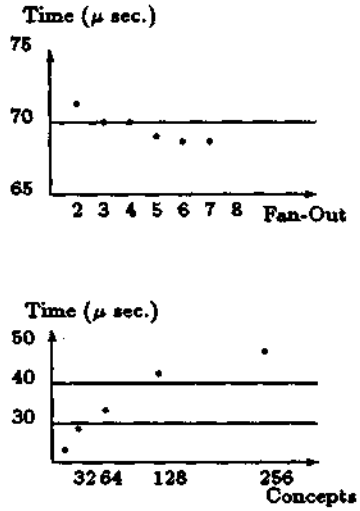


Figure 4: Retrieved Performance on Classification Network

# 5 VLKB Search: Integration with the Knowledge-Based Machine Translation

Language processing is a knowledge-intensive process. Knowledge-Based Machine Translation (KBMT) [Goodman and Nirenburg, 1991] has been proposed and developed based on the assumption that intensive use of linguistic and world knowledge would provide high quality automatic translation.

One of the central knowledge sources of the KBMT is the ontological hierarchy which encodes abstraction hierarchies of concepts in the given domain, property information of each concept, etc. When a parser creates ambiguous parses or when some parts of the meaning representation (as represented in an interlingua) are missing, this knowledge source is accessed to disambiguate or to fill-in missing information.

However, as the size of the domain scales up, access time to the knowledge source grows to the extent that cost-effective bulk processing would not be possible. For example, [Evett, et al., 1990] reports that access to large frame systems on serial computers have a time-complexity of $O(M \times B^d)$ where M is the number of conjuncts in the query, B is the average branching factor in the network, and $d$ is the depth of the network. Thus, even a simplest form of search takes over 6 seconds on a VLKB with 28K nodes measured on a single user mode VAX super mini-computer. Since such search on a VLKB must be performed several times for each parse, the performance issue would be a major concern. Considering the fact that VLKB projects such as CYC [Lenat and Guha, 1990] and EDR [EDR, 1988] aim at VLKBs containing over a million concepts, the performance of VLKB search would be an obvious problem in practical use of these VLKBs. In the massively parallel machines such as SNAP, we should be able to attain time-complexity of $O(D + M)$ [Evett, et al., 1990].

We have carried out experiments to measure KB access time on SNAP. Figure 5 shows the search time for various size of VLKBs ranging from 800 to 64K nodes. Performance was compared with SUN-4 and the CM-2 connection machine. SNAP-1 consistently outperformed other machines (performance curve of SNAP-1 is hard to see in the figure as it exhibited execution time far less than a second.

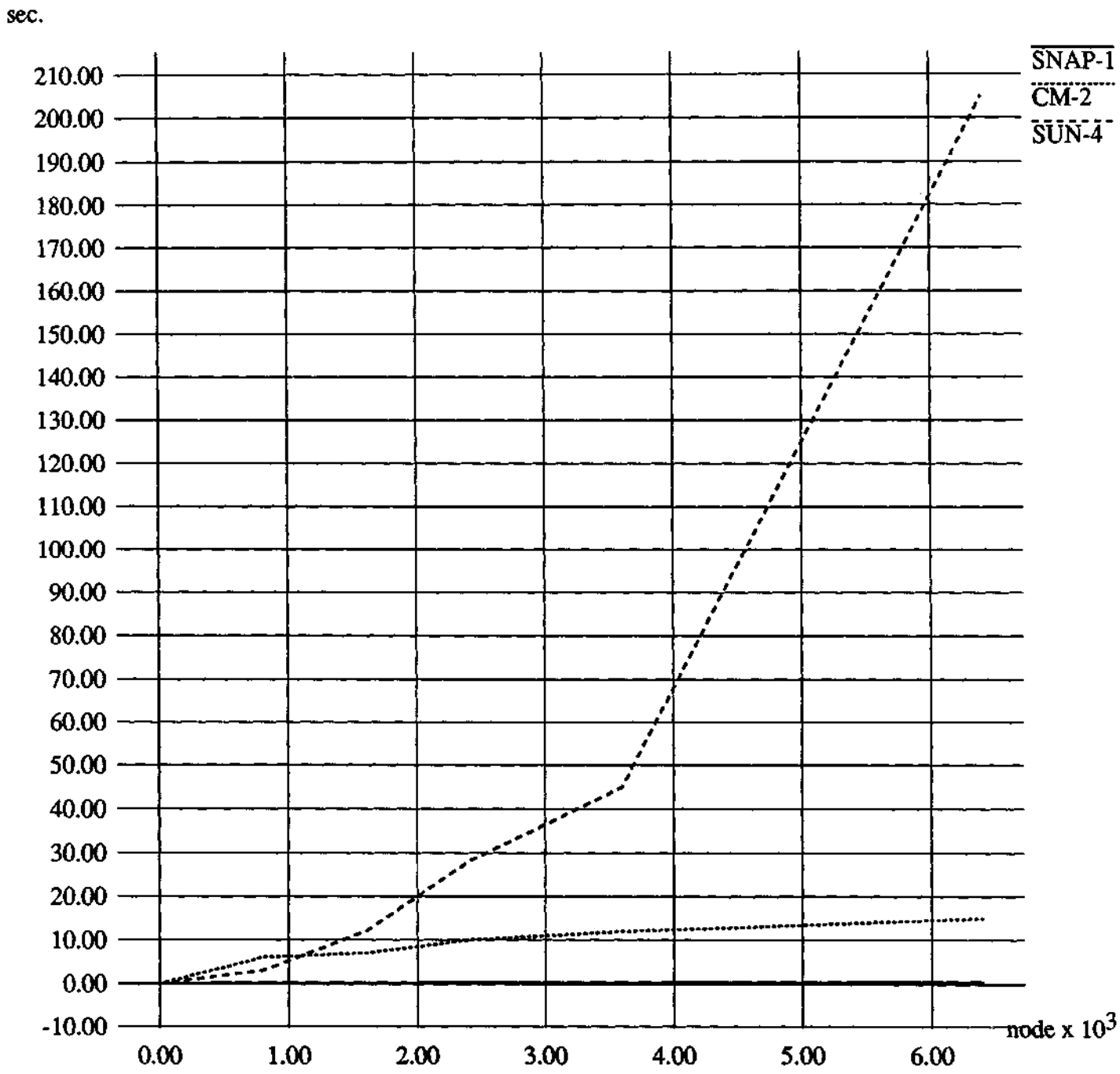


Figure 5: Retrieval time vs. KB size

# 6 Other Approaches

One clear extension of the currently implemented modules is to integrate the classification-based parsing and the VLKB search. The classification-based parsing carry out high performance syntactic analysis and the VLKB search would impose semantic constraints. Integration of these two would require that the SNAP-1 to have a multiple controller because two different marker control processes need to be mixed and executed at the same time. Currently SNAP-1 has only one controller. This would be one of the major items for the up-grade of the architecture. However, the performance gain by this approach would be significant and its impact can be far reaching because a lot of current NLP research has been carried out on the framework of the unification-based grammar formalism and use VLKBs as major knowledge sources.

A more radical approach, however rooted in the traditional model, is to fully map the typed unification grammars [Emele and Zajac, 1990] on the SNAP. The typed unification grammar is based on the Typed Feature Structure (TFS) [Zajac, 1989] and HPSG [Pollard and Sag, 1987], and represents all objects in TFS. Objects includes Phrasal Sign, Lexical Sign, general principles such as the "Head Feature Principle", the "Subcat Feature Principle", grammar rules such as the "Complement Head Constituent Order Feature Principle," the "Head Complements Constituent Order Feature Principle," and lexical entries. The lexical entries can be indexed under the lexical hierarchy. In this approach, all linguistic knowledge is precompiled into a huge network. Parsing and generation will be carried out as a search on this network. We have not yet complete a feasibility study for this approach on SNAP. However, as of today, we consider this approach is feasible and expect to attain single-digit millisecond order performance on an actual implementation. The dynamic network modification, address propagation, and marker propagation rules are especially useful in implementing this approach.

Natural language processing model on semantic networks such as [Norvig, 1986], SNePS [Neal and Shapiro, 1987], and TRUMP, KING, ACE, and SCISOR at GE Lab. [Jacobs, 1991] should fit well with the SNAP-1 architecture. For [Norvig, 1986], SNAP provides floating point numbers to be propagated. As for SNePS, the implementation should be trivial, yet we are not sure the level of parallelism gain by the SNePS model. When the parallelism was found to be low, the coarse-grain processor may fit well with this model. Although we do not have space to discuss in this paper, there are, of course, many other NLP and AI models which can be implemented on SNAP.

# 7 STAR: Subsumptive Translation Architecture

Let us now put all these things into one perspective. We wish to build a theory which takes maximum advantages of memory-based processing, and rule-based processing. The architecture should entail memory-based process at the most specific layer, VLKB search for symbolical-level of constraint satisfaction operation, and classification-based parsing or TFS system to handle abstract grammar-based processing. Memory-based and rule-based processes are no longer mutually exclusive, they are complementary processes (Figure 6).

Also, we have seen elsewhere [Kitano, 1991] that there are several levels of translation such as lexical, phrasal, and sentential.

We argue that the architecture for machine translation systems should entail all these levels of processing in a consistent manner. The STAR Theory is under development to satisfy the requirements argued above.

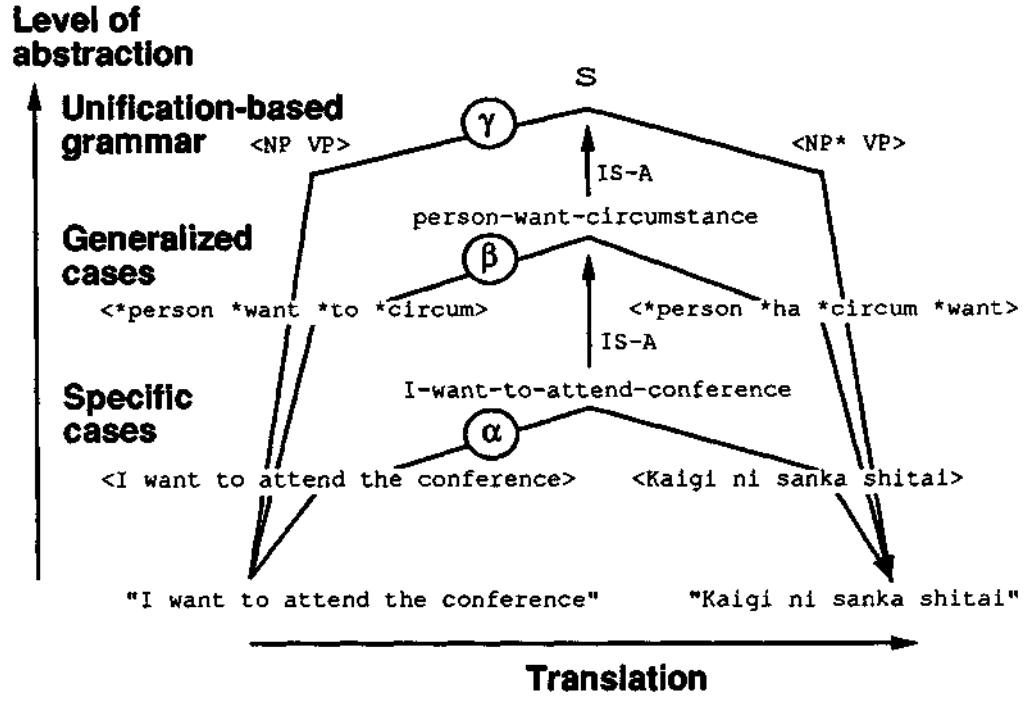


Figure 6: Translation at different levels of abstraction

In the STAR theory, there are several levels of translation which are executed autonomously but influenced by higher-levels of processing. Lexical translation which is a autonomous process is governed by phrasal-leve of translation which, in turn, governed by sentence-level translation. This is analogous to the subsumption architecture [Brooks, 1986].

It is subsumptive from the different point of view, too. Memory-based process is subsumed by the processing level of the generalized-case. The processing at the generalized-case level is subsumed by the rule-base processing. The subsumptive relation in this context is the range of coverage, and should be distinguished from the subsumption relation in between lexical, phrasal, and sentence translation.

Partial implementation on massively parallel machines has been made in DMSNAP and in ASTRAL [Kitano and Higuchi, 1991b]. These systems has been derived from the ΦDMDIALOG system which, in retrospect, was the first machine translation system based on the STAR theory, though it was an incomplete model of the STAR theory.

Justification of this theory comes from several psycholinguistic studies, but the strongest motivation was the author's intuition as a professional simultaneous interpreter.

It is often the case that, when a human interpreter make mistake or when her/his translation collapse, mis-translation or confusion start from the middle of the sentence leaving initial part of the sentence unaffected. Obviously, there is an incremental understanding, translation and generation mechanism which prevents total catastrophe of the translation.

Most current machine translation system would simply halt when it encounters serious flaw in parsing process or when the middle of the sentence was completely ungrammatical.

It is our speculation that the source of the robustness of the human interpreters (against ungrammatical sentences and against the interpreter's own error) is the basic architecture for translation process which is distributed, incremental, parallel, and subsumptive.

Unfortunately, the current implementations are not yet to explore the robustness of the STAR theory as the theory itself is under development.

# 8 Conclusion

In this paper, we have demonstrated that semantic network array processor (SNAP) speeds up various natural language processing tasks. We have demonstrated this fact using three examples: the memory-based parsing, VLKB processing, and Classification-based parsing.

In the memory-based parsing approach, we have attained the speed of parsing in the order of milliseconds without making substantial compromises in linguistic analysis. To the contrary, our model is superior to other traditional natural language processing models in several aspects, particularly, contextual processing.

Next, we have applied the SNAP architecture for a new classification-based parsing model. Here, SNAP is used to search the MSS to test the unifiability of the two feature graphs. We have attained, again, sub-milliseconds order performance per unifiability test. In addition, this approach exhibited desirable scalability characteristics. The search time asymptotically researches to 450 cycles as the size of classification network increases. Also, search time decreases as average fan-out gets larger. These are natural advantages of using parallel machines.

SNAP is not only useful for the new and radical approach, but also beneficial in speeding up traditional NLP systems such as KBMT. We have evaluated the performance to search VLKB which is the major knowledge source for the KBMT system. We have attained sub-milliseconds order performance per a search. Traditionally, on the serial machines, this process has been taking a few seconds posing the major thread to performance on the scaled up systems.

Also, there are many other NLP models (Typed Unification Grammar [Emele and Zajac, 1990], SNePS [Neal and Shapiro, 1987], and others) which may exhibit high performance and desirable scaling property on SNAP.

Currently, we are designing the SNAP-2 reflecting various findings made by the research with SNAP-1. SNAP-2 will be built upon the state-of-the-art VLSI technologies using RISC architecture. At least 32K virtual nodes will be supported by each processing element, providing the system with a minimum of 16 million nodes. SNAP-2 will feature multi-user supports, intelligent I/O, etc. One of the significant features in SNAP-2 is the introduction of a programmable marker propagation rules. This feature allows users to define their own and more sophisticated marker propagation rules.

In summary, we have shown that the SNAP architecture can be a useful development platform for high performance and large-scale natural language processing. This has been empirically demonstrated using SNAP-1. SNAP-2 is expected to explore opportunities of massively parallel natural language processing.

# References


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

[Brachman and Schmolze, 1985] Brachman, R. J. and Schmolze, J. G., "An Overview of The KL-ONE Knowledge Representation System," *Cognitive Science* 9, 171-216, August 1985.

[Brooks, 1986] Brooks, R., "A robust layered control system for a mobile robot," *IEEE J. Robotics and Automation,* RA-2, April 14-23,1986.

[Charniak, 1983] Charniak, E., "Passing markers: A theory of contextual influence in language comprehension," *Cognitive Science,* 7(3), 1983.

[Carpenter and Pollard, 1991] Carpenter, B. and Pollard, C., "Inclusion, Disjointness and Choice: The Logic of Linguistic Classification," *Proc. of ACL-91,* 1991.

[EDR, 1988] Japan Electric Dictionary Research Institute, *EDR Electric Dictionaries,* Technical Report, Japan Electric Dictionary Research Institute, 1988.

[Emele and Zajac, 1990] Emele, M. and Zajac, R., "Typed Unification Grammars," *Proc. of Coling-90,* 1990.

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

[Evett, et al., 1990] Evett, M., Hendler, J., and Spector, L., *PARKA: Parallel Knowledge Representation on the Connection Machine,* UMIACS-TR-90-22, University of Maryland, 1990.

[Hendler, 1988] Hendler, J., *Integrating Marker-Passing and Problem-Solving,* Lawrence Erlbaum Associates, 1988.

[Hirst, 1986] Hirst, G., *Semantic Interpretation and the Resolution of Ambiguity,* Cambridge University Press, Cambridge, 1986.

[Jacobs, 1991] Jacobs, P., "Integrating Language and Meaning," Sowa, J. (Ed.) *Principles of Semantic Networks,* Morgan Kaufmann, 1991.

[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.

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

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

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

[Kitano et al., 1991a] Kitano, H., Hendler, J., Higuchi, T., Moldovan, D., and Waltz, D., "Massively Parallel Artificial Intelligence," *Proc. of IJCAI-91,* 1991.

[Kitano et al., 1991b] Kitano, H., Moldovan, D., and Cha, S., "High Performance Natural Language Processing on Semantic Network Array Processor," Proc. *of IJCAI-91,* 1991.

[Kitano, 1990] 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.

[Kitano et al., 1989] Kitano, 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.

[Lenat and Guha, 1990] Lenat, D.B. and Guha, R.V., *Building Large Knowledge-Based Systems,* Addison-Wesley, 1990.

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

[Neal and Shapiro, 1987] Neal, J. and Shapiro, S., "Knowledge-Based Parsing," Bolc, L., (Ed.) *Natural Language Parsing Systems,* Springer-Verlag, 1987.

[Goodman and Nirenburg, 1991] Goodman, K., and Nirenburg, S., *Knowledge-Based Machine Translation Project: A Case Study,* Morgan Kaufmann, 1991.

[Norvig, 1986] Norvig, P., *Unified Theory of Inference for Text Understanding,* Ph.D. Thesis, University of California Berkeley, 1986.

[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.

[Sowa, 1991] Sowa, J. F. (Ed.), *Principles of Semantic Networks,* Morgan Kaufmann, 1991.

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

[Stanfill and Waltz, 1986] Stanfill, 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 Machines Corp., 1989] Thinking Machines Corp., *Model CM-2 Technical Summary,* Technical Report TR-89-1, 1989.

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

[Waltz and Pollack, 1985] Waltz, D.L. and Pollack, J., "Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation" *Cognitive Science,* 9(1): 51-74, 1985.

[Wilensky, 1987] Wilensky, R., "Some Problems and Proposals for Knowledge Representation", Technical Report UCB/CSD 87/351, University of California, Berkeley, Computer Science Division, 1987.

[Zajac, 1989] Zajac, R., "A Transfer Model Using a Typed Feature Structure Rewriting System with Inheritance," *Proc. of ACL-89,* 1989.