Department of Computer Science

THE UNIVERSITY OF
ARIZONA
TUCSON ARIZONA

Gould-Simpson Building
Tucson, Arizona 85721

RL-30-31

1992

Technical Report List

Abstracts of technical reports available from our department are listed below. Information for obtaining copies of these documents can be found at the end of this list.

TR 92-01 **Image Transfer: An End-to-End Design**

Charles J. Turner and Larry L. Peterson

The transfer of digital images between data archives and scientific workstations is likely to consume a significant amount of network bandwidth in the very near future. This paper examines the image transfer problem from an end-to-end perspective, that is, it describes a complete image transfer protocol that takes into account both the nature of digital imagery and the properties of the underlying network. Specifically, it describes a simple algorithm for encoding images into network packets in such a way that the receiver can recover from dropped packets without requiring the sender to retransmit them. Avoiding retransmissions has the advantages of improving response time and eliminating the need to buffer data at the sender. (21 pages)

TR 92-02 **A Language-Based Approach to Protocol Implementation**

Mark B. Abbott and Larry L. Peterson

Morpheus is special-purpose programming language that facilitates the efficient implementation of communication protocols. Protocols are divided into three categories, called *shapes*, so that they can inherit code and data structures based on their category; the programmer implements a particular protocol by refining the inherited structure. Morpheus optimization techniques reduce per-layer overhead on time-critical operations to a few assembler instructions even though the protocol stack is not determined until runtime. This supports divide-and-conquer simplification of the programming task by minimizing the penalty for decomposing complex protocols into combinations of simpler protocols. (22 pages)

TR 92-03 **Interactive Graph Layout: the Exploration of large Graphs**

Tyson R. Henry

Directed and undirected graphs provide a natural notation for describing many fundamental structures of computer science. Unfortunately graphs are hard to draw in an easy to read fashion. Traditional graph layout algorithms have focused on creating good layouts for the entire graph. This approach works well with smaller graphs, but often cannot produce readable layouts for large graphs.

This dissertation presents a novel methodology for viewing large graphs. The basic concept is to allow the user to interactively navigate through large graphs, learning about them in appropriately small and concise pieces. The motivation of this approach is that large graphs contain too much information to be conveyed by a single canonical layout. For a user to be able to understand the data encoded in the graph she must be able to carve up the graph into manageable pieces and then create custom layouts that match her current interests.

An architecture is presented that supports graph exploration. It contains three new concepts for supporting interactive graph layout: interactive decomposition of large graphs, end-user specified layout algorithms, and parameterized layout algorithms.

The mechanism for creating custom layout algorithms provides the non-programming end-user with the power to create custom layouts that are well suited for the graph at hand. New layout

algorithms are created by combining existing algorithms in a hierarchical structure. This method allows the user to create layouts that accurately reflect the current data set and her current interests.

In order to explore a large graph, the user must be able to break the graph into small, more manageable pieces. A methodology is presented that allows the user to apply graph traversal algorithms to large graphs to carve out reasonably sized pieces. Graph traversal algorithms can be combined using a visual programming language. This provides the user with the control to select subgraphs that are of particular interest to her.

The ability to Parameterize layout algorithms provides the user with control over the layout process. The user can customize the generated layout by changing parameters to the layout algorithm. Layout algorithm parameterization is placed into an interactive framework that allows the user to iteratively fine tune the generated layout.

As a proof of concept, examples are drawn from a working prototype that incorporates this methodology. (104 pages)

TR 92-04 **End User Controlled Interfaces: Creating Multiple View Interfaces for Data-Rich Applications**

Tyson R. Henry and Scott E. Hudson

Direct manipulation interfaces give the user the feeling that he is interacting directly with the data. This feeling is achieved by designing the interface to closely resemble the user's mental model of the data. Thus the success of the interface depends on how closely it matches the user's mental model: how closely the interface objects match and how closely the structure of the interface objects matches. Since the user's mental model may depend on his current interests and the current data set, there is not always a single ''best'' structure for the interface. This is especially true in interfaces to applications with many data objects---*data-rich* applications. This paper presents techniques for creating flexible interfaces that allow the user to customize the structure of the interface. These techniques allow the user to restructure the interface so it matches his current interests and his current mental model of the data. (10 pages)

TR 92-05 **Approximate Matching of Network Expressions with Spacers**

Gene Myers

Two algorithmic results are presented that are pertinent to the matching of patterns of interest in macromolecular sequences. The first result is an output sensitive algorithm for approximately matching network expressions, i.e., regular expressions without Kleene closure. This result generalizes the $O(kn)$ expected-time algorithm of Ukkonen for approximately matching keywords [Ukk85]. The second result concerns the problem of matching a pattern that is a network expression whose elements are approximate matches to network expressions interspersed with specifiable distance ranges. For this class of patterns, it is shown how to determine a backtracking procedure whose order of evaluation is optimal in the sense that its expected time is minimal over all such procedures. (18 pages)

TR 92-06 **Consul: A Communication Substrate for Fault-Tolerant Distributed Programs**

Shivakant Mishra

As human dependence on computing technology increases, so does the need for computer system dependability. This dissertation introduces Consul, a communication substrate designed to help improve system dependability by providing a platform for building fault-tolerant, distributed systems based on the replicated state machine approach. The key issues in this approach--ensuring replica consistency and reintegrating recovering replicas--are are addressed in Consul by providing abstractions called fault-tolerant services. These include a broadcast service to deliver messages to a collection of processes reliably and in some consistent order, a membership service to maintain a consistent systym-wide view of which processes are functioning and which have failed, and a recovery service to recover a failed process.

Fault-tolerant services are implemented in Consul by a unified collection of protocols that provide support for managing communication, redundancy, failures, and recovery in a distributed system. At the heart of Consul is Psync, a protocol that provides for multicast communication based on a context graph that explicitly records the partial (or causal) order of messages. This graph also serves as the basis for novel algorithms used in the ordering, membership, and recovery protocols.

The ordering protocol combines the semantics of the operations encoded in messages with the partial order provided by Psync to increase the concurrency of the application. Similarly, the membership protocol exploits the partial ordering to allow different processes to conclude that a failure has occurred at different times relative to the to the sequence of messages received, thereby reducing the amount of synchronization required. The recovery protocol combines checkpointing with the replay of messages stored in the context graph to recover the state of a failed process. Moreover, this collection of protocols is implemented in a highly-configurable manner, thus allowing a system builder to easily tailor an instance of Consul from this collection of building-block protocols.

Consul is built in the *x*-Kernel and executes standalone on a collection of Sun 3 workstations. Initial testing and performance studies have been done using two applications: a replicated directory and a distributed wordgame. These studies show that the semantic based order is more efficient than a total order in many situations, and that the overhead imposed by the checkpointing, membership, and recovery protocols is insignificant. (145 pages)

TR 92-07 **Multiple Calendar Support for Conventional Database Management Systems**

Michael D. Soo and Richard T. Snodgrass

We propose a solution to the problem of supporting a time-stamp attribute domain in conventional relational database management systems. In contrast to existing proposals, which assume that a single interpretation of time is sufficient for all users and applications, we develop a general solution that supports multiple interpretations of time. The main concept underlying this proposal is that the universal aspects of time are separated from the user dependent aspects, at both the query language and the architectural levels. The user dependent aspects are encapsulated in calendars and calendric systems, each of which are extendible by local site personnel. In this way, the available time support can be customized to local requirements. We briefly describe modifications to SQL2 that support multiple calendars and calendric systems. These modifications reduce the complexity of the language while simultaneously increasing its expressive power. Finally, we describe a set of tools that aid in the generation of calendars and calendric systems. This work can be viewed as a limited but practical application of research into extensible database management systems. (24 pages)

TR 92-08 **Anchors in Tournaments**

Sampath Kannan, Moni Naor, Steven Rudich

We define and study the graph-theoretic notion of an *anchor* in a tournament. An anchor in a tournament is a subset of the nodes of the tournament such that every pair of nodes outside the anchor is 'distinguished' at one of the nodes in the anchor. We show that every n-node tournament has an anchor of size at most $2n/3$ and that there are n-node tournaments where the anchor size is at least $n/2$. We also show that finding a minimum-sized anchor in a tournament is NP-Complete. Finally we point out potential applications of anchors and related concepts to the design of efficient algorithms for tournament isomorphism. (8 pages)

TR 92-09 **SR: A Language for Parallel and Distributed Programming**

Ronald A. Olsson, Gregory R. Andrews, Michael H. Coffin, Gregg M. Townsend

This paper introduces the newest version of the SR concurrent programming language and illustrates how it provides support for different execution environments, ranging from shared-memory multiprocessors to distributed systems. SR uses a few well-integrated mechanisms for concurrency to provide flexible, yet efficient, support for parallel and distributed programming. This paper gives several realistic examples to illustrate these language mechanisms. (12 pages)

TR 92-10 **Bounds and Approximations for Overheads in the Time to Join Parallel Forks**

Peter J. Downey

This paper studies the effects of overheads in massively parallel processing. The execution times for tasks on individual processors are modeled as independent and identically but arbitrarily distributed random variables. The time to execute a process fork is assumed to be distributed exponentially. The main result bounds (in distribution and expectation) the overhead time in forking a large number of tasks across n machines and then waiting for the join event. The model used is appropriate for massive parallelism (when n is large): in fact the bound serves as a heavy traffic limit approached as $n \to \infty$. In this model, the expected total time to reach the final join consists of a forking overhead that grows linearly with the number of processors n, a time for parallel execution of tasks that

decreases in n, and finally a synchronization delay that is a concave sublinear function of n (no worse than $n^{1/2}$). This dependence on n means that there is a limit to n past which speed-up is impossible and performance degrades. An interesting aspect of the analysis is that the original problem reduces to a previously studied problem in queueing theory: estimating total delay in an infinite-server resequencing system. The new results thus provide new bounds and approximations in the theory of $M/G/\infty$ resequencing queues. (20 pages)

TR 92-11 **High-Performance Cross-Domain Data Transfer**

Peter Druschel and Larry L. Peterson

We have designed and implemented an operating system facility for high-performance cross-domain data transfer on uniprocessors and shared-memory multiprocessors. This facility, called *fast buffers* (fbufs), is designed to take advantage of modern RISC architectures that support large virtual address spaces. Its goal is to help deliver the high-bandwidth afforded by emerging high-speed networks to user-level processes, both in monolithic and small-kernel based operating systems. (19 pages)

TR 92-12 **Approximate Regular Expression Pattern Matching with Concave Gap Penalties**

James Knight and Gene Myers

Given a sequence A of length M and a regular expression R of length P, an approximate regular expression pattern matching algorithm computes the score of the best alignment between A and one of the sequences exactly matched by R. There are a variety of schemes for scoring alignments. In a concave gap-penalty scoring scheme, a function $\delta(a,b)$ gives the score of each aligned pair of symbols a and b, and a *concave* function $w(k)$ gives the score of a sequence of unaligned symbols, or gap, of length k. A function w is concave if and only if it has the property that for all $k > 1$, $w(k+1) - w(k) \leq w(k) - w(k-1)$. In this paper we present an $O(MP(\log M + \log^2 P))$ algorithm for approximate regular expression matching for an arbitrary δ and any concave w. (25 pages)

TR 92-13 **Computer Science in Japanese Universities**

David Notkin and Richard D. Schlichting

This paper presents some impressions of computer science in Japanese universities based on the authors' sabbatical visits. The focus is primarily on such structural aspects of the discipline as departmental organization, faculty and student populations, funding, research activity, and computing facilities. Perhaps the key observation is that Japanese cultural practices influence the way in which computer science is approached in Japanese universities to a larger degree than might be imagined. (22 pages)

TR 92-14 **Estimating the Number of Solutions for Constraint Satisfaction Problems**

Nai-Wei Lin

Knowledge about the number of solutions of constraint satisfaction problems can be used to improve the efficiency of logic programs and deductive databases, e.g., to control task granularity in parallel systems, to map predicates to processors in distributed systems, or to plan the evaluation order of body goals and the instantiation order of variable values. Many constraint satisfaction problems are NP-complete. Thus it is very unlikely to find an efficient algorithm to compute the number of solutions for constraint satisfaction problems. This paper presents a greedy approximation algorithms for estimating the number of solutions for constraint satisfaction problems on finite domain. The time complexity of this algorithm is $O(n^3 m^3)$ for a problem involving n variables and m domain values. Based on this algorithm, a set of flexible algorithms is also developed. User can choose the appropriate algorithm according to the efficiency and precision requirements. (15 pages)

TR 92-15 **Unification of Temporal Data Models**

Christian S. Jensen, Michael D. Soo, and Richard T. Snodgrass

To add time support to the relational model, both first normal form (1NF) and non-1NF approaches have been proposed. Each has associated difficulties. Remaining within 1NF when time support is added may introduce data redundancy. The non-1NF models may not be capable of directly using existing relational storage structures or query evaluation technologies. This paper describes a new, conceptual temporal data model that better captures the time-dependent semantics of the data while permitting multiple data models at the representation level. This conceptual model effectively moves the distinction between the various existing data models from a semantic basis to a physical,

performance-relevant basis.

We define a conceptual notion of a bitemporal relation where tuples are stamped with sets of two-dimensional chronons in transaction-time/valid-time space. Next, we describe three representation schemes: a tuple-timestamped 1NF representation, a backlog relation composed of 1NF time-stamped change requests, and a non-1NF attribute value-timestamped representation. We further investigate several variants of these representations. We use snapshot equivalence to relate the three representational data models with the conceptual bitemporal data model.

We then consider querying within the two-level framework. To do so, we first define an algebra at the conceptual level. We proceed to map this algebra to the representation level in such a way that new operators compute equivalent results for different representations of the same conceptual bitemporal relation. This demonstrates that all of these representations are faithful to the semantics of the conceptual data model, with many choices available that may be exploited to achieve improved performance. (26 pages)

TR 92-16 **Time-stamp Semantics and Representation**

Curtis E. Dyreson

Many database management systems and operating systems provide support for time values. At the physical level time values are known as time-stamps. A time-stamp has a physical realization and a temporal interpretation. The physical realization is a pattern of bits while the temporal interpretation is the meaning of each bit pattern, that is, the time each pattern represents. All previous proposals defined time-stamps in terms of seconds. However, as we show, there are at least seven definitions of this fundamental time unit. We propose a more precise temporal interpretation, the base-line clock, that constructs a time-line by using different well-defined clocks in different periods. We also propose time-stamp formats for events, intervals, and spans. These formats can represent all of time to the granularity of a second, and all of recorded history to a finer granularity of a microsecond.

Our proposed formats were designed to be more space and time efficient than existing representations. We compare our formats with those used in common operating systems and database management systems. (41 pages)

TR 92-17 **Extending Normal Forms to Temporal Relations**

Christian S. Jensen, Richard T. Snodgrass, and Michael D. Soo

Normal forms play a central role in the design of relational databases. Recently several normal forms for temporal relational databases have been proposed. The result is a number of isolated and sometimes contradictory contributions that only apply within specialized settings. This paper attempts to rectify this situation. We define a consistent framework of temporal equivalents of all the important conventional database design concepts: functional and multivalued dependencies, primary keys, and third, Boyce-Codd, and fourth normal forms. This framework is enabled by making a clear distinction between the logical concept of a temporal relation and its physical representation. As a result, the role played by temporal normal forms during temporal database design closely parallels that of normal forms during conventional database design. We compare our approach with previously proposed definitions of temporal normal forms and temporal keys. To demonstrate the generality of our approach, we outline how normal forms and dependency theory can also be applied to spatial databases, as well as to spatial-temporal databases. (24 pages)

TR 92-18 **The Run-Time Implementation Language for Version 8.7 of Icon**

Kenneth Walker

This report describes a language, RTL, to aid in the implementation of a run-time system for the Icon language. The design of RTL is motivated by the needs of an optimizing compiler for Icon. The optimizing compiler needs a variety of information about run-time routines and needs to be able to use both general and specialized versions of these routines to generate efficient code. RTL code is used to produce both a link library and a data base of information for use by the compiler. (41 pages)

TR 92-19 **Abstractions for Constructing Dependable Distributed Systems**

Shivakant Mishra and Richard D. Schlichting

Distributed systems, in which multiple machines are connected by a communications network, are often used to build highly dependable computing systems. However, constructing the software

required to realize such dependability is a difficult task since it requires the programmer to build fault-tolerant software that can continue to function despite failures. To simplify this process, canonical structuring techniques or programming paradigms have been developed, including the object/action model, the primary/backup approach, the state machine approach, and conversations. In this paper, some of the system abstractions designed to support these paradigms are described. These abstractions, which are termed fault-tolerant services, can be categorized into two types. One type provides functionality similar to standard hardware or operating system services, but with improved semantics when failures occur; these include stable storage, atomic actions, resilient processes, and certain kinds of remote procedure call. The other type provides consistent information to all processors in a distributed system; these include common global time, group-oriented multicast, and membership services. In addition to describing the fundamental properties of these abstractions and their implementation techniques, a hierarchy highlighting common dependencies between services is presented. Finally, a number of systems that use these abstractions are overviewed, including the Advanced Automation System (AAS), Argus, Consul, Delta-4, ISIS, and MARS. (41 pages)

TR 92-20 **Modularity in the Design and Implementation of Consul**

Shivakant Mishra, Larry L. Peterson and Richard D. Schlichting

Many applications constructed as Autonomous Decentralized Systems require high dependability, often leading to the use of distributed architectures and their associated fault-tolerance techniques. Consul is a system designed to support the use of such techniques in the construction of fault-tolerant, distributed systems structured according to the state machine approach. Here, the way in which modularity has been used in the design and implementation of Consul is described. Our approach to this issue makes it easy to configure a system customized to the needs of a specific application, as well as facilitating the development of the individual components that make up Consul. (12 pages)

TR 92-21 **Approximate Pattern Matching and Its Applications**

Sun Wu

Pattern matching is one of the most important problems in computer science. It is used in a wide range of applications including text editors, information retrieval, compilers, command interpreters, computational biology, and object recognition. In some instances, however, the pattern and/or the data file may contain errors and approximate matching rather than exact matching is required.

In this thesis, we study approximate pattern matching problems. Our study is based on the Levenshtein distance model, where errors considered are 'insertions', 'deletions', and 'substitutions'. In general, given a text string, a pattern, and an integer k, we want to find substrings in the text that match the pattern with no more than k errors. The pattern can be a fixed string, a limited expression, or a regular expression. The problem has different variations with different levels of difficulties depending on the types of the pattern as well as the constraint imposed on the matching.

We present new results both of theoretical interest and practical value. We present a new algorithm for approximate regular expression matching, which is the first to achieve a sub-quadratic asymptotic time for this problem. For the practical side, we present new algorithms for approximate pattern matching that are very efficient and flexible. Based on these algorithms, we developed a new software tool called 'agrep', which is the first general purpose approximate pattern matching tool in the UNIX system. 'agrep' is not only usually faster than the UNIX 'grep/egrep/fgrep' family, it also provides many new features such as searching with errors allowed, record-oriented search, AND/OR combined patterns, and mixed exact/approximate matching. 'agrep' has been made publicly available through anonymous ftp from cs.arizona.edu since June 1991. (92 pages)

TR 92-22 **An Overview of the Temporal Query Language TQuel**

Richard T. Snodgrass

We provide an overview of the previous decade of research on the temporal query language TQuel. TQuel is a minimal extension to Quel, the query language for Ingres. TQuel supports valid time and transaction time. Unlike many other temporal query languages, it supports aggregates, valid-time indeterminacy (where it is not known exactly when an event occurred), database modification, and

schema evolution. We discuss all of these aspects, first informally through examples, then formally by presenting their tuple calculus semantics.

We start with the language itself, building incrementally from the core constructs to the more advanced features, in almost three dozen example TQuel statements. We follow the same approach in presenting the formal semantics. We present tuple calculus equivalents both for generic TQuel statements and for several examples.

TQuel is based on the predicate calculus. To execute a query, a procedural equivalent is required. We define a temporal algebra, again incrementally, and give several important properties, such as closure, completeness, and reducibility to the snapshot algebra. We also show how each TQuel statement can be mapped into the algebra. Finally, we discuss two important topics in implementing the temporal algebra: query optimization, specifically the applicability of existing optimization strategies, and the physical structure of pages storing temporal tuples. The paper ends with a BNF syntax of TQuel that incorporates all of these features. (50 pages)

TR 92-23 **A Model for Incorporating Rule Based Knowledge into Direct Manipulation User Interfaces via Snapping**

Scott E. Hudson and Andrey K. Yeatts

The graphical interaction technique of snapping has a natural condition/action semantics similar to that of rules in rule based knowledge systems. This correspondence lends itself well to both new interaction techniques and to the implementation of direct manipulation interfaces directly from rule knowledge. In this paper, we look at the application of these methods to the construction of user interface builders. The approaches taken to build such systems have been concentrated either towards direct manipulation editors or to systems that attempt to automatically generate much or all of the interface. We consider how a middle ground between these approaches might be constructed, by explicitly representing the results of inference rules in the direct manipulation framework and by using semantic snapping techniques to give the user direct feedback and interactive control over the application of rules. (15 pages)

TR 92-24 **Automated Integration of Communication Protocol Layers**

Mark B. Abbott and Larry L. Peterson

Integrating protocol data manipulations is a strategy to increase the throughput of network protocols. This paper presents a program structure that can be used to integrate a broad range of protocols. This program structure provides a basis for the automatic synthesis of integrated protocols, and serves as a guide to custom, hand-coded implementations. In addition to describing this program structure and showing how integrated protocols can be synthesized, the paper also examines the performance limits and scalability of strictly layered versus integrated protocol implementations. (33 pages)

TR 92-25 **Experience with Modularity in Consul**

Shivakant Mishra, Larry L. Peterson, and Richard D. Schlichting

The use of modularity in the design and implementation of complex software simplifies the development process, as well as facilitating the construction of customized configurations. This paper describes our experience using modularity in Consul, a communication substrate used for constructing fault-tolerant, distributed programs. First, Consul is presented as an example of how modularity is feasible in both the design and the implementation of such systems. Second, modularity issues that arose during development are discussed. These include deciding how the system is divided into various modules, dealing with problems that result when protocols are combined, and ensuring that the underlying object infrastructure provides adequate support. The key observation is that dependencies between modules---both direct dependencies caused by one module explicitly using another's operation and indirect dependencies where one module is affected by another without direct interaction---make modularization especially difficult in systems of this type. While our observations are based on designing and implementing Consul, the lessons are applicable to any fault-tolerant, distributed system. (15 pages)

TR 92-26 **X-Icon: An Icon Window Interface; Version 2**

Clinton L. Jeffery

This document describes the calling interface and usage conventions of X-Icon, a window system interface for the Icon programming language that supports high-level graphical user interface programming. It presently runs on UNIX and VMS systems under Version 11 of the X Window System, and on OS/2 2.0 under Presentation Manager. (50 pages)

TR 92-27 **How Using Busses in Multicomputer Programs Affects Conservative Parallel Simulation: Detailed Measurements**

Mary L. Bailey, Michael A. Pagels, and Kachung Kevin Wong

In this paper we consider the effect of using a bus interconnection structure on the overheads present in conservative parallel simulations of multicomputer programs. We use a modified version of the Poker Programming Environment to empirically measure the overhead in two parallel algorithms using buses. We discuss the sources of overhead and compare them with those found using point-to-point communication. Preliminary results indicate that the overheads encountered using a bus interconnection structure were not predicted by our previous results using point-to-point communications. (18 pages)

TR 92-28 **Engineering Efficient Code Generators using Tree Matching and Dynamic Programming**

Christopher W. Fraser, David R. Hanson and Todd A. Proebsting

Many code generator generators use tree pattern matching and dynamic programming. This note describes a simple program that generates matchers that are fast, compact, and easy to understand. It is simpler than common alternatives: 200-700 lines of Icon versus 3000 lines of C for Twig and 5000 for `burg`. Its matchers run up to 25 times faster than Twig's. They are necessarily slower than `burg`'s BURS (bottom-up rewrite system) matchers but they are more flexible and still practical. (12 pages)

TR 92-29 **Super-pattern Matching**

James R. Knight and Eugene W. Myers

This paper presents a class of problems which has direct application to the problem of gene recognition in molecular biology and indirect applications to such problems as handwriting and speech recognition. These problems involve finding matches to patterns of patterns or *super-patterns*, given solutions to the lower-level patterns. The expressiveness of this problem class rivals that of traditional artifical intelligence characterizations, and yet polynomial time algorithms are described for each problem in the class. (26 pages)

TR 92-30 **A Software Platform for Constructing Scientific Applications from Heterogeneous Resources**

Patrick T. Homer and Richard D. Schlichting

Support for heterogeneous processing is useful for increasing the functionality available to designers of scientific applications. For example, rather than implement an application requiring remote vector processing and local visualization as two separate programs, such support allows an alternative structure in which the application is a single logical program with transparent transfer of control and data between phases. In addition to being simpler and more intuitive, such structuring makes it feasible to enhance the way in which users interact with the application to do, for instance, model steering. Here, a software platform that facilitates the construction of this type of scientific application is described. Its key component is Schooner, an interconnection system that includes an intermediate data representation, a simple specification language, and a heterogeneous remote procedure call (RPC) facility; to provide sophisticated visualization capabilities and an execution framework, AVS is included as well. Two applications built using this platform, one from molecular dynamics and the other involving neural nets, are also described. One important conclusion is that enhanced monitoring and interaction facilities impose very little overhead for applications such as these. (28 pages)

TR 92-31 **FT-SR: A Programming Language for Constructing Fault-Tolerant Distributed Systems**

Richard D. Schlichting and Vicraj T. Thomas

The design and implementation of FT-SR, a programming language oriented towards constructing fault-tolerant distributed systems, is described. The language, which is based on the existing SR language, is unique in that it has been designed to support equally well any of the programming paradigms that have been developed for this type of system, including the object/action model, the restartable action paradigm, and the state machine approach. To do this, the language is designed to support the implementation of systems modeled as collections of fail-stop atomic objects. Such objects execute operations as atomic actions except when a failure or series of failures cause underlying implementation assumptions to be violated; in this case, notification is provided. It is argued that this model forms a common link among the various paradigms and hence, is a realistic basis for a language designed to support the construction of systems that use any or all of these approaches. An example program consisting of a data manager and its associated stable storage is given; the manager is built using the restartable action paradigm, while the stable storage is structured using the replicated state machine approach. Finally, an implementation of the language that uses the *x*-kernel and runs standalone on a network of Sun workstations is discussed. (28 pages)

TR 92-32 **Window Interface Tools for X-Icon**

Jon M. Lipp

This document describes the WIT library, a user interface tool kit available to Icon programmers using the window interface available in Version 8.7 of Icon. The tool kit extends the basic X-Icon window model by providing for layout of and event routing to virtual input/output devices attached to windows. (42 pages)

TR 92-33 **Proceedings of *x*-kernel Workshop**

November 11, 1992

The "*x*-kernel Workshop 1992" was an effort to unite researchers working with the *x*-kernel. The *x*-kernel, originally an operating system in its own right, has become a communications infrastructure ready for grafting onto systems such as Mach. These proceedings are a collection of slides and papers from the workshop discussing the current state, recent work, proposed extensions, licensing issues, and useful experiences with the *x*-kernel. (150 pages)

TR 92-34 **XIB: X-Icon Interface Builder**

Mary Cameron

This document describes XIB, a user interface builder used to specify and generate graphical user interfaces for X-Icon. XIB allows quick prototyping of user interface designs: Designs can easily be created, tested, and modified in a ‘‘what you see is what you get’’ environment. To provide maximum flexibility with respect to tool and interface creation, interfaces can be generated with or without an underlying tool. XIB is essentially a front-end for the Icon WIT library. (14 pages)

TR 92-35 **End-User Specification of Interactive Displays**

Shamim P. Mohamed

Representing data graphically can often increase its understandability. However most visualization systems to date have allowed users to only choose from a small number of pre-defined display methods. Also, these methods tend to be closely associated with the underlying application. We propose techniques where the user can specify the displays to be drawn in a flexible manner. The specification takes the form of a visual language, and represents data that has been selected by a query process. (125 pages)

TR 92-36 **A Sub-quadratic Algorithm for Approximate Limited Expression Matching**

Sun Wu, Udi Manber, and Gene Myers

In this paper we present an efficient sub-quadratic time algorithm for matching strings and limited expressions in large texts. Limited expressions are a subset of regular expressions that appear often in practice. The generalization from simple strings to limited expressions has a negligible affect on the speed of our algorithm, yet allows much more flexibility. Our algorithm is similar in spirit to that of Masek and Paterson, but it is much faster in practice. Our experiments show a factor of 4 to

5 speedup against the algorithms of Sellers and Ukkonen independent of the sizes of the input strings. Experiments also reveal our algorithm to be faster, in most cases, than a recent improvement by Chang and Lampe, especially for small alphabet sizes for which it is 2 to 3 time faster. (18 pages)

TR 92-37 **Combinatorial algorithms for DNA sequence assembly**

John D. Kececioglu and Eugene W. Myers

The trend towards very large DNA sequencing projects, such as those being undertaken as part of the human genome initiative, necessitates the development of efficient and precise algorithms for assembling a long DNA sequence from the fragments obtained by shotgun sequencing or other methods. The sequence reconstruction problem that we take as our formulation of DNA sequence assembly is a variation of the shortest common superstring problem, complicated by the presence of sequencing errors and reverse complements of fragments. Since the simpler superstring problem is NP-hard, any efficient reconstruction procedure must resort to heuristics. In this paper, however, a four phase approach based on rigorous design criteria is presented, and has been found to be very accurate in practice. Our method is robust in the sense that it can accommodate high sequencing error rates and list a series of alternate solutions in the event that several appear equally good. Moreover it uses a limited form of multiple sequence alignment to detect, and often correct, errors in the data. Our combined algorithm has successfully reconstructed non-repetitive sequences of length 50,000 sampled at error rates of as high as 10 percent. (45 pages)

TR 92-38a **Supporting Heterogeneity and Distribution in the Numerical Propulsion System Simulation Project**

Patrick T. Homer and Richard D. Schlichting

The Numerical Propulsion System Simulation (NPSS) project has been initiated by NASA to explore the use of computer simulation in the development of new aircraft propulsion technology. With this approach, each engine component is modeled by a separate computational code, with a simulation executive connecting the codes and modeling component interactions. Since each code potentially executes on a different machine in a network, a simulation run is a heterogeneous distributed program in which diverse software and hardware elements are incorporated into a single computation. In this paper, a prototype simulation executive that supports this type of programming is described. The two components of this executive are the AVS scientific visualization system and the Schooner heterogeneous remote procedure call (RPC) facility. In addition, the match between Schooner's capabilities and the needs of NPSS is evaluated based on our experience with a collection of test codes. The basic conclusion is that, while Schooner fared well in general, it exhibited certain deficiencies that dictated changes in its design and implementation. This discussion not only documents the evolution of Schooner, but also serves to highlight the practical problems that can be encountered when dealing with heterogeneity and distribution in such applications. (18 pages)

Technical Reports are available via anonymous FTP from cs.arizona.edu in the **reports** directory or via electronic mail by sending a message to ftpmail@cs.arizona.edu containing the word **help**. Questions may be directed to tr_libr@cs.arizona.edu.

Because of the costs of printing and mailing, only one free hard copy will be sent to an address. 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 also be ordered via electronic mail to tr_libr@cs.arizona.edu.

Please send me the reports checked below: (RL-30-31)

report	*additional copies*	*report*	*additional copies*
☐ TR 92-01	$2.50		
☐ TR 92-02	$2.50		
☐ TR 92-03	$5.50		
☐ TR 92-04	$1.50		
☐ TR 92-05	$2.00		
☐ TR 92-06	$6.50		
☐ TR 92-07	$2.50		
☐ TR 92-08	$1.50		
☐ TR 92-09	$2.00		
☐ TR 92-10	$2.00		
☐ TR 92-11	$2.00		
☐ TR 92-12	$2.50		
☐ TR 92-13	$2.50		
☐ TR 92-14	$2.00		
☐ TR 92-15	$2.50		
☐ TR 92-16	$3.50		
☐ TR 92-17	$2.50	☐ TR 92-28	$2.00
☐ TR 92-18	$3.50	☐ TR 92-29	$2.50
☐ TR 92-19	$3.50	☐ TR 92-30	$2.50
☐ TR 92-20	$2.00	☐ TR 92-31	$2.50
☐ TR 92-21	$5.50	☐ TR 92-32	$3.50
☐ TR 92-22	$3.50	☐ TR 92-33	$6.50
☐ TR 92-23	$2.00	☐ TR 92-34	$2.00
☐ TR 92-24	$3.00	☐ TR 92-35	$6.00
☐ TR 92-25	$2.00	☐ TR 92-36	$2.00
☐ TR 92-26	$3.50	☐ TR 92-37	$3.50
☐ TR 92-27	$2.00	☐ TR 92-38a	$2.00

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.

☐ 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

Technical Librarian
Department of Computer Science
The University of Arizona
Gould-Simpson 721
Tucson, Arizona 85721
USA