Department of Computer Science

THE UNIVERSITY OF ARIZONA
TUCSON ARIZONA

Gould-Simpson Building
Tucson, Arizona 85721

RL-29

January 1992

Technical Report List

Abstracts of technical reports available from our department are listed below. If you would like to receive a copy of any of these documents, return the form at the end of this list. For those reports that are free, only one copy will be sent to one address because of the costs of printing and mailing. See the form at the end of the list for information concerning the purchase of additional copies.

TR 91-14 **Measuring the Overhead in Conservative Parallel Simulations of Multicomputer Programs: Detailed Measurements**
Mary L. Bailey and Michael A. Pagels
In this paper we show that it is feasible to characterize the overheads present in conservative parallel simulations of multicomputer programs. We use a modified version of the parallel simulator from the Poker Programming Environment to empirically measure the overhead in two parallel algorithms which use three different interconnection structures. We discuss the sources of overhead and qualitatively discuss their relative importance. (25 pages)

TR 91-15 **Supporting Valid Time in an Historical Relational Algebra: Proofs and Extensions**
Edwin McKenzie and Richard T. Snodgrass
We define a relational algebra for historical relations. This extension of the conventional relational algebra supports valid time, the time when an object or relationship in the enterprise being modeled is valid. Historical versions of the projection, selection, union, difference and cartesian product operators are defined, and a new operator that performs a combination of historical selection and projection, historical derivation, is introduced. Two additional operators support aggregates. The algebra is shown to be closed, to be relationally complete, to reduce to the conventional relational algebra, and to have the expressive power of the temporal query language TQuel. We examine query optimization, page structure, and incremental update of materialized views, demonstrating that this algebra may be efficiently implemented. We conclude with a comparison with other proposed historical algrebras. (63 pages)

TR 91-16 **The Implementation of an Optimizing Compiler for Icon**
Kenneth W. Walker
There are many optimizations that can be applied while translating Icon programs. These optimizations and the analyses needed to apply them are of interest for two reasons. First, Icon's unique combination of characteristics requires developing new techniques for implementing them. Second, these optimizations are useful in variety of languages and Icon can be used as a medium for extending the state of the art.

Many of these optimizations require detailed control of the generated code. Previous production implementations of the Icon programming language have been interpreters. The virtual machine code of an interpreter is seldom flexible enough to accommodate these optimizations and modifying the virtual machine to add the flexibility destroys the simplicity that justified using an interpreter in the first place. These optimizations can only reasonably be implemented in a compiler. In order to explore these optimizations for Icon programs, a compiler was developed. This dissertation describes the compiler and the optimizations it employs. It also describes a run-time system designed to support the analyses and optimizations. Icon variables are untyped. The compiler contains a type inferencing system that determines what values variables and expression may take on during program execution. This system is effective in the presence of values with pointer semantics and of assignments to components of data structures.

The compiler stores intermediate results in temporary variables rather than on a stack. A simple and efficient algorithm was developed for determining the lifetimes of intermediate results in the presence of goal-directed evaluation. This allows an efficient allocation of temporary variables to intermediate results.

The compiler uses information from type inferencing and liveness analysis to simplify generated code. Performance measurements on a variety of Icon programs show these optimizations to be effective. (118 pages)

TR 91-17 **A Fragment Assembly Project Environment**
Sandra Miller and Gene Myers

This document describes a prototype software environment in support of DNA sequencing projects. The system consists of three tools for inserting sequence data into a project, naming collections of sequences, and an X-windows based browser for viewing the assembled sequences. The underlying algorithms for assembling the data are those packaged in the fragment assembly suite of Kececioglu and Myers. The browser, called FAB, allows hierarchical navigation through the solution space, permits modification of the raw data with a multi-alignment editor, and supports version control of such modifications. (12 pages)

TR 91-18 **The Jade File System**
Herman Rao

File systems have long been the most important and most widely used form of shared permanent storage. File systems in traditional time-sharing systems such as Unix support a coherent sharing model for multiple users. Distributed file systems implement this sharing model in local area networks. However, most distributed file systems fail to scale from local area networks to an internet. This dissertation recognizes four characteristics of scalability: size, wide area, autonomy, and heterogeneity. Owing to size and wide area, techniques such as broadcasting, central control, and central resources, which are widely adopted by local area network file systems, are not adequate for an internet file system. An internet file system must also support the notion of autonomy because an internet is made up by a collection of independent organizations. Finally, heterogeneity is the nature of an internet file system, not only because of its size, but also because of the autonomy of the organizations in an internet.

This dissertation introduces the Jade File System, which provides a uniform way to name and access files in the internet environment. Jade is a logical system that integrates a heterogeneous collection of existing file systems, where heterogeneous means that the underlying file systems support different file access protocols. Because of autonomy, Jade is designed under the restriction that the underlying file systems may not be modified. In order to avoid the complexity of maintaining an internet-wide, global name space, Jade permits each user to define a private name space. In Jade's design, we pay careful attention to avoiding unnecessary network messages between clients and file servers in order to achieve acceptable performance. Jade's name space supports two novel features: It allows multiple file systems to be mounted under one directory, and it permits one logical name space to mount other logical name spaces.

A prototype of Jade has been implemented to examine and validate its design. The prototype consists of interfaces to the Unix File System, the Sun Network File System, and the File Transfer Protocol. (144 pages)

TR 91-19 **A Compositional Architecture for Portable, Scalable Distributed Operating Systems**
Peter Druschel

The design of an object-oriented architecture for distributed operating systems is proposed that allows applications to dynamically compose distributed services from a mixture of system, application, and third-party provided software components. By decoupling execution, protection and modularity, the architecture supports the configuration of scalable-distributed operating systems, and ensures portability over a wide range of hardware platforms. (12 pages)

TR 91-20 **Improving the Running Times for some String-Matching Problems**

Sun Wu, Udi Manber, and Eugene Myers

We present new algorithms for three basic string-matching problems: 1) An algorithm for approximate string matching of a pattern of size m in a text of size n in worst-case time $O(n + nm/\log n)$, and average-case time $O(n + nd/\log n)$, where d is the number of allowed errors. 2) An algorithm to find the edit distance between two sequences of size n and m $(n > m)$ in time $O(n + nd/\log n)$, where d is the edit distance. 3) An algorithm for approximate matching of a regular expression of size m in a text of size n in time $O(n + nm/\log_{d+2} n)$, where d is the number of allowed errors. The last algorithm is the first $o(mn)$ algorithm for approximate matching to regular expressions. (17 pages)

TR 91-21 **A Pattern Matching System for Biosequences**

Gerhard Mehldau

String pattern matching is an extensively studied area of computer science. Over the past few decades, many important theoretical results have been discovered, and a large number of practical algorithms have been developed for efficiently matching various classes of patterns. A variety of general pattern matching tools and specialized proogramming languages have been implemented for applications in areas such as lexical analysis, text editing, and database searching. Most recently, the field of molecular biology has been added to the growing list of applicaitons that make use of pattern matching technology.

The requirements of biological pattern matching differ from traditional applications in several ways. First, the amount of data to be processed is very large, and hence highly efficient pattern matching tools are required. Second, the data to be searched is obtained from biological experiments, where error rates of up to 5% are not uncommon. In addition, patterns are often averaged from several, biologially similar sequences. Therefore, to be useful, pattern matching tools must be able to accomodate some notion of approximate matching. Third, formal language notions such as regular expressions, which are commonly used in traditional applications, are insufficient for describing many of the patterns that are of interest to biologists. Hence, any conventional notation must be significantly enhanced to accomodate such patterns. Taken together, these differences combine to render most existing pattern matching tools inadquate, and have created a need for specialized pattern matching systems.

This dissertation presents a pattern matching system that specifically addresses the three issues outlined above. A notation for defining patterns is developed by extending the regular expression syntax in a consistent way. Using this notation, virtually any pattern of interest to biologists can be expressed in an intuitive and concise manner. The system further incorporates a very flexible notion of approximate pattern matching that unifies most of the previously developed concepts. Last, but not least, the system employs a novel, optimized backtracking algorithm, which enables it to efficiently search even very large databases. (109 pages)

TR 91-22 **Sabbatical in Japan: Collected Trip Reports**

Richard D. Schlichting

During the course of a sabbatical in Japan in 1990, the author visited 20 universities, industrial research laboratories, and government laboratories. This report collects together trip reports from a number of those visits. Statements made herein represent the personal opinion of the author. (27 pages)

TR 91-23 **Adapting AVS to Support Scientific Applications as Heterogeneous, Distributed Programs**

Patrick T. Homer and Richard D. Schlichting

Most scientific applications are currently structured as a series of computational steps, each of which is implemented by a separate program with files being used to transmit data between the steps. For example, a vectorized computation with graphical output may involve executing the computation on a remote supercomputer, transferring the output file over the Internet, and then viewing the results on a local workstation. Here, an alternative model is described in which the application is constructed as a heterogeneous, distributed program in order to improve facilities for user interaction. In this model, the application is structured as a collection of interacting processes (tasks) in which distribution and heterogeneity of machine architecture and programming language are handled transparently by a remote procedure call (RPC) mechanism. The specific focus here is on describing how AVS (Application Visualization System) from Stardent Computer, Inc. has been adapted to support this type of application using the Schooner RPC system. An example involving self-organizing neural nets is used to illustrate the details of this approach. (16 pages)

TR 91-24a **A Multi-Paradigm Programming Language for Constructing Fault-Tolerant, Distributed Systems**

Richard D. Schlichting and Vicraj Thomas

The design of FT-SR, a programming language oriented towards constructing fault-tolerant distributed systems, is presented. The language, which is based on the existing SR language, is unique in that it has been designed to be a multi-paradigm language that can support equally well any of the various programming paradigms that have been developed for this type of system. These paradigms include the object/action model, the restartable action paradigm, and the replicated state machine approach. The programming model underlying the design of the language is based on the concept of fail-stop atomic objects; such objects either execute operations as atomic actions, or fail in a detectable way. It is argued that this model forms a common link among the various paradigms and hence, is a realistic basis for a multi-paradigm language. An example program consisting of a data manager and its associated stable storage is also given; the manager is built using the restartable action paradigm, while the stable storage is structured using the replicated state machine approach. Finally, the implementation strategy for the language runtime system is discussed. (20 pages)

TR 91-25 **Temporal Specialization and Generalization**

Christian S. Jensen and Richard Snodgrass

A standard relation is two-dimensional with attributes and tuples as dimensions. A temporal relation contains two additional, orthogonal time dimensions, namely valid time and transaction time. Valid time records when facts are true in the modeled reality, and transaction time records when facts are stored in the temporal relation.

While, in general, there are no restrictions between the valid time and transaction time associated with each fact, in many practical applications the valid and transaction times exhibit more or less restricted interrelationships which define several types of specialized temporal relations. The paper examines five different areas where a variety of types of specialized temporal relations are present.

In application systems with multiple, interconnected temporal relations, multiple time dimensions may be associated with facts as they flow from one temporal relation to another. For example, a fact may have associated multiple transaction times telling when it was stored in previous temporal relations. The paper investigates several aspects of the resulting generalized temporal relations, including the ability to query a predecessor relation from a successor relation.

The presented framework for generalization and specialization allows researchers as well as database and system designers to precisely characterize, compare, and thus better understand temporal relations and the application systems in which they are embedded. The framework's comprehensiveness and its use in understanding temporal relations is demonstrated by placing previously proposed temporal data models within the framework. The practical relevance of the defined specializations and generalizations is illustrated by sample realistic applications in which they occur. The additional semantics of specialized relations are especially useful for improving the performance of query processing. (38 pages)

TR 91-26 **Exact and Approximation Algorithms for DNA Sequence Reconstruction**

John Dimitri Kececioglu

The DNA sequence in every human being is a text of three billion characters from a four letter alphabet; determining this sequence is a major project in molecular biology. The fundamental task biologists face is to reconstruct a long sequence given short fragments from unknown locations. These fragments contain errors, and may represent the sequence on one strand of the double-helix, or the reverse complement sequence on the other strand. The Sequence Reconstruction Problem is, given a collection F of fragment sequences and an error rate $0 \leq \varepsilon < 1$, find a shortest sequence S such that every fragment $F \in F$, or its reverse complement, matches a substring of S with at most $\varepsilon|F|$ errors.

Sequence Reconstruction is NP-complete. We decompose the problem into (1) constructing a graph of approximate overlaps between pairs of fragments, (2) selecting a set of overlaps of maximum total weight that induce a consistent layout of the fragments, (3) merging the overlaps into a multiple sequence alignment and voting on a consensus. A solution to (1) through (3) yields a reconstructed sequence feasible at error rate $2\varepsilon/(1-\varepsilon)$ and at most a factor $1/(1-\varepsilon)$ longer than the shortest reconstruction, given some assumptions on fragment error. We define a measure of the overlap in a reconstruction, show that maximizing the overlap minimizes the length, and that approximating (2) within a factor of α approximates Sequence Reconstruction within a factor of $(1-\varepsilon)\alpha$ under the overlap measure.

We construct the overlap graph for (1) in $O(\varepsilon N^2)$ time given fragments of total length N at error rate ε. We develop two exact and two approximation algorithms for (2). Our best exact algorithm computes an optimal layout for a graph of E overlaps and V fragments in $O(K(E + V \log V))$ time, where $K \leq 2^E$ is the size of the branch-and-bound search tree. Our best approximation algorithm computes a layout with overlap at least ½ the maximum in $O(V(E + V \log V) \log V)$ time. We construct the multiple sequence alignment and consensus sequence for (3) in $O(H^2 L + M + N)$ time given a layout with at most H mutually overlapping fragments, where L is the length of the alignment and M is the number of pairs of aligned characters in the overlaps.

We evaluate an implementation by comparing the computed reconstruction to the sampled sequence. For a random sequence of length 50,000 sampled at 10% error with 500 fragments of length 500, the software reconstructed the correct layout. For a human DNA sequence of length 50,000 containing 18 repeated elements of length 300 to 2,000 sampled as above at 5% error, the software found a shorter layout, though over 95% of the layout was correct. When covered by three or more fragments, the reconstructed sequence had less than 1 error in 5,000 characters, given input with 1 error in 10.

This is the first treatment of Sequence Reconstruction with inexact data and unknown complementarity. (137 pages)

TR 91-27 **On the Complexity of Dataflow Analysis of Logic Programs**

Saumya Debray

It is widely held that there is a correlation between complexity and precision in dataflow analysis, in the sense that the more precise an analysis algorithm, the more computationally expensive it must be. The details of this correspondence, however, appear to not have been explored extensively. This paper reports some results on this tradeoff in the context of Horn logic programs. A formal notion of the ‘‘precision’’ of an analysis algorithm is proposed, and this is used to characterize the worst case computational complexity of a number of dataflow analysis algorithms with different degrees of precision. (27 pages)

TR 91-28 **Compiler Optimizations for Low-level Redundancy Elimination: An Application of Meta-level Prolog Primitives**

Saumya Debray

Much of the work on applications of meta-level primitives in logic programs focusses on high-level aspects such as source-level program transformation, interpretation, and partial evaluation. In this paper, we show how meta-level primitives can be used in a very simple way for low-level code optimization in compilers. The resulting code optimizer is small, simple, efficient, and easy to modify and retarget. An optimizer based on these ideas is currently being used in a compiler that we have developed for Janus. (14 pages)

TR 91-29 **An Overview of Sequence Comparison Algorithms in Molecular Biology**

Eugene W. Myers

Molecular biologists frequently compare biosequences to see if any similarities can be found in the hope that what is true of one sequence either physically or functionally is true of its analogue. Such comparisons are made in a variety of ways, some via rigorous algorithms, others by manual means, and others by a combination of these two extremes. The topic of sequence comparison now has a rich history dating back over two decades. In this survey we review the now classic and most established technique: dynamic programming. Then a number of interesting variations of this basic problem are examined that are specifically motivated by applications in molecular biology. Finally, we close with a discussion of some of the most recent and future trends. (23 pages)

TR 91-30 **Temporal Indeterminacy**
Curtis E. Dyreson and Richard T. Snodgrass
In temporal indeterminacy, it is known that an event stored in a temporal database did in fact occur, but it is not known exactly when the event occurred. We present the "possible tuples" data model, in which each indeterminate event is represented with an interval that delimits when the event might have occurred, and a probability distribution over that interval. We extend the TQuel query language with constructs that specify the user's confidence in the underlying temporal data and in the relationships among that data. We provide a formal tuple calculus semantics, and show that this semantics reduces to the determinate semantics. We outline an efficient representation of temporal indeterminacy, and efficient query processing algorithms, demonstrating the practicality of our proposed approach. (35 pages)

TR 91-31 **A System for Pattern Matching Applications on Biosequences**
Gerhard Mehldau and Gene Myers
This paper presents the design and implementation of ANREP, a system for finding matches to network expressions composed of ‘‘spacers’’ and approximate matches to network expressions. A user specifies such patterns via a declarative, free-format, and strongly typed language called A that is presented here in a tutorial style through a series of progressively more complex examples. The sample patterns are for protein and DNA sequence motifs, the application domain for which ANREP was specifically created. ANREP provides a unified framework for almost all previously proposed motif patterns and extends them by providing approximate matching, a feature heretofore unavailable except for the limited case of individual sequences. The performance of ANREP is discussed and an appendix gives a concise specification of syntax and semantics. A portable C software package implementing ANREP is available via anonymous remote file transfer. (21 pages)

TR 91-32 **Consul: A Communication Substrate for Fault-Tolerant Distributed Programs**
Shivakant Mishra, Larry L. Peterson, and Richard D. Schlichting
This paper describes a communication substrate, called Consul, upon which fault-tolerant distributed systems can be built. Consul facilitates the development of such systems by providing various fault-tolerant services for constructing programs based on the replicated state machine approach. These services include a broadcast service, a membership service, and a recovery service. Consul is unique in two respects. First, its services are implemented using a collection of algorithms that exploit the partial ordering of messages exchanged in the system. Such algorithms are generally more efficient than those that depend on a total ordering of events. Second, its underlying architecture is configurable. That is, the application configures Consul from a collection of building block protocols, including protocols for ordering messages, detecting failure, agreeing on membership, and restoring the state of a failed process. The paper sketches Consul's architecture, presents the algorithms used by its protocols, and reports on the performance of an implementation in the *x*-Kernel. (33 pages)

TR 92-33 **On the Query Complexity of Learning and a Technique for Lower Bounds on Monotone Formulae**
Sampath Kannan
We consider the problem of learning parametrized concept classes with membership and equivalence queries. If C_n is the concept class being learned, we show that if equivalence queries can be made from a larger but still ‘reasonable’ hypothesis class, then there exist $O(n \log |C_n|)$ queries that exactly learn the target concept $c \in C_n$. For example, for any fixed polynomial $p(n)$, the concept class consisting of all boolean n-input circuits of size $p(n)$ can be learnt with polynomially many queries if equivalence queries are allowed to ask about circuits of size $O(q(n)p(n))$, where $q(n)$ is another polynomial in n. We also show that our results are best possible in terms of how big the hypothesis class needs to be and thereby give a way of deriving a lower bound of $\Omega(n^2)$ on the size of monotone formulae for majority and other boolean functions. This matches the best known lower bounds for majority. (10 pages)

Icon Newsletter

This newsletter contains information about the Icon programming language. Typical topics include reports on language design, implementation, and notices of new publications. It is issued aperiodically, two or three times a year.

Software Distributions

The Department distributes a variety of software in machine-readable form. Examples are the Icon programming language, SB-Prolog, SR, *x*-Kernel, and the Scorpion System. Information about the contents and availability of specific distributions can be obtained by checking the boxes listed on the next page. Most distributions are available for a nominal fee, which includes media and the associated documentation, or via anonymous FTP from cs.arizona.edu.

First copies of reports are free. Additional copies may be purchased for the amount indicated. Requests that require payment must be accompanied by a check or a prepaid purchase order in U.S. dollars for the proper amount payable to The University of Arizona. Free copies may be ordered by electronic mail to tr_libr@cs.arizona.edu.

Please send me the reports checked below: (RL-29)

report	*additional copies*
☐ TR 91-14	2.50
☐ TR 91-15	4.50
☐ TR 91-16	6.00
☐ TR 91-17	2.00
☐ TR 91-18	6.50
☐ TR 91-19	2.00
☐ TR 91-20	2.00
☐ TR 91-21	5.50
☐ TR 91-22	2.50
☐ TR 91-23	2.00
☐ TR 91-24a	2.00
☐ TR 91-25	3.00
☐ TR 91-26	6.50
☐ TR 91-27	2.50
☐ TR 91-28	2.00
☐ TR 91-29	2.50
☐ TR 91-30	3.00
☐ TR 91-31	2.50
☐ TR 91-32	3.00
☐ TR 91-33	1.50

☐ Please add my name to the distribution list for the Icon Newsletter

Please send me information on the following software distributions:

- ☐ The Icon programming language
- ☐ The SR programming language
- ☐ The SB-Prolog System
- ☐ The *x*-Kernel
- ☐ The Scorpion System