ForMAT and Parka: A Technology Integration Experiment and Beyond*

David Rager[1], James Hendler[1], and Alice Mulvehill[2]

[1] Department of Computer Science, University of Maryland, College Park MD 20742
[2] BBN Systems and Technologies, 10 Moulton Street, Cambridge MA 02138

Email: rager@cs.umd.edu, hendler@cs.umd.edu, amm@bbn.com

Abstract. This report describes a Technology Integration Experiment (TIE) between the University of Maryland and The MITRE Corporation which was undertaken as part of the (D)Arpa/Rome Laboratory Planning Initiative (ARPI). This work led to an integration of the UM Parka-DB tool into the MITRE ForMAT transportation planning tool.

1 Background

In case-based planning (CBP), previously generated plans are stored as cases in memory and can be reused to solve similar planning problems. CBP can save considerable time over planning from scratch (generative planning), thus offering a potential (heuristic) mechanism for handling intractable problems. The CaPER system [8], focused on new approaches to CBP. In particular, one drawback of CBP systems has been the need for a highly structured memory that requires significant domain engineering and complex memory preindexing schemes to enable efficient case retrieval

In contrast, the CaPER CBP system used the Parka high performance knowledge representation system [9] to retrieve plans quickly from a large memory that was not preindexed. Thus, it was relatively inexpensive to access memory frequently, and memory could be probed flexibly at case retrieval time. CaPER issued a variety of queries that resulted in the retrieval of one or more plans (or parts of plans) that were combined to solve the target planning problem. These plans could be merged and harmful interactions among them resolved using annotations on a plan to capture interdependencies among its actions [8, 7, 5, 9].

Transportation Logistics Planning The United States Transportation Command (USTRANSCOM) is responsible for generating and maintaining the plans by which United States military forces are deployed. This responsibility includes determining the transportation needs for missions short and long, small and very large. For large missions, the

* This research was supported in part by grants from NSF(IRI-9306580), ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065), the ARPA/Rome Laboratory Planning Initiative (F30602-93-C-0039), ARPA I3 Initiative (N00014-94-10907), ARPA contract DAST-95-C0037 and the Army Research Laboratory. Dr. Hendler is also affiliated with the UM Institute for Systems Research (NSF Grant NSF EEC 94-02384).

process by which these transportation plans are constructed can be very complex and time consuming. Representatives from the various services and commands involved in a plan must collectively decide how best to allocate the limited transportation resources (aircraft, ships, trucks and trains) to achieve the many military goals of the mission. The end result of this process is an Operational Plan (OPLAN) which specifies where and when the forces involved in a mission are to be moved. Associated with an OPLAN are one or more Time Phased Force Deployment Data (TPFDD) which describe what, when, and how the forces for a mission will be deployed. The OPLAN and TPFDDs are stored and maintained until their execution is called for. At that time, the plan will generally have to be modified to fit the particular details of the current situation.

ForMAT (Force Management and Analysis Tool) provides an environment in which a force development and planning specialist can view, modify, and create the basic structures of TPFDDs (called force modules, or FMs). FMs prescribe a force or set of forces that can be used to satisfy some planning requirement. The FM is typically a grouping of combat, combat support, and combat service support forces, and ranges in size from the smallest combat element to the largest combat group. It may specify accompanying supplies and the required movements, resupply, and personnel necessary to sustain forces for a minimum of 30 days. The elements of a FM are linked together so that they may be extracted from, or adjusted as, an entity to enhance the flexibility and usefulness of a plan. One or more FMs for use in a given plan are stored in a TPFDD. In theory, FMs form a library which can be drawn upon to quickly build a new plan. In a crisis, new TPFDDs will be built, at least in part, from FMs within one or more existing TPFDDs.

The force modules that compose TPFDDs are themselves composed of smaller units called Unit Line Numbers (ULNs). A ULN identifies a force, support for a force, or a portion of a force. A ULN is often described by its Unit Type Code, which can span a wide range of items from tactical fighter squadrons and army battalions to dog teams, or even a Catholic chaplain. Finding appropriate ULNs (and therefore FMs) in previous TPFDDs is a complex task, similar to case-retrieval in case-based planning.

2 High Performance Support for ForMAT

The integration of ForMAT and the University of Maryland system began from a Technology Integration Experiment (TIE) which examined whether the “structure matching” system developed for supporting the case-based reasoning in CaPER could also handle retrieval in ForMAT. Before we describe this integration, we review the approach to structure matching in CaPER and the parallel versions of the algorithms that provided the original speedup needed to support a knowledge base as large as ForMAT required. (Note that we eventually were able to support ForMAT on a single processor using a data-based version of Parka. This work is described in Section 3.)

2.1 Representing Structures for Matching

Our description of the problem of structure matching follows that given in [12]. A knowledge base (KB) defines a set, P, of unary and binary predicates. Unary predicates have the form $P_i(x)$ and binary predicates have the form $P_j(x_1, x_2)$, where each x_i is a variable on the set of frames in the KB. An existential conjunctive expression on these predicates is a formula of the form $\exists x_1, \ldots, x_m : P_1 \wedge P_2 \wedge, \ldots, \wedge P_n$, where $n \geq 1$. Our task is to retrieve all structures from memory which match a given conjunctive expression. Therefore, we would like to find all such satisfying assignments for the x_i.

We can view the problem of matching knowledge structures in two ways. The first is as a subgraph isomorphism problem [3]. We view variables as nodes and binary predicates as edges in a graph. We want to find structures in memory which "line up" with the graph structure of the query expression. The other way to view the matching problem is as a problem of unification or constraint satisfaction. If we can find a structure in memory which provides a consistent assignment to the variables x_i (i.e., unification), then that structure matches the conjunctive expression.

Overview of the Algorithm. The structure matching algorithm operates by comparing a retrieval probe, P, against a knowledge base to find all structures in the KB which are consistent with P. This match process occurs in parallel across the entire knowledge base. A Parka KB consists of a set of frames and a set of relations (defined by predicates) on those frames. Most relations are only implicitly specified and so must be made explicit by expanding the relation with the appropriate inference method. By computing inherited values for a relation, all pairs defining the relation are made explicit. We currently allow only unary and binary relations.

A retrieval probe is specified as a graph consisting of a set of variables $V(P)$ and a set of predicates (or constraints) $C(P)$ that must simultaneously hold on frames bound to those variables. The result of the algorithm is a set of k-tuples, where each k-tuple encodes a unique $1-1$ mapping of frames to variables in $V(P)$, that unifies with the description of the structure in memory with $C(P)$. The set of frames that can bind to each variable is initially restricted by a set of constraints indicated by unary predicates. Each unary constraint may only constrain the values of one variable. Examples of these constraints are "X is a dog" or "the color of X is yellow". We allow set theoretic combinations of the unary constraints, for example "X is a dog and the color of X is yellow", or "X is a dog but X is not yellow"[4] The domains for each variable are maintained throughout the match process and are further restricted as more constraints are processed.

Constraints between frames bound to variables are specified by a set of binary constraints. For example, we can say "the color of X must be Y", or "X is a part of Y", for some X and Y in $V(P)$. Binary constraints are processed by "expanding" the binary relation given in the constraint. By expansion we mean that all pairs participating in a relation R in the KB are made explicit by invoking the inference method for the associated predicate. The pairs allowed to participate in the expanded relation

[3] More specifically, this is a problem of Directed Acyclic Graph (DAG) isomorphism with typed edges, the edges being the relations in the KB between frames.

[4] Variables in the query probe which do not appear in a unary constraint are treated differently. Variables not contained in a unary constraint are still able to be constrained by intersecting the instances of the range and domain of the predicates in binary constraints in which the variable appears.

are restricted to those in the domains of the variables related by R. For example, a binary constraint may be expressed as (Color X Y). In this case the values for each concept in the domain of X are computed for the color predicate and pairs that have values outside the domain of Y are excluded. Two additional binary predicates, "eq" and "neq" are provided to provide codesignation and non-codesignation of variables. These constraints act as a filter, eliminating any tuples from the result for which the constrained variables are(not) bound to the same frame.

The result of a structure match is a set of k-tuples, where each tuple corresponds to a satisfying assignment of the k variables. Alternatively, the result can be viewed as a relation. Initially, the matcher begins with an empty set of relations. During the match, several intermediate relations may be constructed. Simple binary relations result from the expansion of a binary constraint. These are later fused (via a relational join operation) or filtered (via codesignation or non-codesignation) until a single relation remains. The algorithm selects binary constraints to process using a greedy algorithm based on a simple cost model stored in the metadata.

3 Integrating ForMAT and PARKA

Based on our successes in the creation of the single processor version of PARKA, we focused our effort on the design of a new version which would optimize single-processor performance using a database system to provide both efficiency and scalability. The resulting system, PARKA-DB was used in the actual support of ForMAT described below. (For convenience we drop the "DB" and refer to the system by the original name "PARKA" in the remainder of this report.)

3.1 TPFDD Casebase

Initially, we encoded one of the ForMAT casebases into the PARKA knowledge representation system. MITRE provided the casebase of TPFDDs, and at UMCP a program was written to recode it into the database tables used by PARKA. The casebase contains 15 TPFDDs, consisting of a total of 319 FMs and about 14,000 ULNs. The corresponding PARKA knowledge base consists of 54,580 frames, 31,314 structural links, and 607,750 assertions. In addition, a domain-specific ontology was created containing approximately 1,200 frames for domain concepts such as "FM", "ULN", service branch, capabilities, functions, geographic locations, etc. Frames in the base ontology are organized in an abstraction ("is-a") hierarchy.

The initial casebase was tested using a graphical interface front end to PARKA (see Fig. 1). Through this testing we developed the base ontology and ensured that the PARKA casebase could handle the retrieval tasks required by ForMAT. Two addition features were added to PARKA to support ForMAT queries, string matching and variable restrictions. String matching provides the ability to search for values that contain a substring, such as searching for a description string that contains the substring "dog." Variable restrictions allow the user to restrict results to a set of values. In ForMAT it is used to restrict the search space of a query to a specific set of FMs instead of the entire casebase.

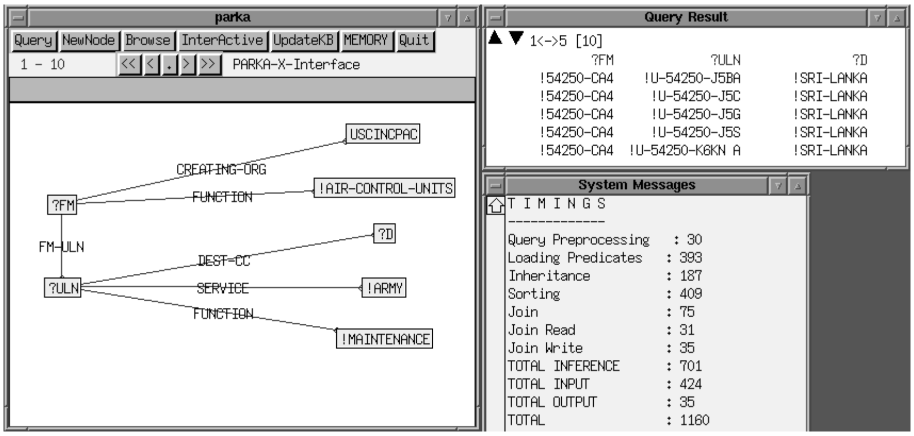

Fig. 1. The PARKA front-end graphical representation of a ForMAT query.

3.2 How ForMAT Uses PARKA

PARKA was designed as a knowledge base with a simple application program interface (API). Any program that conforms to the API can use the PARKA back-end. The PARKA browsing and quering tools are a graphical interface front-end accessing the PARKA back end using the same API. Code was added to ForMAT to allow it to access PARKA. By conforming to the PARKA API, ForMAT can use any back-end version of PARKA, such as our recently developed distributed version or any of the parallel versions, transparently to the user.

The ForMAT system from MITRE used a similar API to access its casebase retrieval system. To integrate ForMAT and PARKA, the ForMAT casebase code was removed and replaced by code to connect the two APIs. This code uses the ForMAT query specification to create the PARKA query. Because ForMAT is written in LISP and PARKA is in C, the LISP foreign function interface was used to communicate between the two languages.

When a ForMAT user wants to search the casebase, they build a query using the ForMAT FM Query interface (See Fig. 2). When the query is executed, the query is converted into a PARKA query which is passed to the back-end where the retrieval is done. The results are passed back to ForMAT where they are displayed to the user.

ForMAT supports two types of retrieval, exact and general. An exact query searches for FMs with specific features, while a general query searches for FMs with similar but not necessarily exact features. An exact ForMAT query is very much like a PARKA query. The ForMAT interface allows the user to specify some FM and ULN features. These features are then converted into the binary predicates used in the PARKA casebase. The query shown in Fig. 2 searches for a force module with three features. The corresponding PARKA query includes those features as binary predicates plus a predicate that specifies the object with those features is a force module.

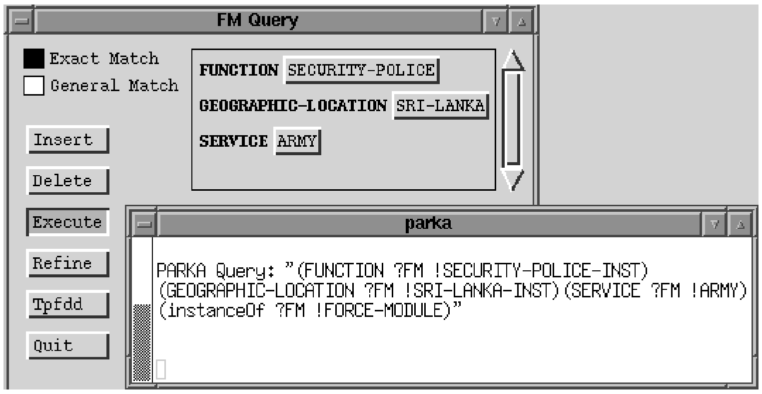

Fig. 2. An example of the ForMAT query window and a PARKA query.

For general queries, PARKA needs to perform multiple queries, each more general than the previous one. For example, a query may look for FMs with a geographic-location in Saudi Arabia. An exact match would only return FMs with the exact value of Saudi Arabia. If no such FMs exist, none would be returned. A general match would perform a second query if no FMs were found. The second query would relax the constraints on geographic-location and search for any values under Middle East in the base ontology. This generalization would continue until results are found.

ForMAT also allows users to specify queries that include OR and NOT, but Parka only does conjunctive queries. For disjunction, the query is converted into multiple queries and the results are combined. (A and (B or C)) becomes the two queries (A and B) and (A and C). NOT is handled by doing two queries, first the query is done without the predicates within the NOT. The results are saved and the query is done again to find the FMs to remove from the previous results. For example (A AND (NOT B)) would become A set-difference (A AND B).

3.3 Comparisons

We used a set of 379 queries to compare the two retrieval methods. The queries are actual ForMAT queries done by users during testing and military planning exercises (taken from history files). The queries were run 10 times each as exact queries and the times were averaged. Both methods returned the same FMs for all queries.

The timing results are graphed in Fig. 3. The speeds of the two methods are roughly the same, with PARKA having some advantage on larger queries. On average, across the 379 cases, the PARKA results are 10ms faster than the original ForMAT algorithm. Most of the queries take less than 100ms using either retrieval method. These are queries that only search FM features. The slower queries are ones that also search for a combination of FM and ULN features. ForMAT's algorithm is based on a linear search algorithm that searches all the ULNs that belong to the FMs returned by the FM part of the query. Depending on the query, a large number of ULNs (up to 14,000) could be searched. The more FMs searched by the query, the better the PARKA algorithm does in comparison to ForMAT's search of all the ULNs.

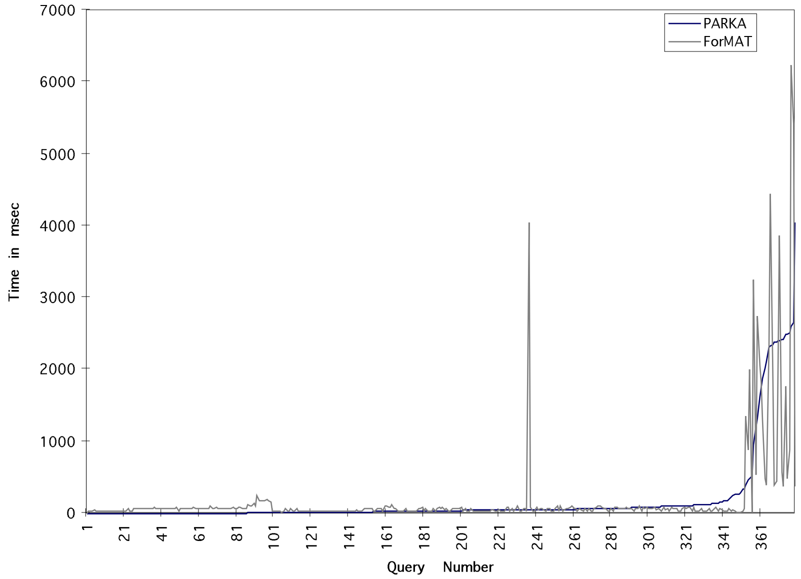

Fig. 3. Graph of ForMAT and PARKA query times sorted by PARKA times.

One problem with these results is that the queries were very varied, and were collected across many uses of ForMAT. However, the PARKA algorithms were developed so that memory caching would provide a significant improvement on queries which were related to each other as was expected would happen in a typical ForMAT use. Therefore, for a more concise comparison, we needed a set of specific queries representing a "typical" use of ForMAT. MITRE provided a set of queries for this purpose. In particular, as part of the Joint Warfare Interoperability Demonstrations (JWID), ForMAT was configured to communicate with the TARGET system[11]. One set of queries in this experiment required 17 different searches of the casebase with a wide range of queries including both FM and ULN features.

Our comparison tests showed that the FMs returned by the two systems were the same, but PARKA was significantly faster than ForMAT. Figure 4 shows the results of this test. The total time of the TARGET query is 73.9 seconds for ForMAT and 8.1 seconds for PARKA. Thus, we see that for this query set PARKA is about 9 times as fast as ForMAT alone.

We are able to demonstrate from this experiment that PARKA does much better at casebase queries that include ULN features. This is because including the ULNs increases the search space from 319 FM structures to 319 FM plus 14,000 ULN structures. This shows that the PARKA algorithm will do better than the ForMAT algorithm as the size of the casebase grows.

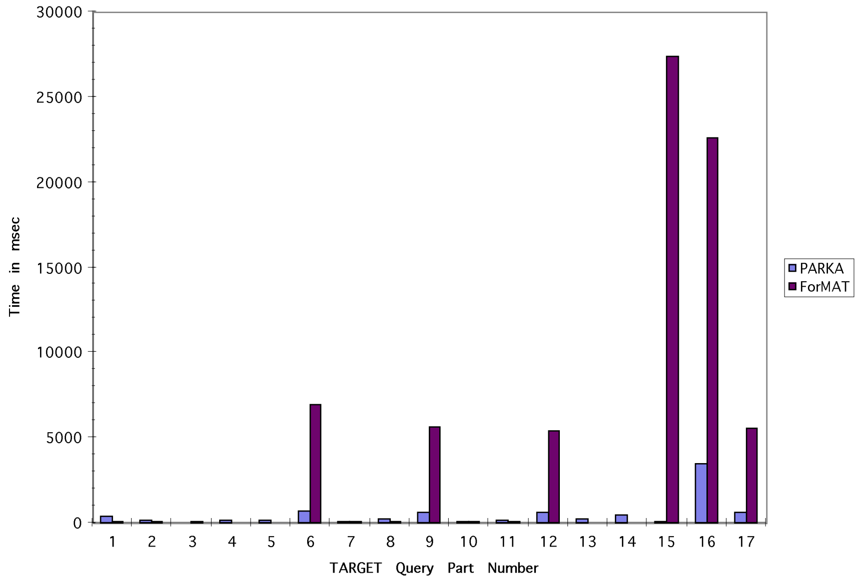

Fig. 4. Graph of ForMAT and PARKA query times for the parts of a TARGET query.

Although ForMAT and PARKA are fully integrated, there is still room for improvement. Currently, PARKA only returns the names of the FMs that were retrieved from the casebase. The LISP structures corresponding to these names are then retrieved from a hash table of FMs. The time it takes to do this is included in the PARKA timing results. A closer integration which allowed ForMAT to use the data stored within the PARKA KB would eliminate the need for the separate LISP structures and improve the overall system performance and memory requirements.

4 Continuing Work

The integration of ForMAT and PARKA is continuing as part of a Phase III ARPI project. One significant improvement made recently is that an integration of the PARKA system and another UM system was exploited to allow ForMAT to become significantly more interoperable with external data sources. In particular, ForMAT users may find additional information such as aircraft and vehicle specifications, airport capacity, maps, or weather information helpful when creating new TPFDDs. The HERMES system[10] was developed at the University of Maryland by VS Subrahmanian to

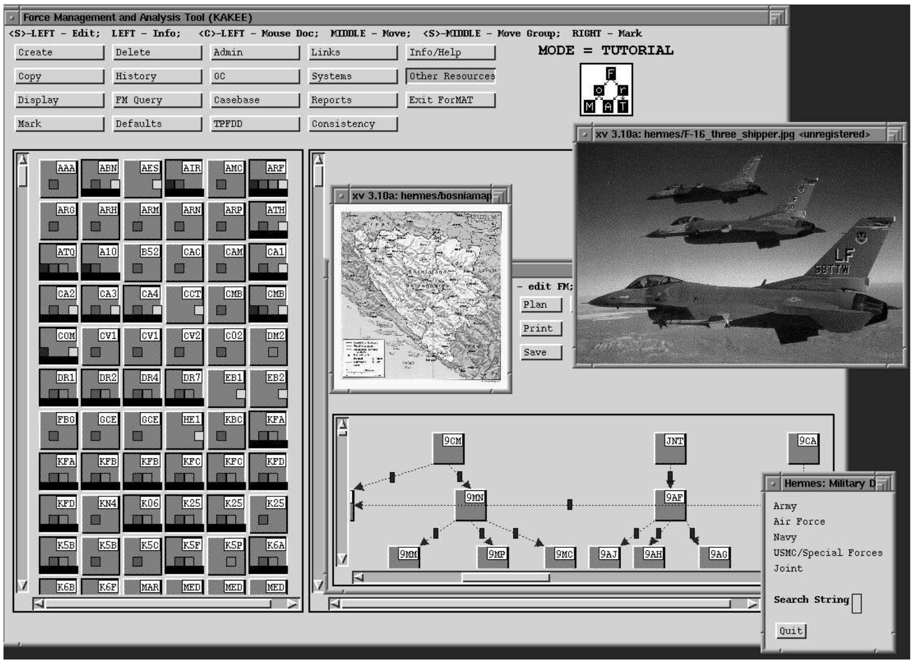

Fig. 5. The working prototype combines ForMAT's interface, PARKA's inference engine and Hermes access to web pages and graphics to provide a unified system for case-based transportation planning.

provide for the semantic integration of different and possibly heterogeneous information sources. We have added an interface to the HERMES server to the ForMAT tool. Through a hierarchical menu the user has access to a number of online factbooks, maps, and weather information. HERMES returns a list of information sources that can be viewed in Netscape or a graphics viewer. Figure 5 shows the integrated ForMAT/PARKA/HERMES system in use for a Bosnia scenario developed as part of the Phase III case-based planning cluster.

Currently, we are looking at implementing a new system which will be a successor to ForMAT written to more completely exploit PARKA's capabilities as well as to greatly extend the coverage of ForMAT. We currently plan that this system will be jointly implemented by BBN and the University of Maryland with the goal of transitioning the resulting system into military use.

5 Conclusion

We believe that the joint ForMAT/PARKA system was one of the successful demonstrations that the technology base for Arpa provides approaches that can be exploited by the military. The new version of ForMAT, via the integration of PARKA, is more capable, faster, and more usable than the earlier version. The University of Maryland team profited by use of real data and a specific knowledge base to focus our optimization efforts on. The ForMAT system gained by use of this new technology. Final experiments proved that the system had gained a significant speedup (in the experiment with the TARGET query the integrated system was nearly 10 times as fast as the unaugmented ForMAT system). In addition, this effort is now continuing, with a joint tool under development that is expected to surpass the original in speed, size and, most importantly, functionality.

References

[1] Andersen, W., Evett, M., Hendler, J. and Kettler, B. "Massively Parallel Matching of Knowledge Structures," in *Massively Parallel Artificial Intelligence*, Kitano, H. and Hendler, J. (eds.), AAAI/MIT Press, 1994.

[2] Evett, M.P., Hendler, J.A., and Spector, L., "Parallel Knowledge Representation on the Connection Machine," *Journal of Parallel and Distributed Computing*, 1994.

[3] Evett, M.P. *PARKA: A System for Massively Parallel Knowledge Representation*, Ph.D. thesis, Dept. of Computer Science, University of Maryland, College Park, 1994.

[4] Evett, M.P., Hendler, J.A., and Andersen, W.A., "Massively Parallel Support for Computationally Effective Recognition Queries", *Proc. Eleventh National Conference on Artificial Intelligence,* 1993.

[5] Hendler, J. High Performance Artificial Intelligence, *Science*, Vol. 265, August, 1994.

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

[7] Kettler, B.P., Hendler, J.A., Andersen, W.A., Evett, M.P., "Massively Parallel Support for Case-based Planning", *IEEE Expert*, Feb, 1994.

[8] Kettler, Brian "Case-based Planning with a High-Performance Parallel Memory," Doctoral Dissertation, Department of Computer Science, University of Maryland, October, 1995.

[9] K. Stoffel, J. Hendler and J. Saltz, High Performance Support for Very Large Knowledge Bases, *Proc. Frontiers of Massively Parallel Computing*, Feb, 1995 (Extended Abstract).

[10] VS Subrahmanian, Sibel Adali and Ross Emery, A Uniform Framework For Integrating Knowledge In Heterogeneous Knowledge Systems, *Proceedings of the Eleventh IEEE International Conference of Data Engineering*, March, 1995.

[11] Walker, E. and Cross, S. eds. *Proceedings of the ARPA-Rome Laboratory Planning Initiative Annual Workshop.* San Mateo, Ca., Morgan-Kaufmann Publishers, 1994.

[12] Watanabe, L., and Rendell, L., "Effective Generalization of Relational Descriptions", AAAI Eighth National Conference on Artificial Intelligence, 1990.