Massively Parallel Support for Computationally Effective Recognition Queries

Matthew P. Evett[1] **James A. Hendler**,[2] **William A. Andersen**[3]

Department of Computer Science
University of Maryland
College Park, MD 20742

Abstract

PARKA, a frame-based knowledge representation system implemented on the Connection Machine, provides a representation language consisting of concept descriptions (frames) and binary relations on those descriptions (slots). The system is designed explicitly to provide extremely fast property inheritance inference capabilities. PARKA performs fast "recognition" queries of the form "find all frames satisfying p property constraints" in $O(d+p)$ time—proportional only to the depth, d, of the knowledge base (KB), and independent of its size. For conjunctive queries of this type, PARKA's performance is measured in tenths of a second, even for KBs with 100,000+ frames, with similar results for timings on the Cyc KB. Because PARKA's run-time performance is independent of KB size, it promises to scale up to arbitrarily larger domains. With such run-time performance, we believe PARKA is a contender for the title of "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 evaporated because those techniques did not deliver. It wasn't that their formalisms and theory were unacceptable (in fact, they worked fine for relatively small, contrived domains). Real-life domains, however, are orders of magnitude larger, and the run-time performance of these earlier AI techniques is often completely unacceptable for such domains (and even much smaller ones); that is, these techniques, while quite useful, are *computationally ineffective* (Shastri 1986).

The field of knowledge representation (KR) offers many problems for which these classic AI techniques have yielded unacceptably slow performance. One example is *recognition*, the problem of identifying those frames that satisfy a given set of property constraints. For example, "Find all x such that x is yellow, alive, and flies." Existing KR systems efficiently solve the converse problem—retrieving the properties of any given frame—but cannot do the same for recognition. In general, they answer recognition queries by traversing the entire knowledge base (KB), collecting the set of frames satisfying the given constraints. Their run-time for recognition queries is no better than linear in the size of the KB, i.e., $O(n)$.

It has been our goal to design a KR system fast enough to provide computationally effective recognition queries (and other types of queries) on KBs large enough to support real world commonsense reasoning. 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 high run-time performance. It effects recognition queries in time virtually independent of the size of the KB, and dependent only on the KB's depth, d. (For large KB's, usually $d \approx lg(n)$.)

In the remainder of this paper, we discuss the design and implementation of PARKA, and present a short analysis of the expected and observed run-time performance of those operations used in answering recognition queries. To validate PARKA's inference mechanisms on a KB of realistic size and topology, we test PARKA's performance on the *Cyc* KB and argue that the performance shows that PARKA offers computationally effective recognition queries on realistically large KBs.

2. Description of PARKA

PARKA was designed as a general-purpose KB, for use by other AI systems. We chose a frame-based symbolic representation paradigm for two main reasons: first, frame systems have been used in KR for year and remain a common paradigm in contemporary AI research (Sowa 1991; *J.CMA* 1992, e.g.); second, semantic nets readily lend themselves to the data-level parallelism design so important in efficient parallel implementation.

Most KR systems have two primary goals: expressiveness and formality. The authors of these systems want to express as many semantic concepts as possible with an unambiguous, rigorous formalism. For them, run-time efficiency is of secondary importance. They emphasize classic search and rule-based approaches, consequently suffering run-time that is at best linear, and at worst exponential, in the size of the problem. While these methodologies have a place in AI, their application to large KB's leads to unacceptable run-time performance, and it is expected that realistic KB's will be *very* large, indeed—on the order of 10's or 100's of millions of frames (Lenat & Guha 1990; Stanfill & Waltz 1986, etc.) Such KB's would be orders of magnitude larger than any existing today.

To achieve computational effectiveness on large KBs, we designed PARKA with run-time performance as a primary goal of the system. Thus, we somewhat constrained PARKA's expressiveness—this was unavoidable because many operations, are, in general, *NP* or even undecidable for term-subsumption languages that are sound and complete.

[1]Email: evett@cs.umd.edu

[2]Email: hendler@cs.umd.edu

[3]Email: waander@cs.umd.edu

Recognition, in particular, is *NP*, though it is a special case of classification, which can be undecidable (Nebel 1990). So, although PARKA's semantics are roughly based upon those of NETL (Fahlman 1979) and KL-ONE (Brachman & Schmolze 1985), we avoided semantic constructs lacking a computationally effective implementation. We believe PARKA's run-time performance on even very large KBs more than compensates for its slighlty restricted expressiveness when compared to that of other, serial, KR systems.

2.1 Design specifics

PARKA is a basic frame system: each frame corresponds to a concept represented in the KB and the collection of relations to which it belongs. Relations among concepts are represented as directed graphs whose arcs (or links) are stored as frame pointers in the *slots* of frames. Properties of frames are represented as relations for which the domain is the frames having the property, and the range is the corresponding property values. The KB can be viewed as the network formed by the frames and the links that connect them, similar to a semantic network (Fahlman 1979, e.g.)

Ontological relations among frames are encoded via the IS-A ("is a") relation, which is intimately involved in the calculation of property inheritance inferences (see below) As such, it has special status in PARKA. The subgraph consisting of all IS-A links and frames is referred to as the *IS-A hierarchy* of the net. All PARKA IS-A hierarchies are rooted and acyclic.

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.

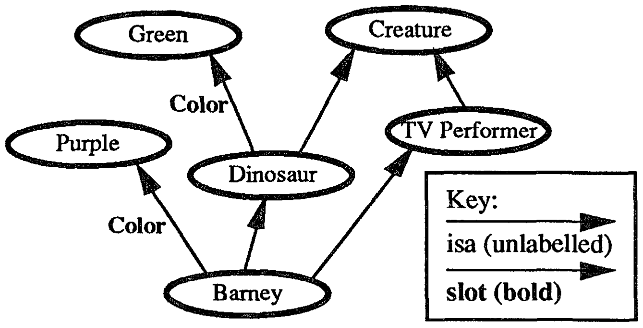

Figure 1: Subset of a Frame Network

2.2 Inheritance Mechanism.

PARKA employs a property inheritance mechanism on its IS-A hierarchy. 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 metric based on Touretzky's *inferential distance ordering* (IDO)) that *is* explicitly-valued for that property. Because PARKA supports multiple inheritance, it is possible to have more than one such node. In that case, PARKA disambiguates among viable ancestors.

The first version of PARKA (Evett, Spector & Hendler 1993; Evett & Hendler 1992) handled multiple-inheritance—as have other frame-based KR systems (i.e., Brachman & Schmolze 1985, e.g.)—by using inheritance path length to disambiguate multiple inheritance paths. The system chooses the ancestor having the shortest IS-A path from the inheritor. This inheritance 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).

The current implementation uses a top-down, path-based, credulous inheritance mechanism based on Touretzky's IDO metric to disambiguate multiple inherited values that works like this: assume the frame in question, X, is not explicitly valued for the given property, P. Let B be the set of ancestors $\{B_1, B_2, ...\}$ of X that are explicitly valued for P. X takes B_i's value for P as its own, provided B_i is an element of B such that there is no B_j $(j \neq i)$ 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 to be ambiguously valued for property P.

Unfortunately, many retrieval operations involving top-down, path-based inheritance mechanisms, including IDO, have been shown to be NP-hard (Selman & Levesque 1989). To calculate these operations in a timely manner, we adopted a slightly weaker ordering scheme for inheritance disambiguation. Again, let B be the set of ancestors of X that are explicitly valued for property P. X takes B_i's value for P as its own, where B_i is that element of B with the largest *topological number*. The topological number, $topo(Z)$, of a frame Z, is defined inductively: $topo(rootNode)=0$ and for all other frames, Z, $topo(Z) = 1 + \max_{y \in C}(topo(y))$ where C is the set of frames that are parents of Z.

Though PARKA's disambiguation mechanism is not quite as powerful as complete IDO, it enjoys many of the same advantages and is considerably stronger than a simple path-length based scheme. It does not suffer from the problems of *redundancy* noted in (Touretzky 1986) and in only one case does our inheritance scheme differ from IDO: in full IDO, if there are two explicitly valued ancestors, X and Y, but X is also an ancestor of Y, then Y is "more specific" than X, and so its property value is chosen. Our topological disambiguation scheme, however, may arbitrarily disambiguate among the two ancestors, even when neither is an IS-A ancestor of the other. In the vast majority of cases, though, PARKA's mechanism is equivalent to IDO. PARKA's encoding of the Cyc commonsense KB (see Section 3.4) revealed no cases in which PARKA disambiguates an inheritance relation that is ambiguous via IDO. If Cyc turns out to be typical of future KBs, this shortcoming of topologicial disambiguation will have little impact on most inferences.

2.3 Implementation

PARKA's internal representation of a frame consists of a block of processors, one for each of that 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 one of the processors 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 smaller rectangles represent the remaining processors of those blocks. The square-bracketed values represent the address of the processor representing the property of that corresponding name.

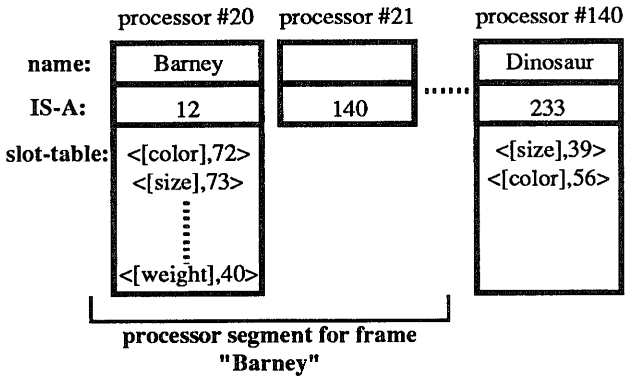

Figure 2: Internal CM representation of a small subset of a frame network

To determine if a frame, *Y*, is explicitly valued for a given property, *P*, PARKA determines the (explicit) value of *P* for every frame in the KB by scanning through every frame's slot table in parallel, seeking an entry corresponding to *P*. If a matching entry is found in frame *Y*'s slot table, the corresponding value stored there is the address of the processor representing the frame that is the value of property *P* for frame *Y*. In general, slot retrieval is proportional to the size of the largest slot table in the KB. Because property values tend to be scattered across inheritance paths, these tables are typically quite small. The largest slot table in our implementation of the Cyc KB had only 43 entries. Even so, PARKA maintains a cache of the most recently accessed properties to accelerate explicit property look-up.

3. Performance

To demonstrate that PARKA provides computationally effective KR, we implemented and timed several retrieval operations on very large KBs which included inheritance queries on very large, pseudo-random KB's and recognition queries on the Cyc KB.

3.1 Inheritance Queries

Almost all KB queries involve some inferencing along the IS-A hierarchy because almost all involve calculating the inherited value of a set of properties for a set of frames. PARKA uses several data structures to make such inheritance inferencing very fast, including using multiple processors to represent frames having multiple parents. PARKA 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 a nascent *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*:

1. Frames with a topological value of *i* (i.e., those at *topological level i*) that are explicitly valued for the property set their wave value to *k*, where *k*'s high order bits are *i*, and *k*'s low order bits are the frame's property value.
2. Every frame (processor) not explicitly valued for the property in question that has parent nodes at topological level *i*-1 "pulls" down the value of the activation wave from those parents.
3. Non-explicitly valued frames at topological level *i* 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 inheritance scheme outlined in section 2.2.

Figure 3: The Basic Inheritance Algorithm

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. We refer to such queries as "top-down", and they are the bane of most serial KR systems, requiring $O(n)$ time to effect, where *n* is the size of the KB. Serial systems use *indexing* schemes to mitigate this computational morass, but indexing can be unsatisfactory for a variety of reasons (as we discuss in (Kettler, Hendler & Andersen 1993b)) including that it is typically infeasible to explicitly index all properties.

The comparison between serial and parallel run-times is more striking when realizing that for realistic networks $d \approx lg(n)$. It is commonly believed that such network shallowness will persist and probably be accentuated as net size increases. Our PARKA implementation of the Cyc commonsense KB (see section 3.4), enjoys a similarly shallow IS-A topology.

To compare PARKA against a serial representation system, we created a serial version of PARKA, called SPARKA ("serial-PARKA"). To make the comparison as fair as possible, we implemented SPARKA as a severely stripped-down version of a more complete serial implementation (detailed in (Spector, Evett & Hendler 1990)). 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 PARKA's response to example queries on topologies of varying size and depth. Then, we timed SPARKA on the same queries and networks. The networks were quite large—up to 128K nodes[4]. Because encoding such large networks by hand was not possible, we developed algorithms for generating pseudo-random networks with certain topological characteristics. These techniques are described in (Evett, Spector & Hendler 1993; Evett & Hendler 1992). Our experience with the Cyc KB (see section 3.4) has affirmed our belief that the topologies used to measure PARKA's performance reflect those of realistic KBs.

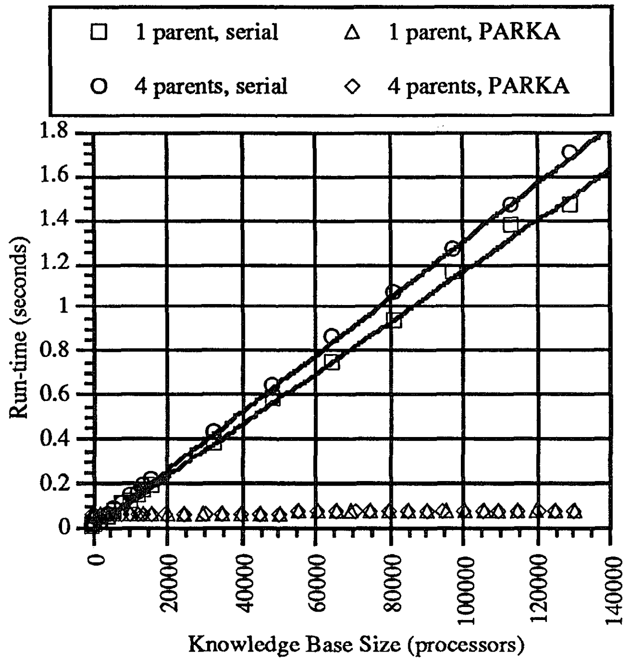

Figure 4: Run-time performance of inheritance queries on 8-level networks of varying sizes.

Figure 4 shows PARKA's run-time for frame networks of depth 8, and of varying size. This timing suite isolates the effect of network size on run-time. These timings support our supposition that PARKA's computation of inheritance queries is independent of network size[5]. Thus, PARKA's performance should scale up to arbitrarily larger KBs. The serial system's run-time, on the other hand, was linear with respect to network size. The figure contains best-fit curves to highlight this linear relation.

The networks used in the timings were of two topological types: trees (each frame with exactly one parent) and directed graphs (each frame with between one and four parents). We used different topologies to demonstrate that PARKA's run-time is independent of upward IS-A fan-out.

A comparison between the performance figures of SPARKA and PARKA in Figure 4 demonstrates that the latter remains computationally effective even for very large KBs, while the former's performance is unacceptable for large networks. We anticipate that this contrast will become increasingly stark for much larger KBs of applications in real-world domains.

3.2 Recognition Queries

The ability to solve *recognition queries* has driven much of PARKA's design. The problem of recognition is well-known in the field of KR (Wilensky 1986, e.g.) and is the problem of answering KB queries of the form: "find all frames x such that $P_1(x,c_1) \wedge P_2(x,c_2) \wedge \ldots \wedge P_p(x,c_p)$", where $P_i(x,c_i)$, $\forall i$, 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 are extremely time-consuming for serial-based systems, often running in time no better than $O(pn)$, even on systems employing a highly constrained description language.

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

But PARKA does even better, using 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 Figure 3. At each propagation step i, all frames at topological level j, such that $p \geq j \geq i$, 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. Thus, the complete propagation requires $d+p-1$ propagation steps.

3.3 Using Cyc for Validation of PARKA

We tested our run-time predictions by timing PARKA's performance for recognition queries on an implementation of the Cyc KB (Lenat & Guha 1990). Our motivation for using Cyc 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

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

[5]Actually, we observed a 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 (Evett & Hendler 1993)). 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.

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.

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)→ fatherOf(Mary,John)), to theorem proving using general "wffs", and to unsound inference methods such as analogy.

PARKA implements only some of the inference capabilities provided by CycL, particularly those having to do with property inheritance. For our tests we represented only that subset of Cyc that involved IS-A based property inheritance and ontologies. This subset contained a total of 26,214 units, 8591 (33%) of which were collections, and 17,623 (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 IS-A relation was 23, (i.e., shallow relative to KB size, as expected.)

3.4 Performance of Recognition Queries

To test recognition query performance, we timed queries similar to those usedby CycL to find units "similar" to a given unit. Units are considered similar if they share the same values for a number of properties exceeding some threshold. First, we selected a Cyc unit (*#%Burma-1986*) with a relatively large number (22) of local assertions (i.e., explicitly-valued properties), assigning an arbitrary ordering to those properties. We then ran recognition queries in PARKA to identify those frames that matched at least 50% of the first *n* slots ($1 \leq n \leq 22$) of *#%Burma-1986*. The recognition queries themselves, then, involved between 1 and 22 conjuncts. The run-time performance of these queries is plotted in Figure 5.

As Figure 5 clearly shows, the time required to perform recognition queries grows only linearly in the number of conjuncts, *p*, and overall performance, even for a query of 22 conjuncts, is excellent. The run-time matches the *O(d+p)* performance predicted by analysis. This performance compares very favorably with recognition queries on serial systems, which require *O(pn)* time for the same queries, where *n* is the size of the KB. Recognition queries in PARKA are independent of KB size, and should scale up to arbitrarily larger domains. Indeed, in (Kettler et al 1993a) we report sub-second run-time performance of recognition queries in a case-based planning system using KB's of over 100,000 frames. This is a speed-up of more than 10,000 over the highly optimized serial version of PARKA.

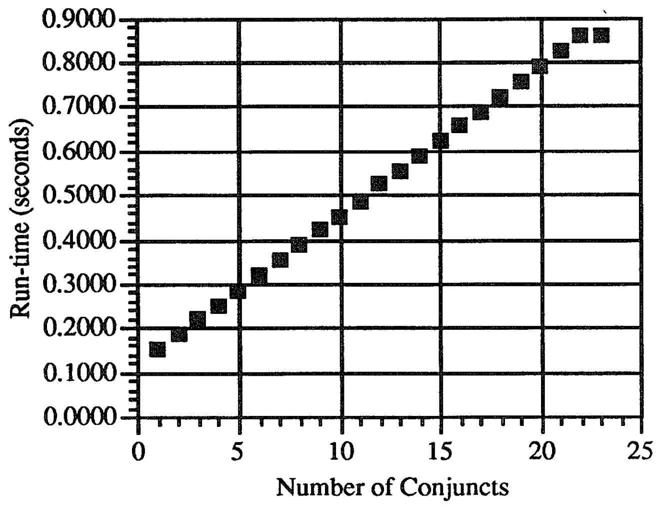

Figure 5: Run-time of recognition queries of various sizes on the PARKA implementation of the Cyc KB

This experiment was designed not ony to demonstrate the *O(d+p)* complexity of conjunctive recognition queries in PARKA, but also to show that PARKA can supply fast matching for analogy-related functions, a task that traditionally has been difficult for serial systems. For example, the CycL query that most nearly corresponds to those of Figure 5 finds only an arbitrary subset of the matching units. By exhaustively matching the probe against the entire KB simultaneously, PARKA finds *all* appropriate matches.

4. Future & Related Work

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

Also, we 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 can use several inferencing techniques that aren't possible on the SIMD CM-2. In particular, 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.

There are a few other parallel KR systems (Geller 1991; Moldovan, Lee & Lin 1989 to name two) and these are discussed more fully in (Evett 1993).

5. Conclusion

Using KBs with over 100,000 frames, we have shown that PARKA computes property inheritance and recognition queries 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. Because empirical evidence to date supports our analytical claims, we believe that PARKA's performance will scale up to larger KBs, even to the those necessitated by memory based reasoning[6] technology. Thus, we argue that PARKA can supply computationally effective recognition queries for realistic KBs.

Acknowledgments

The author wishes to thank the Systems Research Center at the University of Maryland, for their early support in this research, and for the continuing use of their hardware in the development of PARKA. The authors also wish to 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 was 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, R.J. I Lied about the Trees. *AI Mag.* **6**, 3 (Fall, 1985).

Brachman, R.J. and Schmolze, J.G. An Overview of the KL-ONE Knowledge Representation System. *Cog. Sci.*, **9**, 2 (April-June 1985).

J. Computers and Mathematics with Applications — special issue on semantic networks, **23**(2-5), 1992.

Evett, M.P. PARKA: A System for Massively Parallel Knowledge Representation. Ph.D. diss., Dept. of Computer Science, Univ. Maryland, College Park, 1993. Forthcoming.

Evett, M.P., Spector, L. and Hendler, J.A. Massively Parallel Frame-Based Property Inheritance in PARKA. To appear in *Journal for Parallel and Distributed Computing*.

Evett, M.P. and Hendler, J.A. Degradation of Interprocessor Communication Operations on the Connection Machine. Tech. Rep., Department of Computer Science, Univ. Maryland, College Park, March, 1993.

Evett, M.P. and Hendler, J.A. 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, S.E. *NETL: A System for Representing and Using Real World Knowledge*. MIT Press, Cambridge, MA, 1979.

Geller, J. Advanced Update Operations in Massively Parallel Knowledge Representation. Tech. Rep. CIS-91-28, Dept. Computer and Information Science, New Jersey Institute of Technology, Newark, NJ, 1991.

Hammond, K. *Case-Based Planning: Viewing Planning as a Memory Task.* Academic Press, 1989.

Kettler, B.P., Hendler, J.A., Andersen, W.A., and Evett, M.P. (1993a) "Massively Parallel Support for a Case-based Planning System". In Proceedings of the Ninth IEEE Conference on AI Applications, IEEE 1993.

Kettler, B.P., Hendler, J.A. and Andersen, W.A. (1993b) Why Explicit Indexing Can't Work. Tech. Rep., Department of Computer Science, Univ. Maryland, College Park, April, 1993.

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

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

Moldovan, D., Lee, W., and Lin, C. SNAP: A Marker-Propagation Architecture for Knowledge Processing. Tech. Rep. CENG 89-10, Dept. Electrical Engineering-Systems, Univ. of Southern California, Los Angeles, CA, 1989.

Nebel, B. Terminological Reasoning Is Inherently Intractable. *AIJ*, **43**, 2 (May, 1990).

Selman, B. and Levesque, H. The Tractability of Path-Based Inheritance. *Proceedings of IJCAI-89*, Morgan-Kaufman, San Mateo, CA, 1989.

Shastri, L. Massive Parallelism in Artificial Intelligence. Tech. Rep. MS-CIS-86-77 (LINC LAB 43), Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, 1986.

Sowa, J. (ed.) *Principles of Semantic Networks.* Morgan-Kaufman, San Mateo, CA, 1991.

Spector, L., Evett, M. and Hendler, J.. Knowledge Representation in PARKA. Tech. Rep. TR-2409, Department of Computer Science, University of Maryland, College Park, MD, Feb. 1990.

Stanfill, C. and Waltz, D. Toward Memory-Based Reasoning. *Communications of the ACM*, Vol. 29, No. 12, December 1986, pp. 1213-1228.

Touretzky, D.S. *The Mathematics of Inheritance Systems*. Morgan Kaufmann, Los Altos, CA, 1986.

Von Eicken, T., Culler, D., Goldstein, S. and Schauser, K. Active Messages: a Mechanism for Integrated Communication and Computation. Tech. Rep. UCB/CSD 92/#675, Computer Science Division, EECS, University of California, Berkeley, CA, 1992.

Wilensky, R. Some Problems and Proposals for Knowledge Representation. Tech. Rep. UCB/CSD 86/294, University of California, Berkeley, May 1986.

[6]We use the term "MBR" in a broader sense than in (Stanfill & Waltz 1986) to include such paradigms as case-based reasoning (Hammond 1989; Kettler et al 1993; Kitano & Higuchi 1991.)