

Massively Parallel Support for Efficient Knowledge Representation

Matthew P. Evett,¹ William A. Andersen,² James A. Hendler³

Department of Computer Science
University of Maryland
College Park, MD 20740
USA

Abstract

PARKA is a frame-based knowledge representation system implemented on the Connection Machine. The system, which provides extremely fast property inheritance inference capabilities, performs fast "recognition" queries of the form "find all frames satisfying m property constraints" in $O(d+m)$ time—proportional only to the depth of the knowledge base (KB), and independent of its size. PARKA's performance is measured in tenths of a second, even for KBs with 100,000+ frames. We show similar results on the Cyc KB, and for queries involving knowledge structure pattern matching in support of case-based planning. With such run-time performance, PARKA is possibly the "fastest knowledge representation system in the world".

1. Introduction

Currently, AI is experiencing a period of soul-searching. Critics contend that the promise of the AI techniques of the 80's has evaporated because those techniques did not deliver. It is not that their formalisms and theory were unacceptable—they worked fine for small, contrived domains. Real-life domains, however, are orders of magnitude larger. Run-time performance of these earlier AI techniques is completely unacceptable for such domains; these techniques were *computationally ineffective*⁴.

Knowledge representation (KR) forms a core component of most AI applications. Thus, the speed of KR systems is a primary contributor to poor run-time performance. For several years, it has been our goal to design a KR system with computationally effective inferencing on knowledge bases large enough to support real-world, commonsense reasoning. We believe such a system will serve as a foothold for the development of realistic AI applications requiring rapid response time. Our system, PARKA, is a symbolic, frame-based KR system that takes advantage of the Connection Machine's (CM) massive parallelism to deliver the run-time performance we have mandated.

In this paper, we discuss the design and implementation of PARKA. Then, we present a short analysis of the expected and observed run-time performance of several different PARKA operations. We argue that PARKA offers computationally effective knowledge representation on realistically large

knowledge bases, and as such, demonstrates that such systems can serve as a basis for computationally effective AI applications in real-world domains.

2. Description of PARKA

PARKA was designed as a general-purpose knowledge base (KB), for use by other AI systems. For PARKA, we chose to use a frame-based symbolic representation paradigm for several reasons. First, frame systems have been used in KR for years, and remain a common paradigm in contemporary AI research. Second, many other representation schemes are merely notational variants ([Fahlman 1979; Brachman & Schmolze 1985] are just two). Our PARKA research and results should be extensible to those paradigms. Third, semantic nets readily lend themselves to the data-level parallelism design so important in efficient parallel implementation.

Most KR systems have just two primary goals: expressiveness and formality. Authors of these systems want to express as many semantic concepts as possible with an unambiguous, rigorous formalism. Run-time efficiency is of secondary importance to these systems.

We designed PARKA, however, with run-time performance as the primary goal of the system. We built only as much expressiveness and formalism into the system as would still allow computationally effective inferencing. Our bias toward run-time performance over general expressiveness derives from our belief in memory-based reasoning⁵ (MBR).

We are strong supporters of MBR paradigms and their importance in future AI work. We believe that memories and memory association form the pillar of most intelligence tasks, sharing this belief with many others ([Hammond 1989; Kettler *et al.*, 1991; Kitano & Higuchi, 1991; Stanfill & Waltz, 1986] are just a few). This idea counters prevailing wisdom of some in the AI community, who emphasize search and rule-based approaches. For most rule-based systems, run-time is at best linear and at worst exponential in the size of the problem, leading to unacceptable run-time performance on realistically large KBs. In contrast, the ability to access quickly a large collection of exemplars of previously categorized episodes in memory is the hallmark of the MBR paradigm. PARKA directly supports exactly this type of fast access to such KBs. This ability more than compensates for PARKA's somewhat restricted expressiveness when compared to that of other, serial, KR systems.

2.1 Design Specifics

PARKA is a basic frame system, with each frame corresponding to a concept represented in the KB and the collection of relations to which it belongs. Relations among

¹Email: evett@cs.umd.edu

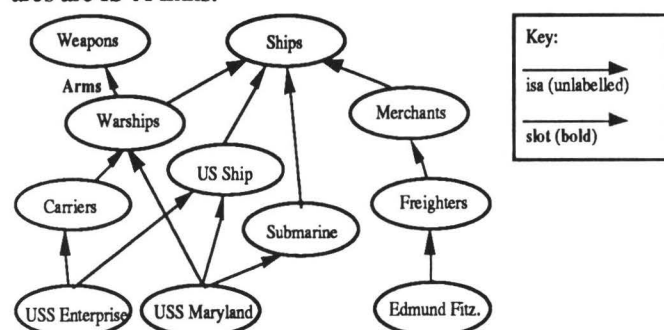
²Email: waander@cs.umd.edu

³Email: hendler@cs.umd.edu

⁴The term was coined by Shastri.

⁵We use the term "MBR" in a broader sense than do Stanfill & Waltz [1986], to include such paradigms as case-based reasoning.

A small subset of a frame network is shown in Figure 1. The frames are shown as ovals. The properties of each frame are represented by the arcs emanating from them. Unlabelled arcs are IS-A links.



2.2 Inheritance Mechanism

PARKA employs a property inheritance mechanism on its IS-A hierarchy. In PARKA, a frame is said to be *explicitly-valued* for a given property if it is incident on a property link of that type—that is, if the frame contains a slot by that name. Any frame not explicitly-valued for a given property *inherits* the value of its nearest ancestor(s) (using a variant of Touretzky's *inferential distance ordering* scheme) that is explicitly-valued for that property. If there is more than one such node (which is possible because PARKA allows multiple inheritance), the inherited value is either ambiguous, or is arbitrarily chosen from among the possible candidates.

The first version of PARKA [Evetts *et al.*, 1993; Evetts & Hendler, 1992] handled multiple-inheritance—as have many previous frame-based KR systems [Brachman & Schmolze, 1985] is just one)—by using inheritance path length to disambiguate multiple inheritance paths. This paradigm suffers from redundancy, ambiguity, and other problems, and has been soundly criticized in the philosophical community as being epistemologically inadequate. Many of these criticisms are detailed in [Touretzky, 1986] and [Brachman, 1985].

To address the problems of multiple inheritance, PARKA uses a top-down credulous inheritance mechanism based on Touretzky's *inferential distance ordering* (IDO) metric to disambiguate multiple inherited values. It works like this: assume the frame in question, X , is not explicitly valued for the given property, A . Let B be the set of ancestors $\{B_1, B_2, \dots\}$ of X that are explicitly valued for A . X takes B_i 's value for A as its own, provided B_i is an element of B such that there is no B_j ($\neq B_i$) such that B_j is an IS- A descendant of B_i . If more than one element of B meets this criterion, X is said

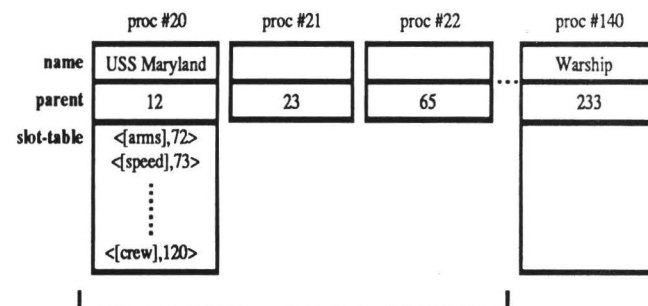
Unfortunately, many retrieval operations involving top-down inheritance mechanisms, including IDO, have been shown to be NP-hard [Selman & Levesque, 1989]. To allow calculation of these operations in a timely manner, we adopted a slightly weaker ordering scheme for inheritance disambiguation. Let B be the set of ancestors of X explicitly valued for property A . X takes B_i 's value for A as its own, where B_i is that element of B with the largest *topological number*. If more than one element of B meets this criterion, X is said to be ambiguously valued for property A . The topological number, $topo(Z)$, of a frame Z , is defined inductively: $Topo(\text{<root node>}) = 0$. For all other frames, Z , $topo(Z) = 1 + \text{the maximum } topo(Y)$, for each Y , a parent of Z . PARKA can calculate inheritance inferences using this disambiguation metric in $O(d)$ time, independent of KB size, where d is the ISA-depth of the frame network. Our topological disambiguation scheme differs from inferential distance ordering because it may arbitrarily disambiguates among two ancestors when neither is an IS-A ancestor of the other. PARKA discovers ambiguity only if the ancestors have the same topological number.

Though PARKA's disambiguation mechanism is not as powerful as complete IDO, it is considerably stronger than a simple path-length based scheme. Notably, it does not suffer from the problems of *redundancy* noted in [Touretzky, 1986]. Also, examination of the PARKA encoding of the Cyc commonsense KB (see Section 3.3) reveals no ambiguous inheritance relations incorrectly disambiguated by PARKA. If Cyc turns out to be exemplary of other, future KBs, then this shortcoming promises to have very little impact on most inferences.

2.3 Implementation

PARKA's internal representation of a frame consists of a block of processors, one for each frame's IS-A parents. These processors are contiguous across the CM's processor address space. One processor of each block is distinguished as a referent for all other frames pointing to the represented frame.

The slots of each frame are encoded in a *slot table*, stored in a processor of that frame's block. The table is a list of pairs (*<property>*, *<property-value>*), each component being a processor address. Figure 2 illustrates the internal representation on the CM of part of the net shown in Figure 1. The large rectangles represent the distinguished processor of each block of processors corresponding to a particular frame. The small rectangles represent the remaining processors of those blocks. [Bracketed] values represent addresses of the processor representing the correspondingly named property.



processor segment for frame "USS Maryland"

Figure 2: Internal CM representation of a small subset of a frame network

To look up the value of a given property, X , for a frame,

Y, PARKA scans through the frame's slot table, looking for an entry corresponding to the given property. If a matching slot table entry is found, the corresponding value stored there is the address of the processor representing the frame that is the value of property X for frame Y. In general, slot retrieval is proportional to the size of the largest slot table in the KB. PARKA maintains a cache of the most recently accessed properties to accelerate this process.

3. Performance

To demonstrate that PARKA can provide computationally effective knowledge representation, we implemented and timed a number of retrieval operations on very large KBs. Tests included inheritance queries on very large pseudo-random KB's, conjunctive queries on the Cyc KB, and structure retrievals on a large KB supporting a case-based planner.

3.1 Inheritance Queries

Almost all KB queries involve some inferencing along the IS-A hierarchy because almost all queries involve having to calculate the inherited value of a set of properties for a set of frames. PARKA uses several data structures to speed such inheritance inferencing, including using multiple processors to represent frames having multiple parents. It uses an activation wave propagation algorithm to calculate the value for a given property of every frame in the KB in time independent of the size of the network.

First, the IS-A root frame is "activated", forming an *activation wave*. At each iteration, this wave is passed downward along IS-A links. The activation wave propagates synchronously; all nodes in the current "wave front" simultaneously activating the incident nodes that have not yet been activated. Each such iteration is a *propagation step*. Because PARKA is implemented on the CM, each propagation step is accomplished with a single parallel operation. In detail, at propagation step i :

- Frames in topological level i explicitly valued for the property set their wave value to k , where k 's high order bits are i (the topological level of the frame that is the origin for this value), and k 's low order bits are the frame's property value.
- Every frame (processor) not explicitly valued for the property in question that has a parent node at topological level $i-1$ "pulls" down the value of the activation wave from those parents.
- Frames at topological level i not explicitly valued choose as their own activation wave value the largest of those pulled down from the parent frames. Because the high order bits of each wave value are the topological level of the origin node, the selected value conforms to the IDO scheme outlined in section 2.2.

The number of propagation steps required to calculate a property value is equivalent to the depth, d , of the network's IS-A hierarchy. Consequently, PARKA's run-time for queries, such as "what things are black?", is $O(d)$, and is independent of the size of the KB. To answer a similar query, most serial KR systems would require $O(b^d)$ time, where b is the average downward fan-out of the IS-A network.

The topologies of most successfully implemented large KBs have been very shallow relative to their breadth. It is commonly believed that this shallowness will persist and be accentuated as net size increases. (We implemented the Cyc commonsense KB in PARKA (see section 3.3), and it does enjoy a similarly shallow IS-A topology.) Consequently, we

expect PARKA's $O(d)$ performance of inheritance inferences to scale up with larger KBs.

To determine how PARKA would compare against a serial representation system, we created a serial version of PARKA, called SPARKA ("serial-PARKA"). To make the comparison of the timings between PARKA and SPARKA as fair as possible, we implemented SPARKA as a severely stripped-down version of a more complete serial implementation (detailed in [Spector *et al*, 1992]). It has very little functionality other than for simple property inheritance calculations, but is optimized to effect those calculations as quickly as possible.

We tested our analytical predictions of PARKA's run-time performance of simple property inheritance queries by timing responses to queries on topologies of varying size and depth. Then, we timed SPARKA on the same queries and networks.

The networks were to be quite large—up to 128K nodes⁶. As encoding the networks by hand was out of the question, we developed a number of different algorithms for generating pseudo-random networks with certain topological characteristics. These techniques are described in [Evetts *et al*, 1993; Evetts & Hendler, 1992]. Our experience with the Cyc KB (see section 3.3) affirmed our belief that the topologies used to measure PARKA's performance are reflective of the topologies of realistic KBs.

Figure 3 shows PARKA's run-time for frame networks of depth 8, and of varying size. These timings isolate the effect of network size on run-time, supporting our supposition that PARKA's computation of inheritance queries is independent of network size⁷. The serial system's run-time, however, was linear with respect to network size. The figure contains best-fit curves to highlight this linear relation.

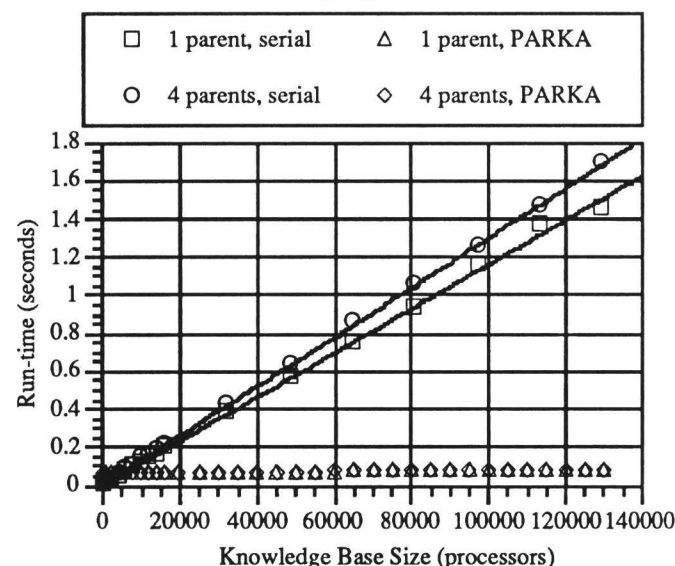


Figure 3: Run-time performance of inheritance queries on 8-level networks of varying sizes.

The networks used in the timings were of two topological

⁶Timings were made on a "quarter" CM-2, consisting of only 16K processors.

⁷Actually, we observed a degree of correlation between PARKA's run-time and the size of the networks. We examined this degradation away from our theoretical performance predictions. The results of this study are detailed in [Evetts & Hendler 1992]. The degradation is completely accounted for by the performance of the CM's interprocessor communication operations. The run-time of these operations degrades proportionally with router network load.

types: trees (each frame having exactly one parent) and directed graphs (each frame having between one and four parents.) We used different types of topologies to demonstrate that PARKA's run-time was independent of upward IS-A fan-out.

A comparison between the serial KR system and PARKA in Figure 3 shows that the latter remains computationally effective even for very large KBs, while the serial system's performance becomes unacceptable for large networks. We fully expect this contrast to become increasingly stark for the much larger KBs undoubtedly required for the large, truly artificially intelligent applications of the future. (Lenat and Guha [1990] suggested that such networks will contain tens of millions of nodes.)

3.2 Recognition Queries on the Cyc KB

The ability to solve *recognition queries* has driven much of PARKA's design. The problem of recognition is well-known in the field of knowledge representation [Fahlman, 1979; Wilensky, 1986; *etc.*] Recognition is the problem of answering KB queries of the form: "find all frames x where $P_1(x, c_1) \wedge P_2(x, c_2) \wedge \dots \wedge P_p(x, c_p)$ ", where $P_i(x, c_i)$ is a unary predicate true for all frames, x , that have value c_i for property P_i . E.g.: "What object is most characterized by this list of property values?" and "what have trunks, tusks, and are big and gray?" Such queries, which are extremely time-consuming for serial-based systems, offer an excellent opportunity to demonstrate the advantages of a parallel system such as PARKA.

PARKA's ability to execute inheritance inferences quickly makes it particularly suitable to recognition. Because PARKA can determine which objects have a given value for a given property in $O(d)$ time, PARKA can determine which objects satisfy a set of p property constraints in no more than $O(dp)$ time, where d is the depth of the net.

In fact, PARKA uses a pipelining technique to evaluate recognition queries in time $O(d+p)$. Because each wave propagation step of an inheritance inference occurs "in lockstep"—all frames at the same topological level calculating their property value simultaneously—pipelining can be added to the basic process outlined in section 2.2. At each propagation step i , all frames at topological level j , such that $j \geq i$, retrieve from their parent(s) the wave activation value having to do with the $(j-i)$ -th element of the set of properties being inferred. The complete propagation requires $d + p - 1$ propagation steps.

PARKA's $O(d+p)$ performance compares very favorably with recognition queries on serial systems, which require $O(pb^d)$ time for the same queries, where b is the average fan-out of the IS-A hierarchy. We tested these predictions by timing PARKA's performance for recognition queries containing a varying number of conjuncts on an implementation of the Cyc KB. The results appear in Figure 4, and support our run-time analysis. We discuss these results in the next section.

3.3 Using Cyc for Validation of PARKA

Our motivation for using the Cyc KB to evaluate PARKA's performance is twofold. First, we want to validate PARKA's inference mechanisms on a KB of large size and realistic topology. Because Cyc is the largest and most comprehensive commonsense KB in existence, it is an obvious choice. A second and more exciting motivation is that we envision some future version of Cyc being built on top of a massively parallel substrate, like PARKA, to make its reasoning services fast enough to be used by an intelligent agent operating in the world in real time.

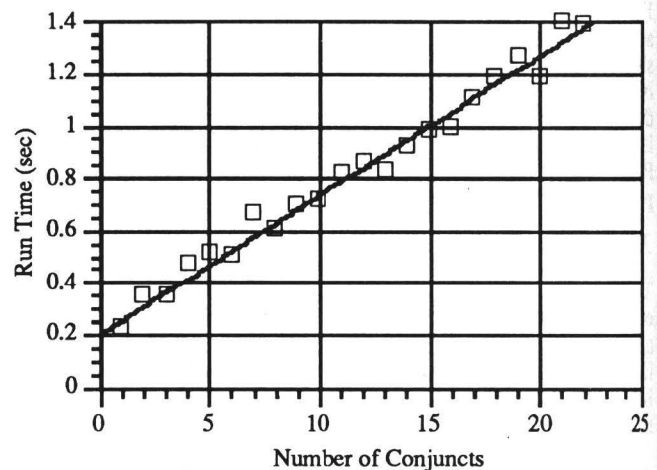


Figure 4: Run-time performance of recognition queries of various sizes on the PARKA implementation of the Cyc KB

On the lowest level, Cyc consists of a frame system (frames are called "units" in Cyc), representing assertions (in the form of binary relations, or slots) about entities in the world. Above that level is the CycL "constraint language", which allows the specification of inferences to be made about units in the KB. The inference mechanisms provided by CycL range from the very simple, such as the slot inverse mechanism (e.g. father(John, Mary) $\rightarrow$ fatherOf(Mary, John)), to theorem proving using general "wffs", to unsound inference methods such as analogy.

3.3.1 The Subset of Cyc in PARKA

PARKA implements only some of the inference capabilities provided by CycL, particularly those having to do with property inheritance. We represented only the subset of Cyc involving IS-A based property inheritance and ontologies.

The version of the Cyc KB we used for our tests contained a total of 26214 units, 8591 (33%) of which were collections, and 17623 (67%) instances. Of the instances, 4031 (15% of the total) were slots (slots are explicitly represented in the Cyc ontology). To accommodate a KB of this size, we used a 16,384 processor CM-2 with a virtual processor ratio of 4:1. The maximum depth of the KB along the specs/instances relation was 23; this is the number of steps required for PARKA's wave propagation algorithms to terminate.

3.3.2 Performance of PARKA on the Cyc KB

For our test on the Cyc KB, we tried a series of recognition queries of a type employed by CycL to find in the KB units "similar" to a given unit. A unit is considered "similar" to a given unit if it has the same slot values for some number of slots exceeding a specified threshold value. We chose a Cyc unit (#%Burma-1986) with a relatively large number of local assertions (22), and executed an algorithm to find matching units for the first n slots ($1 \leq n \leq 22$) of that unit. Whereas Cyc's serial algorithm finds only an arbitrary number of matching units, this algorithm exhaustively matches the probe against the entire KB at one time, thus finding *all* appropriate matches. This experiment was designed to show empirically both the $O(d+m)$ complexity of conjunctive recognition queries in PARKA, and to demonstrate that PARKA can be used to provide fast matching for analogy-related functions, a task which traditionally has been difficult for serial systems.

As Figure 4 clearly shows, the time required to perform

recognition queries grows only linearly in the number of conjuncts, and overall performance, even for a query of 22 conjuncts, is excellent. The run-time matches the $O(d+m)$ performance predicted by analysis.

3.4 Structure Retrieval for Case-Based Planning

As another part of our demonstration of PARKA's computational effectiveness as a KR system for very large AI applications, we implemented a general structure retriever mechanism. PARKA's structural retrieval mechanism has been used to support CaPER, a case-based planning system [Kettler *et al.*, 1992]. Concepts and individuals in CaPER's memory are represented using PARKA frames.

Structure retrieval is a generalization of recognition. The problem of structure retrieval is to find the set of all m -tuples $(x_1, x_2, \dots, x_m)$, where x_i are frames, such that $P_1(x_1, x_2) \wedge P_2(x_2, x_3) \wedge \dots \wedge P_{i-1}(x_{i-1}, x_i) \wedge Q_1(x_1, c_1) \wedge \dots \wedge Q_k(x_k, c_k)$, where $P_i(x_i, x_j)$ are binary predicates between specified tuple components, and $Q_j(x_j, c_j)$ are unary predicates on the specified tuple component.

The problem can be viewed as that of finding all subgraphs (the m -tuples) in a KB frame network that match a specified probe graph (the conjunction of binary and unary predicates). The KB is viewed as a graph, where the vertices are frames, and the edges are the 2-tuples comprising the binary relations represented by slots. A probe graph consists of a set of variables, a set of unary constraints, and a set of binary constraints. The unary constraints restrict the domains of the variables. The binary constraints specify relations (in terms of inherited property values, etc.) that must hold between variables. The algorithm returns all subgraphs that simultaneously satisfy all constraints. Note that this is an exact match algorithm. Future implementations will provide inexact matching capabilities with programmable similarity metrics.

PARKA uses the pipelining technique outlined in section 3.2 for the retrieval of all components of the probe graph having to do with property inheritance, set membership, and subclass/superclass membership. The algorithm uses a mechanism similar to the *join* operation of database theory to combine these results.

We have not yet completed a theoretical analysis of the PARKA's run-time performance of structure retrieval, relying so far on empirical evidence to demonstrate PARKA's computational effectiveness for structure retrieval queries. We compared PARKA to a serial KR system on the same KB to demonstrate the advantages provided by the massively parallel PARKA system. For the serial system we used an optimized serial version of PARKA. The serial version was highly-indexed to allow for fast retrievals of all those frames being valued for a given property.

We timed several structure retrieval probes used by CaPER to retrieve exemplar plans from two case bases, consisting of one hundred and of ten plan cases. These KBs involved a domain of automobile assembly plans that were constructed via use of the Nonlin planner [Tate, 1976]. Each case base was encoded as a PARKA knowledge base. The larger case base consisted of over 10,000 frames, requiring over 35,000 processors. In this KB, a structure retrieval probe with 9 unary and 8 binary constraints took approximately .7 seconds of CM time. Future implementations are expected to reduce retrieval times to under 200 milliseconds for comparable probes through the use of more pipelining and caching techniques. The timings for this and several other probes on these KBs are shown in Figure 5.

As can be seen, for some of the queries PARKA experienced

a speed-up of more than 10,000 over the serial system. The run-time for the serial structure retrieval algorithm was much worse than linear in the size of the KB. The run-time of the parallel algorithm, on the other hand, appeared to grow only logarithmically in the size of the KB.

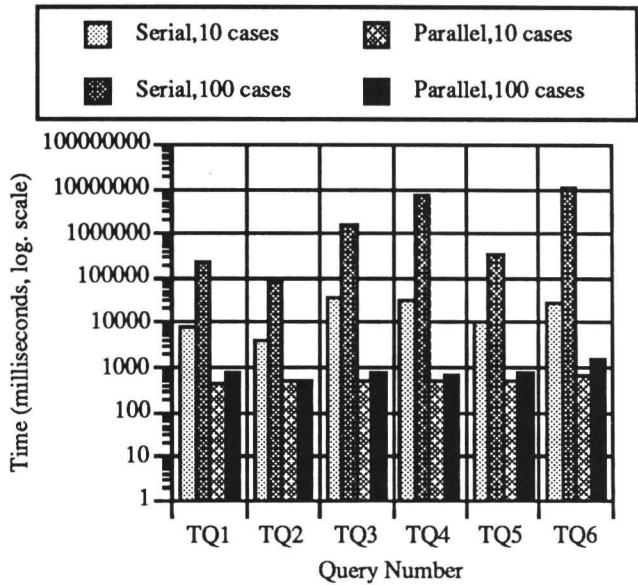


Figure 5: Structure retrieval run-time performance

Because retrieval probes can be specified over any relation in the current KB, this algorithm allows cases to be retrieved on arbitrary features, not restricted to any set of predetermined indexes. Because of this, there is no privileged set of indexing features by which cases can be found in memory. This lack of a need for *a priori* computed indexing features is one of PARKA's strengths. Because of it, PARKA is not hindered from retrieving potentially useful cases that share unindexed features with the target probe (other benefits are discussed in [Kettler *et al.*, 1992].)

We know of only one other massively parallel system that provides something like structure retrieval. DeMara and Kitano describe a generic pattern matching algorithm [Hammond, 1989] running on Moldovan's SNAP system [Moldovan *et al.*, 1989]. The "pattern matching" problem shares some similarities to that of structure retrieval, but is considerably different. The algorithm listed there is for finding matches against 4-node subgraphs, and it does so very quickly due to the special-purpose hardware on which SNAP is implemented. Our algorithm works for arbitrarily specified graphs, and runs on general-purpose hardware.

4. Future work

PARKA was intended as a basis for large AI systems. The implementation of the Cyc KB in PARKA was just the first of what we hope will be a series of uses of PARKA in other large AI systems. Potential uses for PARKA include other case-based AI systems, as the basis of a massively parallel knowledge server, and as part of a knowledge-mining system. In addition, we plan to examine how PARKA might be more fully integrated into the Cyc representation system, proper.

Our immediate plans for PARKA center around completing the core representational capabilities outlined in [14 and 15]. This work will include the implementation of partonomic and set-theoretic representational mechanisms.

Also, we have implemented a simple version of PARKA

on the MIMD CM-5. Preliminary results show that we should be able to represent KBs of over 1M frames on a 1K-processor CM-5 and obtain run-time performance nearly an order of magnitude better than the results in this paper. Because the CM-5 is a MIMD machine (though we use it as a SPMD machine), we plan to use an active messaging scheme [Von Eicken *et al.*, 1992] to increase the flexibility of PARKA's memory association schemes, and to increase the use of pipelining in inferencing.

5. Conclusion

Using KBs with over 100,000 frames, we have shown that PARKA computes several types of KR retrievals in time independent of the size of the KB and dependent only on network depth. The run-time of these operations is in the tenths of seconds. This performance compares very favorably to serial representation systems; for structure retrieval operations we recorded speed-ups of more than 10,000 fold over a highly-optimized serial representation system. Because our empirical evidence so far has supported our analytical claims, we believe that PARKA's performance will scale up to even larger KBs.

We argue that PARKA's performance for these operations has demonstrated that it can offer computationally effective knowledge representation on realistically large KBs. This in turn, demonstrates that massively parallel KR systems can serve as a basis for large AI applications in real-world domains.

Acknowledgments:

We wish to thank the Systems Research Center at the University of Maryland for their support in this research. We also thank the University of Maryland Institute for Academic Computing Services (UMIACS) and Thinking Machines Corp. for the use of their Connection Machines. The support staff at both institutions were very helpful during the development of PARKA.

This work has been supported by AFOSR grant 01-5-28180, ONR grant N0014-88-K-0560, and NSF grant 1R1-8907890.

References:

- [Brachman, 1985] R.J. Brachman. I Lied about the Trees. *AI Mag.* 6, 3 Fall, 1985.
- [Brachman & Schmolze, 1985] R.J. Brachman and J.G. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cog. Sci.*, 9, 2 April-June 1985.
- [DeMara & Kitano, 1992] R. DeMara and H. Kitano. PACE Benchmark for Parallel AI Architectures. Tech. Rep. IMPACT-92-004-USC/CMU, and PKPL 92-5, Department of EE - Systems, University of Southern California, and School of Computer Science, Carnegie Mellon University, 1992.
- [Evetts et al, 1993] M.P. Evett, L. Spector, and J.A. Hendler. Massively Parallel Frame-Based Property Inheritance in PARKA. To appear in *Journal for Parallel and Distributed Computing*, 1993.
- [Evetts & Hendler, 1992a] M.P. Evett and J.A. Hendler. Degradation of Interprocessor Communication Operations on the Connection Machine. Tech. Rep., Department of Computer Science, Univ. Maryland, College Park, November, 1992.
- [Evetts & Hendler, 1992b] M.P. Evett and J.A. Hendler. An Update of PARKA, a Massively Parallel Knowledge Representation System. Tech. Rep., CS-TR-2850, Department of Computer Science, Univ. Maryland, College Park, February, 1992.
- [Fahlman, 1979] S.E. Fahlman. *NETL: A System for Representing and Using Real World Knowledge*. MIT Press, Cambridge, MA, 1979.
- [Hammond, 1989] K. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, 1989.
- [Kettler et al, 1993] B. Kettler, J.A. Hendler, W.A. Andersen, and M.P. Evett. "Massively Parallel Support for a Case-based Planning System." In *Proceedings of the Ninth IEEE Conference on AI Applications*, IEEE 1993.
- [Kitano & Higuchi, 1991] H. Kitano and T. Higuchi. Massively Parallel Memory-Based Parsing. *Proceedings of IJCAI-91*, 1991.
- [Lenat & Guha, 1990] D.B. Lenat and R.V. Guha. *Building Large Knowledge-Based Systems*. Addison Wesley, Reading, Mass., 1990.
- [Moldovan et al, 1989] D. Moldovan, W. Lee, and C. Lin. SNAP: A Marker-Propagation Architecture for Knowledge Processing. Tech. Rep. CENG 89-10, Dept. Electrical Engineering-Systems, University of Southern California, Los Angeles, CA, Sept. 1989.
- [Selman & Levesque, 1989] B. Selman and H. Levesque. The Tractability of Path-Based Inheritance. *Proceedings of IJCAI-89*, Morgan-Kaufman, San Mateo, CA, 1989.
- [Spector et al, 1990] L. Spector, M.P. Evett and J.A. Hendler. Knowledge Representation in PARKA. Tech. Rep. TR-2409, Department of Computer Science, University of Maryland, College Park, MD, Feb. 1990.
- [Spector et al, 1992] L. Spector, W. Andersen, J.A. Hendler, B. Kettler, E. Schwartzman, C. Woods and M.P. Evett. Knowledge Representation in PARKA -- Part 2: Experiments, Analysis, and Enhancements. CS-TR-2837, UMIACS-TR-92-16, Department of Computer Science, Univ. Maryland, January 1992.
- [Stanfill & Waltz, 1986] C. Stanfill and D. Waltz. Toward Memory-Based Reasoning. *Communications of the ACM*, Vol. 29, No. 12, December 1986, pp. 1213-1228.
- [Tate, 1976] A. Tate. Project Planning Using a Hierarchic Non-linear Planner. Research Report No. 25, Department of Artificial Intelligence, University of Edinburgh, 1976.
- [Touretzky, 1986] D.S. Touretzky. *The Mathematics of Inheritance Systems*. Morgan Kaufmann, Los Altos, CA, 1986.
- [Von Eicken et al, 1992] T. Von Eicken, D. Culler, S. Goldstein and K. Schauer. Active Messages: a Mechanism for Integrated Communication and Computation. Tech. Rep. UCB/CSD 92/#675, Computer Science Division, EECS, University of California, Berkeley, CA.
- [Wilensky, 1986] R. Wilensky. Some Problems and Proposals for Knowledge Representation. Tech. Rep. UCB/CSD 86/294, University of California, Berkeley, May 1986.