Parallel Classification for Knowledge Representation on SNAP [1]

Technical Report No. CENG 89-34

Juntae Kim and Dan Moldovan

Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089-0781

January 8, 1990

[1]This research has been funded by the National Science Foundation Grant No. MIP 8902426.

Abstract

Classification is an important procedure for the knowledge representation systems which employ a frame-based description language to represent terminological knowledge. In practical knowledge processing systems very large knowledge bases are needed and the classification on such a large system becomes very time consuming. The processing time grows rapidly as the size of knowledge base grows. In this report, a parallel classification algorithm for the Semantic Network Array Processor (SNAP) is presented. The algorithm uses SNAP's marker passing capability. The time complexities of sequential and parallel classification are compared. Simulation of the parallel classification algorithm was performed using the SNAP simulator. The simulation results and the effect of various factors on the execution time is discussed.

Keywords : Knowledge processing, classification, parallel algorithm, marker-passing architecture.

Contents

1 Classification Problem

1.1 Introduction

Classification is an important procedure for the knowledge acquisition and reasoning on hierarchical knowledge bases in many kinds of knowledge representation systems.

In most hierarchical knowledge representation systems, the subsumption relation constructs a partial ordering on the knowledge base. The process of constructing a concept taxonomy in which more general concepts are located above more specific ones is the classification process. The classification procedure simplifies the task of creating static knowledge bases and performs a class of inference on concept hierarchy which is very useful for many AI applications.

Although the classification process plays a central role in knowledge representation systems, the processing time increases rapidly as the size of the knowledge base increases. This may cause problems when dealing with practical knowledge processing systems which generally have very large knowledge bases.

The classification procedure consists of steps which determine subsumption relationship between two concepts and indentification of the most specific subsumer among all subsumers of a concept. It is shown, however, that any frame-based description language with reasonable expressive power implies the intractability (co-NP-complete) of complete subsumption decision [Brachman and Levesque 84]. We can reduce the expressiveness of representation or use incomplete algorithm (often used in practical systems) to achieve tractability of algorithm.

Even if we choose one of them, subsumption decision is still a time con-

suming procedure. Moreover, to maintain consistency of taxonomy, classification must be performed on each change on taxonomy, which requires $O(m^2)$ subsumption processes to be performed, where m is the number of concepts in the knowledge base. As the size of the knowledge base grows, time for classification may become unreasonably large. In this report, we present a parallel classification algorithm which has time complexity of $O(log\ m)$ using SNAP's marker passing capability.

1.2 Reasoning on concept taxonomy

In many knowledge representation systems, the generalization relation between concepts and superconcepts forms an inheritance hierarchy or taxonomy. Concepts are connected to their superconcepts (subsuming concepts) by the *is-a* or *subsumption* links. All the concepts in the knowledge base are organized hierarchically. More general concepts are placed above less general concepts, and the properties are inherited through the *subsumption* link from the superconcepts to their subconcepts.

There are two kinds of basic reasoning processes which can be performed on concepts taxonomy - *inference* and *recognition*. Inferencing means that one can infer properties of a concept based on the properties of its superconcept using inheritance property. Recognition is the process of finding a concept which is matched to a given description consisting of a set of properties. Recognition also may use inheritance property since the properties of a concepts are not necessarily available at that concept. These kinds of reasoning on concepts taxonomy are called "terminological reasonings".

One important characteristic of a terminological knowledge base is that there does not exist only explicit or given relations, but also implicit relations between concepts as shown in the following example.

```
A parent is a person who has child.
```

```
A grandparent is a person who has child who is a parent.
```

There is no explicit representation that grandparent is a parent, but we know that parent is a grandparent. i.e., the concept "parent" is a superconcept of "grandparent". Obviously, in order to maintain correct taxonomy, we need some kind of reasoning process with which we can find out the implied relations. These are decisions of the subsumption relation between two concepts and searching for Most Specific Subsumer (MSS) of a given concept.

1.3 Subsumption and classification

The subsumption relation between two concepts is defined such that concept C subsumes concept C' only if the set denoted by C necessarily includes the set denoted by C'. The subsumption algorithm described in [Schmolze and Lipkis 83] performs piece by piece comparison of the properties of C with those of C' to determine subsumption relation as follows.

- All primitive concepts that subsumes C also subsumes C'.
- For each roleset of C, some roleset of C' denotes the same relation.
- The value description of C's roleset subsumes that of C''s roleset.

When the subsumption relation is not explicitly present, we must decide whether one concept subsumes the other or not. Even if there is a given superconcept, we must find out the MSS of a given concept to maintain consistency and monotonicity of the concept taxonomy.

Classification is a process which discovers the implicit subsumption relation between concepts and establishes a subsumption link between them. When a new concept is given with a set of properties, the classification finds out the MSS and places the concept at the proper location in the concept

taxonomy. The basic steps of classification are :

1. Comparing the properties of new concept with those of other concepts in taxonomy, to find all superconcepts (subsumers).
2. Searching for MSS among those subsumers and making a subsumption link between the MSS and new concept.

As an example, consider following simple concept hierarchy of artist in figure 1-1.

```
artist     : is-a       person
             profession art
musician   : is-a       artist
             deal-with  musical-inst.
pianist    : is-a       artist
             play       piano
violinist  : is-a       artist
             play       violin

keyboard-inst.  : is-a musical-inst.
string-inst.    : is-a musical-inst.
piano           : is-a keyboard-inst.
violin          : is-a string-inst.
play            : is-a deal-with
```

Figure 1-1 Example concept hierarchy.

This concept hierarchy is also shown in figure 1-4.

We want to classify a new concept : `A person who is professional on art and play string-inst.` The name of this concept is `string-player`. The description of this concept is in figure 1-2 and shown in figure 1-3.

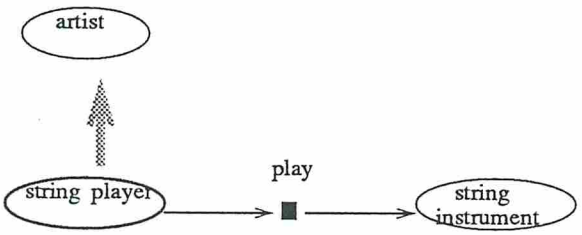

Figure 1-3. Description of string-player

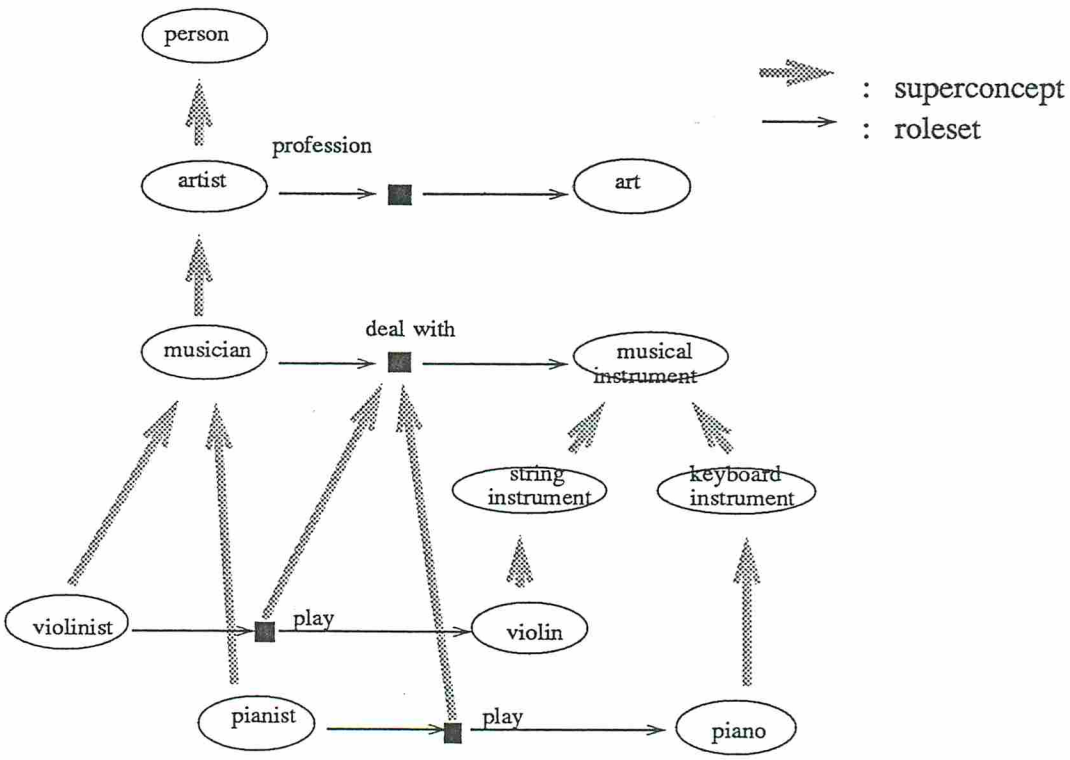

Figure 1-4 Before classification

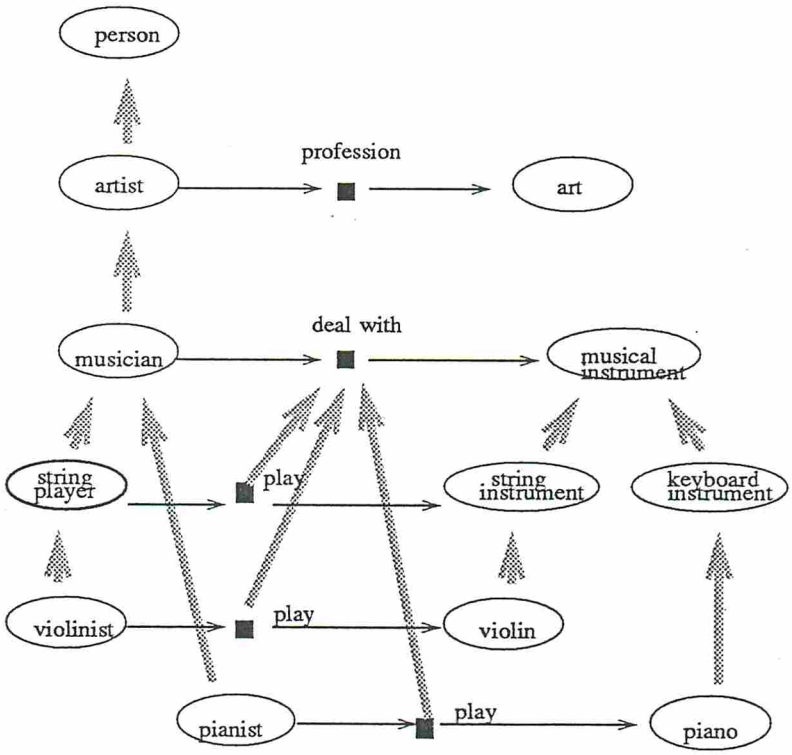

Figure 1-5 After classification

```
string-player : is-a artist
                play string-inst.
```

Figure 1-2 Concept `string-player`

This new concept must be located below the concept `musician` and above the concept `violinist`. After the classification process, `musician` is found as a MSS of `string-player` and the resulting concept hierarchy is shown in figure 1-5.

1.4 Time complexity of classification

It was shown that any frame-based description language with reasonable expressive power implies the intractability (co-NP-complete) of complete subsumption algorithm [Brachman and Levesque 84]. The tractability of algorithm may be achieved by reducing the expressiveness of language or by using incomplete algorithm [Nebel 88] [MacGregor 89].

Even if we choose one of them, the subsumption test is still time consuming. Brachman and Levesque showed that the subsumption test between two concepts for restricted description language takes $O(n^2)$ computation time where n is the length of the description [Brachman and Levesque 84]. Furthermore, for classification of one concept on the taxonomy of M concepts needs $O(M)$ subsumption tests. Therefore, the overall time complexity of classification on a sequential machine will be $O(Mn^2)$ where M is the number of concepts in the taxonomy and n is the average length of description of one concept.

Since the classification process plays a central role in knowledge processing systems, the above time complexity may cause problems as the size of

the knowledge base grows. This becomes critical, especially in real-time applications such as natural language processing or speech translation.

In the next section, a parallel marker passing architecture called SNAP (Semantic Network Array Processor) will be described. In section 3, we present a parallel classification algorithm for SNAP which achieves $O(logM)$ performance.

2 SNAP : a marker passing architecture

2.1 Marker passing architecture

Since the first computational model of spreading activation mechanism was introduced by M. Quillian [Quillian 69], various models of marker passing have been developed for knowledge processing and natural language understanding [Fahlman 79] [Charniak 83] [Hendler 88]. SNAP (Semantic Network Array Processor) [Moldovan 89] is one of these models. It is a highly parallel, marker propagation architecture targeted to AI applications, specifically to natural language understanding. SNAP has many powerful instructions for marker passing. SNAP uses multiple markers to perform inferencing on the knowledge bases similar to KL-ONE [Brachman and Schmolze 85] family knowledge representation. In this section, the knowledge representation schemes of SNAP and the instructions for marker passing are introduced.

2.2 Knowledge representation on SNAP

The knowledge representation of SNAP is a form of semantic network and the knowledge base is distributed over the SNAP array. Each concept is mapped directly into the nodes and the nodes are connected by some relations. Concepts are stored hierarchically using *superconcept*(subsumption) relation.

Since direct mapping of the relations in semantic network needs a large amount of memory to store the names of the relation pointers, relations are implemented not only as a pointer. Relations are implemented by mapping a relation to a *relation node* and linking them with the *concept node* using 64 *primitive relations*. The following primitive relation conventions have been adopted. The others are user definable.

- Superconcept : a relation between a concept and its subsumer.
- Individual : a relation between a type and a individual.
- Split : a relation to separate exclusive types.
- Roleset : a relation to show the properties of a concept.
- Cancel : a relation to be used when an exception exist.

2.3 Instructions and marker propagation rules

A set of powerful instructions specific to knowledge processing are implemented directly in hardware. The instructions that SNAP supports are :

- Create and delete nodes and relations.
- Search a node and set or clear marker.
- Collect marked nodes.

- Propagate makers according to given propagation rule.
- Logical operations on markers.
- Arithmetic operations on markers.
- Some register operations.

There are also several marker propagation rules that govern how markers are passed. Markers are passed from one node to others via relations and they can travel through several relation types at the same time. Multiple markers can be propagated through the network simultaneously. The propagation rules have the format *rule*, *relation 1* and *relation 2* where *relation 1* and *relation 2* are the relations affected by *rule*. The following propagation rules have been defined for SNAP.

- *Seq(r1,r2)* : The *Seq*uence 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*ine rule allows the marker to propagate to all *r1* and *r2* links without limitation.

The instructions and marker propagation rule described above are useful for processing knowledge and some kinds of inferencing. The parallel classification algorithm described in the next section is based on these capabilities of SNAP.

3 Parallel Classification Algorithm

3.1 The algorithm

In this section we introduce a new approach to the classification process - parallel classification using marker passing. The algorithm consists of several inner loops for injecting markers through the semantic network array in which the knowledge base is distributed. It also uses marker propagation rules and logical instructions on markers to calculate the subsumption. Input of the algorithm is a description of a concept, and the output of the algorithm is a concept in the existing knowledge base which is a Most Specific Subsumer of the given concept.

Although this algorithm is based on the SNAP instructions, it can also be used in Connection Machine [Hillis 85] or other parallel machines. However, for efficient processing, simultaneous propagation of multiple markers are essential and, in that sense, SNAP is appropriate for processing the parallel classification algorithm.

SNAP architecture has the capability of passing multiple markers, which is extremely useful for many AI functions. The speed of parallel classification algorithm is achieved from the multiple marker propagation and parallel search capability of SNAP. We can find subsumers (superconcepts) of a new concept not by sequential comparison, but by parallel searching. Finding MSS among the subsumers can also be done in parallel using marker passing.

The main steps of parallel classification algorithm are :

Step 1. To find out all subsumers of input concept by parallel subsumption test against all concepts.

Step 2. To find out MSS (Most Specific Subsumer) of input concept

by parallel searching.

Each step can be done in parallel for all concepts in the taxonomy.

3.2 Parallel subsumption test against all concepts

As shown in the previous section, when a new concept is given with a set of properties (rolesets), we need to perform n subsumption tests against each concept in the taxonomy. In SNAP, however, we do not need to perform subsumption tests concept by concept. Rather, we perform the test simultaneously against all concepts in taxonomy.

Let C be the new concept which has $S_1, S_2, ...S_n$ superconcepts, and has roleset relations $R_1, R_2, ...R_m$ with value description $V_1, V_2, ...V_l$.

Let $Sup(C)$ be the set of all superconcepts of C, $SR(C)$ be the set of all superconcepts of roleset relation of C and $SV(C)$ be the set of all superconcepts of value restriction of C's rolesets.

Let T be the taxonomy on which the concept C will be classified.

Then the algorithm for finding all subsumers of C is as follows. (f- and r- means forward and reverse direction of marker propagation on each relation link)

Phase 1.

For some concept C' to be a subsumer of C, all the superconcept that subsumes C' must also subsume C. i.e, $Sup(C') \subseteq Sup(C)$. In *Phase 1*, we filter out those concepts which violate this condition.

While *Si exists* **do**

set *sub-marker* to node *Si*;
set *ind-marker* to node *Si*;
propagate *sub-marker* :propagation rule *spread (f-sub)*;
propagate *ind-marker* :propagation rule *spread (r-sub)*;

Wait until end of propagation.

For all nodes

if *(not (or sub-marker ind-marker))* **set** *cancel-marker*;
propagate *cancel-marker* :propagation rule *spread (r-sub)*;

Wait until end of propagation.

For all nodes

if *(exist cancel-marker)* **delete** *ind-marker*;

After *Phase 1*, all the nodes which cannot be the subsumer of C are filtered out. In other words, at the end of *Phase 1*, all the nodes marked by ind-markers are the possible subsumers of C.

Phase 2.

Among those concepts marked in *Phase 1*, some concepts which have rolesets that are not in *SR(C)* must also be filtered. Those concepts which have value restriction that are not in *SV(C)* have to be filtered out too. To do this we first mark all the nodes which are members of the sets *SR(C)* or *SV(C)*. Since there are rolesets not only explicitly given by description, but also implicitly given as rolesets of given superconcepts, we must also propagate markers from superconcepts of *C*. By doing this, all the rolesets which must be inherited by concept *C* can be marked.

While *Si exists* **do**

set *sub-marker* to node *Si*;

propagate *sub-marker* :propagation rule *comb (f-sub, f-role)*;

While *Ri exists* **do**

set *sub-marker* to node *Ri*;

propagate *sub-marker* :propagation rule *spread (f-sub)*;

While *Vi exists* **do**

set *sub-marker* to node *Vi*;

propagate *sub-marker* :propagation rule *spread (f-sub)*;

Wait until end of propagation.

After *Phase 2*, all the nodes marked by sub-marker are one of the following :

1. Explicitly given superconcepts of C or their subsumer.
2. Explicitly given rolesets and value restrictions of C or their subsumer.
3. Rolesets and value restrictions which are implicitly given and found by inference as those of concept C. These are properties inherited from superconcepts of C.

Phase 3.

In this Phase, we filter out those concepts which have SR or SV such that $SR \not\subseteq SR(C)$ or $SV \not\subseteq SV(C)$ by propagating cancel-marker.

For all nodes

if *(not (or sub-marker ind-marker))* **set** *cancel-marker*;

propagate *cancel-marker* :propagation rule *comb (r-sub, r-role)*;

Wait until end of propagation.

For all nodes
    if *(exist cancel-marker)* **delete** *ind-marker*;

After *Phase 3*, the remaining nodes marked by ind-marker are actual subsumers of concept C.

This algorithm performs subsumption tests between concept C and all concepts in the taxonomy in parallel, and can finds out all the subsumers of C from T.

3.3 Parallel searching for MSS

After finding out all subsumers of C from T, we must search for Most Specific Subsumer (MSS) of C to properly place C in the hierarchy. In sequential machine, this procedure takes $O(n)$ time, where n is the number of subsumers of C. In SNAP, however, this procedure can be done in constant time by propagating cancel-marker one step. The algorithm is as follows.

For all nodes
    if *(exist ind-marker)* **set** *cancel-marker*;
    propagate *cancel-marker* :propagation rule *seq (f-sub)*;

Wait until end of propagation.

For all nodes

if *(exist cancel-marker)* **delete** *ind-marker*;

After this procedure is done, only those concepts remain which are located at the lowest level of subsumer-hierarchy. These concepts are Most Specific Subsumer (MSS) of the concept C.

3.4 An example

In this section, we will show the operations of above algorithm by tracing markers on each node by using a simple example shown in section 2. The concept description of `string-player` in Figure 1-2 and Figure 1-3 is the input of the classification, and the concept hierarchy shown in Figure 1-1 and 1-4 is the knowledge base. Figure 3-1 shows the propagation of *sub-marker* and *ind-marker* and Figure 3-2 shows the propagation of *cancel-marker* to find out the subsumers of `string-player`.

In this example, $C =$ `string-player` and according to the description of `string-player`, $R =$ `play`, $V =$ `string-instrument` and $S =$ `artist`.

In *Phase 1*, the *sub-marker* and *ind-marker* are propagated from the node `artist` along the *subsumption* link, and the nodes `person`, `artist`, `musician`, `violinist` and `pianist` are all marked. The canceling procedure is not shown here since there is no multiple inheritance in this example. In *Phase 2*, the *sub-markers* are propagated from the `artist`(S), `play`(R) and `string-instrument`(V) along the *subsumption* link and *roleset* link. After *Phase 2*, all the superconcepts and the roleset relations of concept `string-player` , including inherited properties, are marked. In this example, those marked nodes are *Sup(C)* = { `artist` }, *SR(C)* = { `profession`, `play`, `deal-with` } and *SV(C)* = { `art`, `string-instrument`, `musical-instrument` }. Figure 3-1 shows the marker propagation during *Phase 1* and

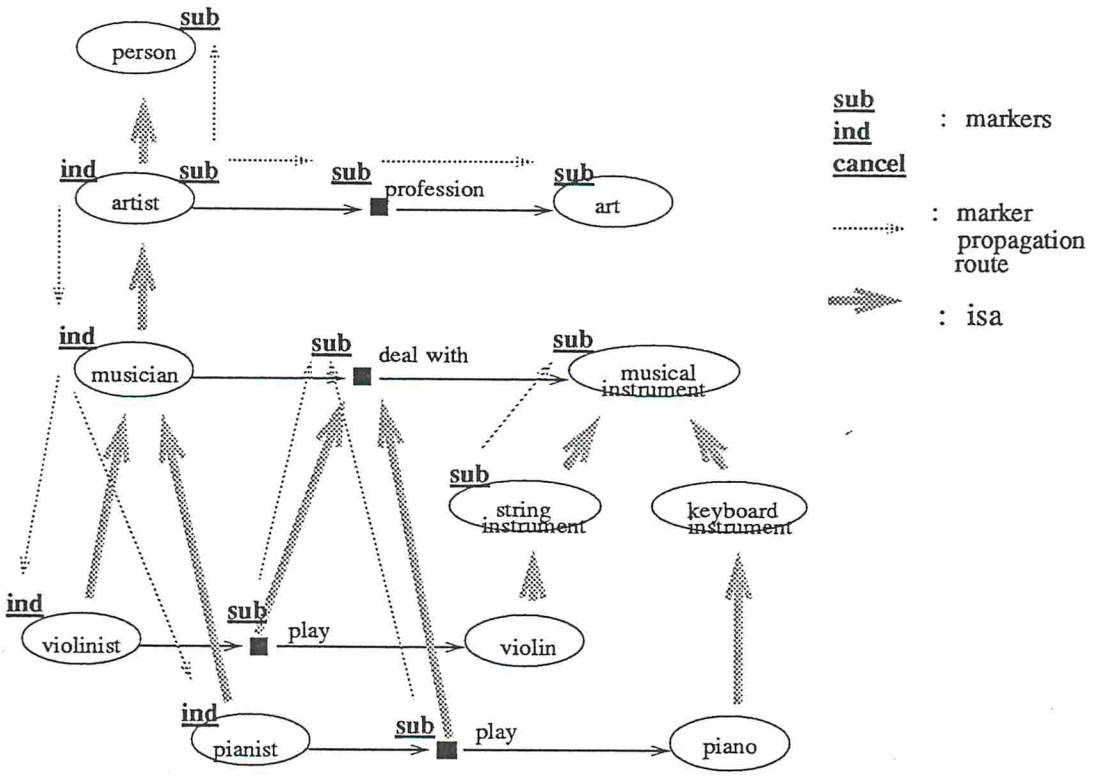

Figure 3-1 Marking sub-marker and ind-marker

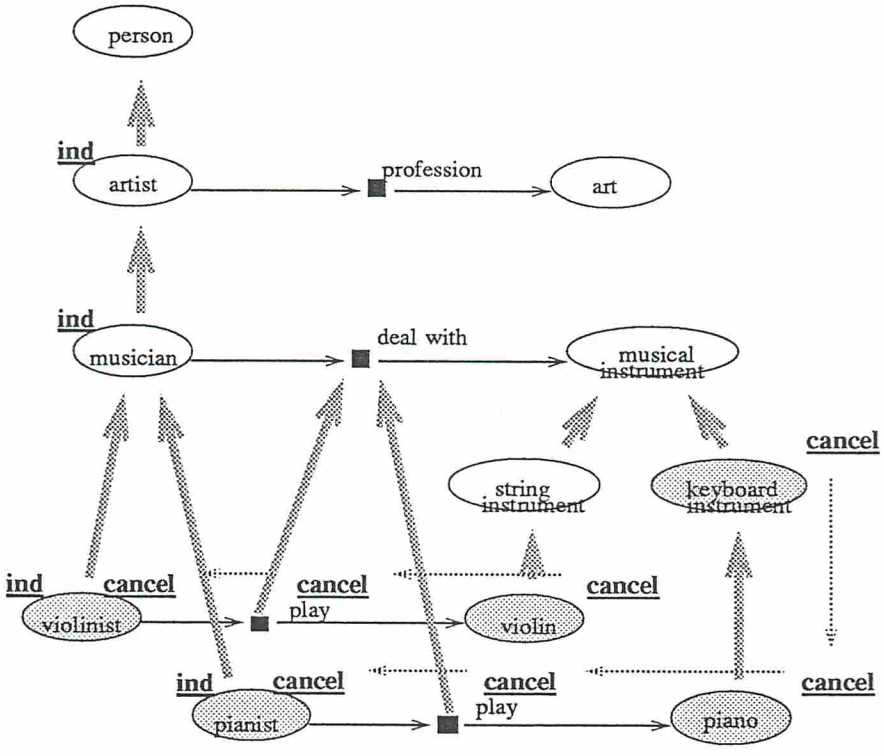

Figure 3-2 Canceling with cancel-marker

Phase 2.

In *Phase 3*, the *cancel-marker* is set to all the nodes which are not marked with *ind-marker* or *sub-marker*, and propagated along the *subsumption* link and *roleset* link in reverse direction. In our example, the *ind-markers* at `violinist` and `pianist` nodes are canceled by the *cancel-marker* since 'play violin' or 'play piano' is not the property of `string-player`. Finally, there remains the `artist` and the `musician`, and they are the subsumers of concept `string-player`. The marker propagation in *Phase 3* is shown in figure 3-2.

After finding out all subsumers, we must search for MSS among those subsumers. To find out the MSS, *cancel-marker* is set to the subsumers and propagated one step up through the *subsumption* link. By doing this, all the subsumers except the lowest level node are cancelled. In our example, the *cancel-marker* is propagated from the `artist` and the `musician`. The marker from `musician` cancels `artist`, and finally the `musician` is identified as a MSS of the `string-player`.

3.5 Time complexity consideration

In section 2, we explained the general subsumption test procedure and the classification algorithm and analyzed their time complexities. It was shown that the subsumption test takes $O(n^2)$ time where n is the number of properties of the concept, and the overall classification algorithm takes $O(mn^2)$ time where m is the number of concepts in the taxonomy.

In order to analyze the time complexity of parallel classification algorithm, the following parameters are defined :

- F_{in} : Average number of fan-in for a concept in hierarchy. (average number of direct superconcepts for one concept)

- F_{out} : Average number of fan-out for a concept in hierarchy. (average number of direct subconcepts for one concept)
- R_{ave} : Average number of roleset relations for one concept which is explicitly given (locally in the inheritance hierarchy).
- L_{ave} : Average length of pattern for one roleset relation. This is the length of longest chain through which markers must propagate to perform inferencing of property inheritance.
- D_{ave} : Average depth of the concept hierarchy.
- M : Total number of concepts in the hierarchy.

With these parameters, we can represent the time for our algorithm as follows

In *Phase 1* of the algorithm, we must propagate markers through all the superconcepts and subconcepts of a given concept. Therefore, it takes $O(F_{out}D_{ave}+F_{in}D_{ave})$ time to finish the propagation. Generally, F_{out} is much greater than F_{in}, so we can simplify the time for *Phase 1* as $O(F_{out}D_{ave})$. In *Phase 2* and *Phase 3*, markers are propagated through subsumption link and roleset link together. Therefore the propagation time is $O((F_{in}+R_{ave})L_{ave})$ and it also can be simplified to $O(R_{ave}L_{ave})$ as above. Therefore the overall time for the algorithm is $O(F_{out}D_{ave}+R_{ave}L_{ave})$.

But the F_{out} and R_{ave} factor only affect the marker injecting time which is relatively small and negligible. Moreover, those factors are not related to the number of concepts in the taxonomy. In other words, F_{out} and R_{ave} do not grow as the size of knowledge base grows, and these factors can be regarded as constants. Furthermore, if we assume that most roleset relations are not transitive (this is generally true. i.e, A like B and B like C do not imply A like C.), L_{ave} is strictly related to D_{ave} and it's less than D_{ave}. Therefore, the time complexity of the algorithm can be simplified as $O(D_{ave})$.

Since we want to know the time complexity of the classification as a function of the size of the knowledge base, D_{ave} must be replaced by a function of M (the number of concepts in taxonomy). Generally, the concept taxonomy has a tree-like structure, and the average depth of the tree can be represented as $log\ N$ where N is the number of nodes in the tree and the base of logarithm is branching factor. Therefore, the time complexity of the parallel classification algorithm as the function of the size of taxonomy is finally $O(log_{F_{out}} M)$.

We have already shown that the sequential classification needs $O(M)$ subsumption test which needs $O({R'_{ave}}^2)$ where R'_{ave} represents the number of overall roleset relations for one concept. This can be calculated as $D_{ave}R_{ave}$ $= (logM)R_{ave}$. Therefore the overall time complexity of sequential classification is $O(M(logM)^2{R_{ave}}^2)$. Thus, the speed up factor achieved by this parallel algorithm is $O(M(logM){R_{ave}}^2)$. Even if we regard R_{ave} as a constant, the speed up factor is $O(M(logM))$ and the parallel algorithm seems to greatly improve the performance as the knowledge base size grows.

4 Simulation Results

In this section we present simulation results of the parallel classification algorithm when parameters vary. An arbitrary size of knowledge base has been generated with different F_{out} and used for simulations. The structure of this knowledge base used in the simulation is similar to the one shown in the previous example which has tree-like pattern. The multiple inheritance case is not considered for simplicity of simulation.

Simulation of the algorithm was done on the SNAP simulator which has been developed by the SNAP research group at USC. The simulator is now implemented on SUN 3/280 using SUN Common LISP [Lin and Moldovan

89]. The structure of SNAP array used in this simulation is :

- Number of chips : 16 chips (4 x 4). Chips in lows and columns are interconnected by bus.
- Number of nodes : 64 nodes / chip.

4.1 The effect of F_{out}

The effect of fan-out (average number of direct subconcepts for one concept) on the classification time was simulated with fixed number of roleset relations and fixed size of knowledge base. The simulation result with knowledge base size of 256 concepts and 1 roleset relations for each concept is shown in figure 4-1 for F_{out} of 2 to 8. The graph shows that F_{out} does not significantly affect the computation time as we mentioned in the previous section. Rather, the computation time decreases slightly as F_{out} increases. This is because the depth of the taxonomy decreases as F_{out} increases when the number of concepts in the knowledge base is fixed.

4.2 The effect of knowledge base size

Figure 4-2 shows the result of simulation with various sizes of a knowledge base. The fan-out was fixed to 2, and the number of roleset relation (locally presented) was fixed to 1 to facilitate the generation of example taxonomy. As shown in the figure, the computation time increases almost logarithmically when the knowledge base size is small. As the knowledge base size grows, the computation time becomes slightly larger than expected time. This is caused by the communication overhead. We have determined that the communication problem is not caused by the interchip interconnection

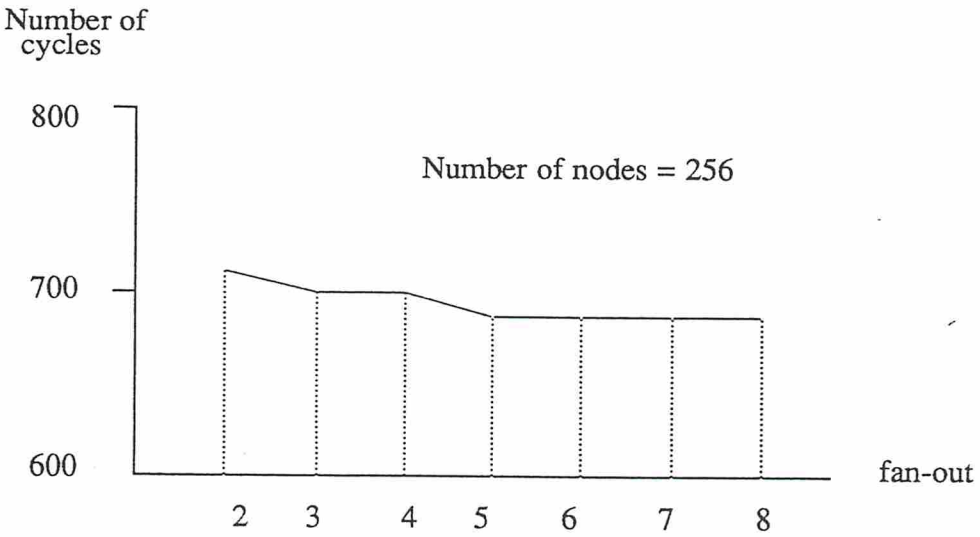

Figure 4-1 Effect of fan-out

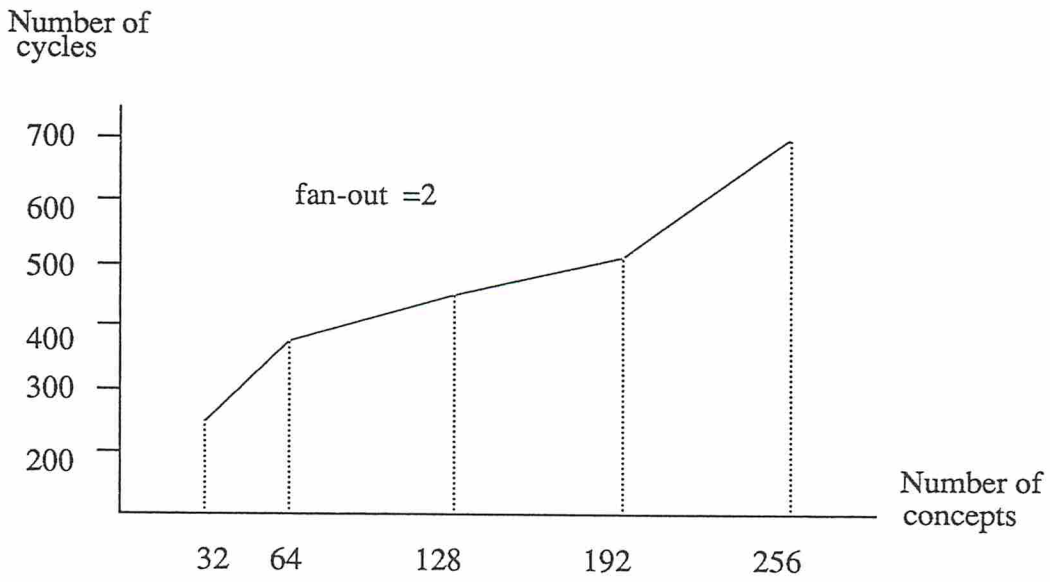

Figure 4-2 Effect of knowledge base size

network, but by the number of nodes inside the chip which need to share the same communication ports. Currently we are performing more simulations to determine the optimal number of nodes per chip. Also, more efficient allocation scheme will reduce this communication overhead.

5 Conclusion

Classification is an important procedure for knowledge representation, knowledge acquisition and reasoning. But the processing time for classification grows rapidly as the size of knowledge base increases, and may become impractical, especially for real-time applications such as natural language speech processing. In this report we described the parallel classification algorithm and its simulation results.

Our parellel classification algorithm is tailored for the SNAP architecture and its knowledge representation scheme. However, it can be implemented on other parallel machine such as CM, and can also be expanded for other knowledge representation system. SNAP is a massively parallel array processor designed for knowledge processing. It supports multiple marker passing and associative processing on the knowledge base which is distributed in the array. By using these capabilities of SNAP, the classification of a concept can be performed in $O(log_{F_{out}} M)$ time, where M represents total number of concepts in the knowledge base and F_{out} represents average number of fan-out for a concept.

Simulation result shows that the processing time of parallel classification is not affected by the branching factor. It is only affected by the size of knowledge base. When the size of knowledge base is large, the computation time becomes slightly larger than expected time due to the communication overhead. This can be reduced by using less node inside the chip and using

more efficient allocation scheme. Even in this case the processing time of parallel classification using marker passing architecture achieves high performance compared with the sequential case. A study on the effect of different array structures and allocation schemes on the processing time will be a direction of future researches.

References

[Brachman and Levesque 84] R. J. Brachman and H. J. Levesque, "The Tractability of Subsumption in Frame-Based Description Languages." in *Proceedings of AAAI-84, The National Conference on Artificial Intelligence*, August 1984.

[Brachman and Schmolze 85] R. J. Brachman and J. G. Schmolze, "An Overview of The KL-ONE Knowledge Representation System," *Cognitive Science* 9, 171-216, August 1985.

[Charniak 83] E. Charniak, "Passing Markers: A Theory of Contextual Influence in Language Comprehension", *Cognitive Science* 7, 171-190, 1983.

[Fahlman 79] Fahlman, S. E. *NETL: A System for Representing and Using Real-World Knowledge.*, The MIT Press, Cambridge, MA, 1979.

[Hendler 88] J. A. Hendler, "Integrating Marker-Passing and Problem Solving", Lawerence Erlbaum Associates, Inc., 1988.

[Hillis 85] D. W. Hillis, *The Connection Machine.*, The MIT Press, Cambridge, MA, 1985.

[Lin and Moldovan 89] C. Lin and D. Moldovan, "SNAP Simulator Results", Technical Report CENG 89-11, University of Southern California, Department of EE-Systems, 1989.

[MacGregor 88] R. MacGregor, "A Deductive Pattern Matcher", *AAAI 88, The Seventh National Conference on Artificial Intelligence*, 403-408, 1988.

[MacGregor 89] R. MacGregor, "The Evolving Technology of the KL-ONE

family of Knowledge Representation Systems", *Workshop on Formal Aspects of Semantic Networks*, January 1989.

[Moldovan 89] D. Moldovan, W. Lee, C. Lin, "SNAP: A Marker Propagation Architecture for Knowledge Processing", Technical Report CENG 89-10, University of Southern California, Department of EE-Systems, 1989.

[Nebel 88] B. Nebel, "Computational Complexity of Terminological Reasoning in BACK", *Artificial Intelligence* 34, (3), April 1988

[Schmolze and Lipkis 83] J. G. Schmolze and T. Lipkis, "Classification in the KL-ONE Knowledge Representation System", in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, IJCAI, 1983.

[Quillian 68] Quillian, M. R. "Semantic Memory," *Semantic Information Processing*, M. Minsky(Ed.), 216-270, The MIT press, Cambridge, MA, 1968.

[Woods 81] W. A. Woods, "Research in Knowledge Representation for Natural Language Understanding", Annual report no. 4785, Bolt Berbank and Newman Inc., 1981.