REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January, 1997	Final March 1993 - December 1996

4. TITLE AND SUBTITLE

ForMAT and Parka: A Technology Integration Experiment and Beyond

5. FUNDING NUMBERS

F3060293C0039

6. AUTHOR(S)

James A. Hendler, Kilian Stoffel, Alice Mulvehill

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Maryland
Department of Computer Science
AV Williams Building
College Park, MD 20742

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy
Office of Naval Research
Resident Representative
101 Marietta Tower, Suite 3805, Atlanta, GA 30303

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution unlimited

12b. DISTRIBUTION CODE

19970213 059

13. ABSTRACT *(Maximum 200 words)*

This report describes a Technology Integration Experiment between the University of Maryland and Mitre Corp. 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. This work also forms one of the cornerstones of the "Case-based Planning" cluster of the current phase of the ARPI.

DTIC QUALITY INSPECTED 4

14. SUBJECT TERMS

Computers, Software, Artificial Intelligence, Planning

15. NUMBER OF PAGES

21

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unclassified

ForMAT and Parka: A technology integration experiment and beyond

J. Hendler, K. Stoffel, and D. Rager
Computer Science Department
University of Maryland

Alice Mulvehill
Bolt BBN Systems & Technologies
Cambridge, MA.

February 10, 1997

Abstract

This report describes a Technology Integration Experiment (TIE) between the University of Maryland and MITRE Corp. 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. This work also forms one of the cornerstones of the "Case-based Planning" cluster of the current phase of the ARPI.

1 Planning Background

In case-based planning (CBP), previously generated plans are stored as cases in memory and can be reused to solve similar planning problems in the future. CBP can save considerable time over planning from scratch (generative planning), thus offering a potential (heuristic) mechanism for handling intractable problems. With our system, CaPER, we are currently developing 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 uses the Parka high performance knowledge representation system [9] to retrieve plans quickly from a large memory that is not preindexed. Thus, it is relatively inexpensive to access memory frequently, and memory can be probed flexibly at case retrieval time. CaPER can issue a variety of queries that result in the retrieval of one or more plans (or parts of plans) that can be combined to solved the target planning problem. These plans can be merged and harmful interactions among them resolved using annotations on a plan to capture interdependencies among its actions [8, 7, 5, 9].

1.1 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 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 a 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 [11]. A knowledge base 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 [1]. 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.

2.1.1 Overview of the algorithm

The structure matching algorithm operates by comparing a retrieval probe, P, against a knowledge base (KB) 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"[2] The

[1] 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.

[2] 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

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 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.

2.1.2 Testing Structure Matching Queries

At the time of the first experiments between CaPER and ForMAT, we had just completed a migration from a CM-2 *SIMD* implementation of Parka to a *MIMD* implementation running on a number of supercomputers including an IBM SP2, a CRAY T3D and several other machines. One domain for testing the MIMD implementation was on knowledge bases that were created as part of doctoral thesis work on the CAPER system itself [8, 7]. Part of this

intersecting the instances of the range and domain of the predicates in binary constraints in which the variable appears.

project involved the automatic seeding of large case memories by a generative planner. One domain used in this work was the "UM Translog" domain, a logistics planning domain developed for the evaluation and comparison of AI planning systems. Case-bases of various size were created, and each contains a number of plans, derivation information, and planning related ontologies.[3] To measure the performance of the structure matcher we used the UM-Translog KB in different sizes (20 cases, 100 cases and 200 cases).

The results, which we review briefly in this section, were interesting for two reasons. On the one hand, they showed that we could clearly scale our algorithms to extremely large knowledge bases (as far as we know the 200 case CaPER case base, which contained over 1.6 million assertions, was the largest meaningful semantic network ever created at the time of these experiments.) More importantly, we showed that the reimplemented system had the capability to handle these large KBs on a single Sparc workstation as well as on the faster machines.

As the CaPER planning system solved a problem, we stored all the queries that were generated. For testing the parallel system we chose several queries at random from a large number of stored queries. The results are summarized in Table 1. Table 1a presents the timings for six typical queries on a single processor SPARC 20 using the three different KBs. Table 1b presents the timings on 1, 8 and 16 nodes of an SP2 using the 200 case KB (the largest of the three). The actual sizes of these KBs are shown in Table 2, where frames is the number of nodes in the DAG, structural links are those in the ISA ontology, and property links are all others (i.e. the number of the edges in the DAG is equal to the structural links plus the property links).

As can be seen, the sequential timings range from under a second for the simplest query to about 7 seconds for the most complex. On the parallel system all queries were executed in under one second, with the simplest query taking only 29 milliseconds on 16 processors, and the most complex taking only 303 milliseconds. Table 1b also shows the efficiencies of the parallel algorithm – the efficiency averaged about 69.3% for eight processors and 59.9% for 16 processors.

[3] A full description of the domain, the complete specification of the planning operators used, and the case-bases themselves are available on the World-Wide Web at *http://www.cs.umd.edu/projects/plus/UMT/*.

Query	20 CB	100 CB	200 CB
1	1020	4740	6990
2	195	1305	1635
3	225	1470	1725
4	630	3570	4590
5	675	3600	4605
6	405	585	645

a: Timings (milliseconds) on a SPARC 20 for a 20, 100 and 200 cases KB

Query	1	8	16	1:8	1:16
1	3041	546	313	69.6	60.7
2	713	129	75	69.1	59.4
3	753	135	79	70.0	56.6
4	1997	361	205	69.1	60.9
5	2003	360	206	69.5	60.8
6	284	52	29	68.2	61.2

b: Timings (milliseconds) on a 1, 8 and 16 nodes of an SP2 on 200 case KB

Table 1: UM-Translog Timings (serial and parallel).

cases	frames	structural links	property links
20	11612	26412	114359
100	59915	130558	800481
200	123173	266176	1620456

Table 2: Sizes of the UM-Translog KBs.

3 Integrating ForMAT and PARKA

Based on the results reported above, it was clear that Parka, the inference engine that supports CaPER, could scale to the needs of ForMAT. In addition, based on our successes in the creation of the single processor version, 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 Figure 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.

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 using the API. By conforming to the PARKA API,

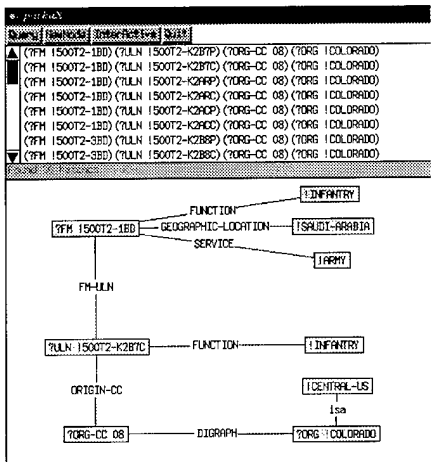

Figure 1: An example of the PARKA front-end graphical representation of a ForMAT query.

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. Only six PARKA API functions need to be called through the foreign function interface.

When a ForMAT user wants to search the casebase, they build a query using the ForMAT FM Query interface. When the query is executed, the query is converted

into a PARKA query which is passed through the PARKA API to the back-end where the retrieval is done. The results are passed back through the API 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 figure 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.

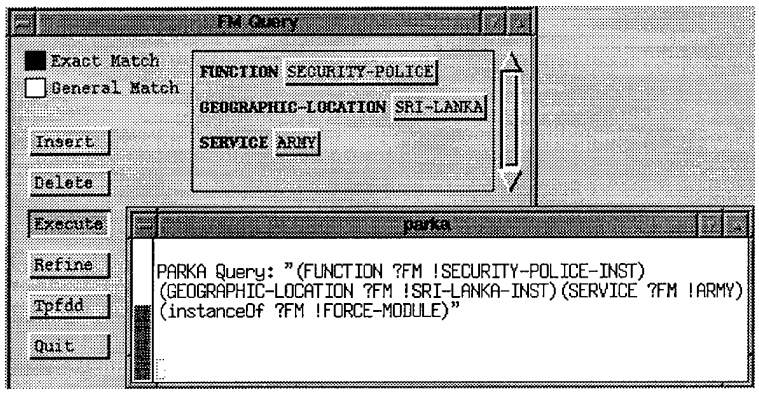

Figure 2: An example of the ForMAT query window and a PARKA query.

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 continues until some results are found. Figure 3 shows a portion of the geographical hierarchy from PARKA. Figure 4 shows a general query as created in ForMAT's FM Query window and the PARKA queries generated by executing that query.

ForMAT also allows users to specify queries that include OR and NOT, but

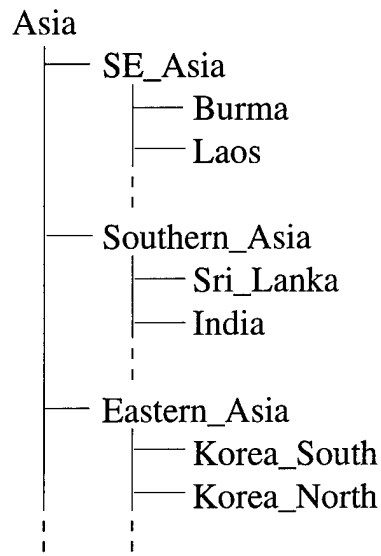

Figure 3: Part of the graphical hierarchy in ForMAT

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 Figure 5. 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

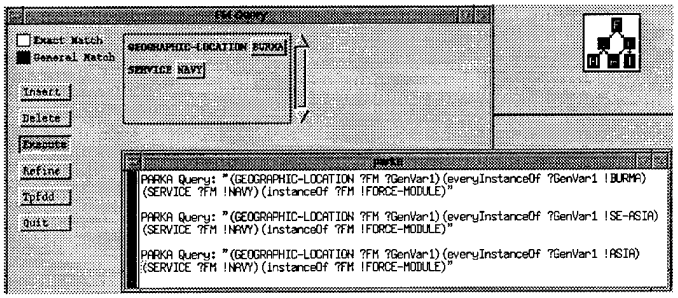

Figure 4: An example of multiple PARKA queries generated from a general query.

method. These are queries that only search FMs. The slower queries are ones that also search for a ULN feature. ForMAT's algorithm is based on a linear search algorithm that searches all the ULNs that belong to the FMs returned by th FM part of the query. Depending on the query, a large number of ULNs (up to 14,000) could be searched. The more FMs returned by the query, the better the PARKA algorithm does in comparison to ForMAT search of all the ULNs.

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 integrated with the TARGET system. One set of queries in this experiment had been particularly time-consuming for ForMAT. Generating a query report for TARGET required 17 different searches of the casebase with a wide range of queries including both FM and ULN queries.

The results showed that the FMs returned by the two systems were the same, but PARKA was significantly faster than ForMAT. Figure 6 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. (In experiments ongoing at the time of this publication, we have ported the new version of Parka to the SP2 and done some optimizations for the TPFDD data. Current results show the parallel version of the TARGET

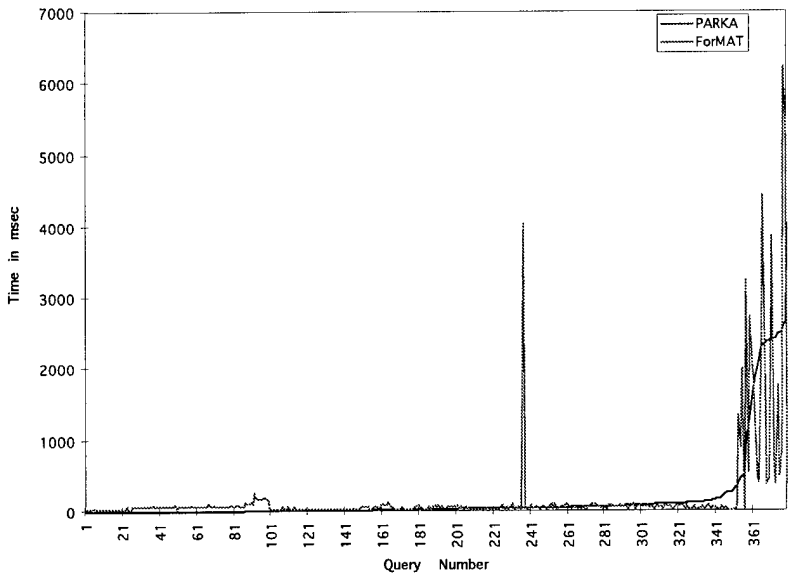

Figure 5: Graph of ForMAT and PARKA query times sorted by PARKA times.

set running on an IBM SP2 over 100 times as fast as the original ForMAT.)

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.

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 casebase would eliminate the need for the separate LISP structures and improve the overall system performance.

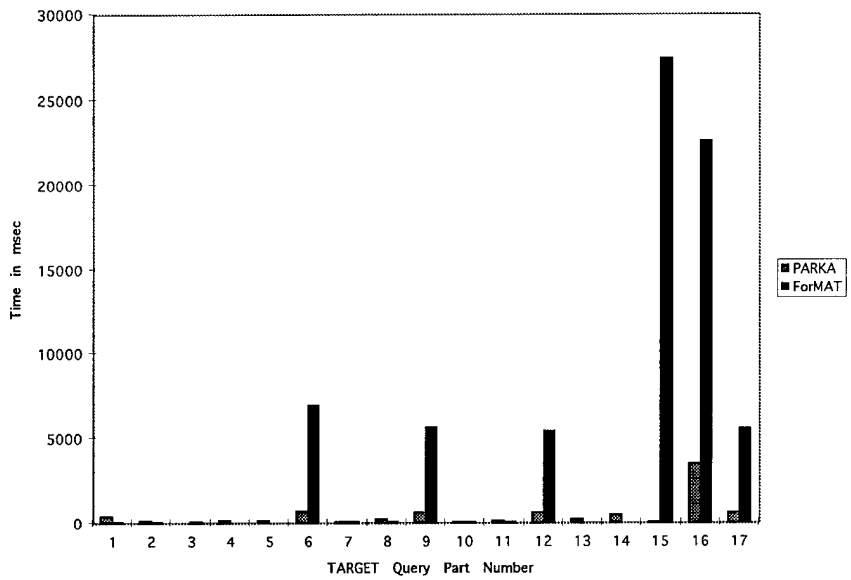

Figure 6: Graph of ForMAT and PARKA query times for the parts of a TARGET query.

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 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 7 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 suc-

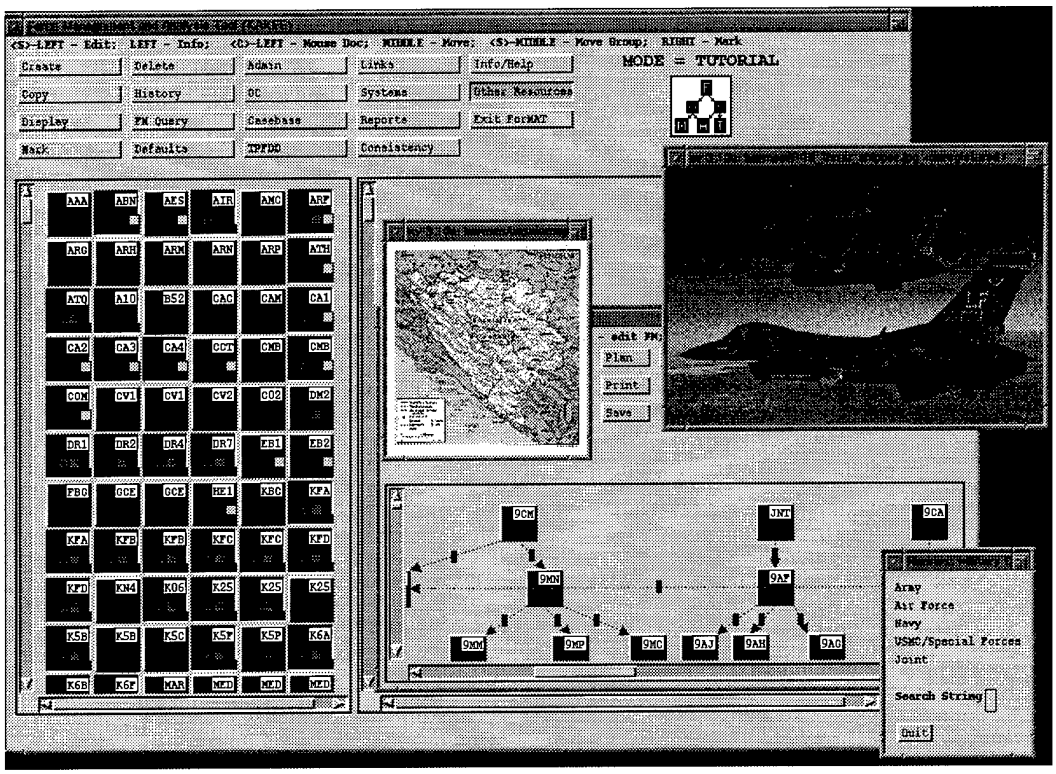

Figure 7: The working prototype combines ForMAT's interface, Parka's inference engine and Hermes access to web pages and graphics to provide a unified system for case-based transportation planning.

cessor to ForMAT written to more completely exploit Parka's capabilities as well as to greatly extend the coverage of the 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, more than 100 times as fast in parallel). 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] M.P. Evett. *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] Watanabe, L., and Rendell, L., "Effective Generalization of Relational Descriptions", AAAI Eighth National Conference on Artificial Intelligence, 1990.

Appendix A: List of publications resulting from this contract

W. Andersen, M. Evett, J. Hendler, and B. Kettler "Massively Parallel Matching of Knowledge Structures," in Parallel Artificial Intelligence, Kitano, H. and Hendler, J. (eds.), AAAI/MIT Press, 1994 ii. (earlier version in) Building and Knowledge Sharing, K. Fuchi and T. Yokio (eds), Ohmsha Ltd., Tokyo Japan, 1994

K. Erol, J. Hendler, and D. Nau "A Critical Look at Critics in HTN Planning", Proc. International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Aug, 1995.

K. Erol, J. Hendler, and D. Nau, "UMCP: A sound and complete procedure for HTN planning", Proc. of the 2nd International Conference on AI Planning Systems, June, 1994.

M. Evett, W. Andersen and J. Hendler "Providing Computationally Effective Knowledge Representation via Massive Parallelism," in Processing for AI, L. Kanal, V. Kumar, H. Kitano and C. Suttner (eds.) Elsevier, 1994

M. Evett, J. Hendler, A. Mahanti, D. Nau PRA*: The SIMD Parallelization of a Memory-Limited Heuristic Search Algorithm, J. Parallel and Distributed Computing, 25(2), 1995.

M. Evett, J. Hendler and L. Spector, Parallel Knowledge Representation on the Connection Machine, Journal of Parallel and Distributed Computing,22(2), 1994.

M. Evett,J. Hendler, and W. Andersen, "Massively Parallel Support for Computationally Effective Recognition Queries," Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93), AAAI Press, Menlo Park, CA, 1993

M. Evett, W. Andersen and J. Hendler, "Massively Parallel Support for Efficient Knowledge Representation," Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93), Morgan Kaufmann, 1993.

J. Hendler, K. Stoffel and A. Mulvehill. High Performance Support for Case-Based Planning Applications, in A. Tate (ed) Planning Technology, MIT/AAAI Press, Menlo Park, CA., USA, May 1996

J. Hendler, "Types of Planning — can artificial intelligence yield insights into prefrontal function?" in Boller and Grafman (eds.) Frontal Lobes – Annals of the New York Academy of Science, Vol 769, 1995 (p. 265-276).

J. Hendler, B. Kettler, W. Andersen, M. Evett, S. Kambhampati, and A. Agrawala, "Masively Parallel Support for Case-Based Planning," . Arpa/Rome Lab Knowledge-Based Planning and Scheduling Initiative, Morgan-Kaufmann, CA, Feb. 1994.

J. Hendler,S. Kambhampati, L. Ihrig, S. Katukam, J. Chen, and A. Agrawala."Integrated Approaches for improving the effectiveness of Plan Reuse," . Arpa/Rome Lab Knowledge-Based Planning and Scheduling Initiative, Morgan-Kaufmann, CA, Feb. 1994.

J. Hendler, Editorial: Experimental AI Systems, of Experimental and Theoretical AI, 7(1-2), 1995.

J. Hendler,D. Musliner and R. Kohout, Supporting Intelligent Real-Time Control: Dynamic Reaction on the Maruti Operating System, Proc. Israeli Symposium on Artificial Intelligence, Jerusalem, Israel, Jan 1995.

J. Hendler, High Performance Artificial Intelligence, Vol 265, Aug 12, 1994.

J. Hendler, Artificial Intelligence: yesterday, today and tomorrow, Currents in Modern Thought, World & I, Aug 1995.

L. Ihrig and S. Kambhampati. "Design and Implementa- tion of a Replay Framework based on a Partial order Planner." Proc. National Conference on Artificial Intelligence (AAAI-96), 1996.

L. Ihrig and S. Kambhampati. "Integrating Replay with EBL to improve planning performance." Current trends in AI Planning, IOS Press, 1995.

L. Ihrig and S. Kambhampati. "Derivation Replay for Partial-order Planning," Proc. 12th Natl. Conf. on Artificial Intelligence (AAAI-94), August 1994.

S. Katukam and S. Kambhampati. "Learning Explanation- based Search control rules for partial-order planning," Proc. 12th Natl. Conf. on Artificial Intelligence (AAAI- 94), August 1994.

S. Kambhampati, S. Katukam and Y. Qu. "Failure driven Dynamic Search Control for Partial Order Planners: An Explanation-based approach", Artificial Intelligence, July 1995.

S. Kambhampati, C. Knoblock and Q. Yang. Planning as Refinement Search: A Unified framework for evaluating design tradeoffs in partial order planning. Artificial Intelligence. Special issue on Planning and Scheduling. Vol. 76. No. 1-2, September 1995. pp. 167-238.

S. Kambhampati. Comparative analysis of Partial Order and HTN Planning. SIGART Bulletin Special section on Evaluation of Plans, Planners and Planning Agents. Vol. 6, No. 1, January, 1995. pp. 16-25.

S. Kambhampati and D.S. Nau, "On the nature and role of modal truth criteria in planning", Artificial Intelligence, 1995.

S. Kambhampati, "Multi-Contributor Causal Structures for Planning: A Formalization and Evaluation," Artificial Intelligence Vol. 69, No. 1-2, pp. 235-278.

S. Kambhampati and S. Kedar, "A Unified Framework for Explanation-Based Generalization of Partially Ordered and Partially Instantiated Plans," ASU CSE-TR-92-008, Artificial Intelligence, Vol 67, No. 2, June 1994. pp. 29-70.

S. Kambhampati. "Refinement planning: Status and Prospectus" In Proc. National Conference on AI, 1996.

S. Kambhampati. "AI Planning: A prospectus on theory and applications" Position Statement,ACM Computing Surveys, Symposium on Artificial Intelligence, September 1995.

S. Kambhampati and X. Yang. "On the role of Disjunctive representations and Constraint Propagation in Refinement Planning" Proc. of Knowledge Representation and Reason- ing, 1996.

S. Kambhampati, Formalizing Dependency Directed Back-tracking and Explanation-based Learning in Refinement Search. Proc. National Conference on Artificial Intelligence (AAAI-96), 1996.

S. Kambhampati, L. Ihrig and B. Srivastava. "A Candidate Set based analysis of subgoal interactions in conjunctive goal planning." In Proceedings of 3rd Intl. Conf. on AI Planning Systems, May 1996.

S. Kambhampati and B. Srivastava. "Universal Classical Planner: An Algorithm for Unifying State-space and Plan- space Planning," Current trends in AI Planning, IOS Press, 1995.

S. Kambhampati. "Admissible pruning strategies based on plan-minimality for plan-space planning" Proc. 14th Intl. Joint Confernce on Artificial Intelligence. August 1995.

S. Kambhampati. "Refinement Search as a unifying frame- work for analyzing planning algorithms," Proc. 4th Intl. Conf. on Principles of Knowledge Representation and Reasoning, May 1994.

S. Kambhampati. "Design Tradeoffs in partial order (plan-space) planning," Proc. 2nd Intl. Conf. on AI Planning Systems, June 1994.

S. Kambhampati and D.S. Nau. "On the nature of modal truth criteria in planning," Proc. 12th Natl. Conf. on Artificial Intelligence (AAAI-94), August 1994.

S. Kambhampati. "On the Utility of Systematicity: Under-standing Trade-offs between redundancy and commitment in partial-ordering planning," Proceedings of the 13th Intl. Joint Conf. on Artificial Intelligence, Chamberry, France.

B. Kettler, W. Andersen, J. Hendler, and M. Evett, Massively Parallel Support for Case-based Planning. IEEE Expert, February 1994

B. Kettler and J. Hendler, "Evaluating the CAPER Planning System," Proc. Israeli Symposium on Artificial Intelligence, Jerusalem, Israel, Jan 1995.

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

V. Manikonda, J. Hendler and P.S. Krishnaprasad "Formalizing Behavior-Based Planning for Nonholonomic Robots,"Proc. International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Aug, 1995.

V. Manikonda, J. Hendler and P.S. Krishnaprasad "Formalizing "A Motion Description Language and a Hybrid Architecture for Motion Planning with Nonholonomic Robots," Proc. International Conference on Robotics and Automation, Nagoya, Japan, 1995.

D. McDermott and J. Hendler,Planning: What it is, What it could be, Intelligence, 76, 1996.

D. Musliner, J. Hendler, A. Agrawala, E. Durfee and J. Strosnider, The Challenges of Real-Time AI, Computer, 28(1), January, 1995.

D. Nau, K. Erol, and J. Hendler, "HTN Planning: Complexity and Expressivity" Proc. 12th Natl. Conf. on Artificial Intelligence (AAAI-94), Seattle, WA. August 1994.

Y. Qu and S. Kambhampati. "Learning control rules for expressive plan-space planners: Factors influencing the performance." Current trends in AI Planning, IOS Press, 1995.

K. Stoffel, and J. Hendler, "Parka on MIMD-Supercomputers," in J. Geller (ed.) Parallel Processing in AI, 1996

K. Stoffel, J. Hendler and J. Saltz, "Parka on MIMD-Supercomputers," 3rd International Workshop on Parallel Processing in AI, Montreal, Aug., 1995.

L. Spector and J. Hendler, "The use of supervenience in Dynamic-World Planning," Proc. of the 2nd International Conference on AI Planning Systems, June, 1994.

R. Tsuneto, K. Erol, J. Hendler, and D. Nau "Commitment Strategies in Hierarchical Task-Network Planning", Proc, Thirteenth Natl. Conf. on Artificial Intelligence (AAAI-96)", Portland, OR August 1996.