FESTUS OLUBUKUNMI AJIBUWA

School of Science and Engineering

ALTERNATIVE COMPUTING PARADIGMS

A Research work
Submitted as partial fulfillment of academic requirements for:

ATLANTIC INTERNATIONAL UNIVERSITY

Alternative Computing Paradigms

Abstract

This research work examines the future of computers, considers Integrated Circuit technology in the areas of how it drives computer design, and what the fundamental limitations are. Examines the proposed alternatives, including neurobiologically inspired computing, DNA computing, Tuple Board Paradigm, Mobile Computing and Quantum Computing. Views of different schools of thoughts, organizations, companies and Universities are critically assessed and compiled within this write-up.

Nowadays computers are in the centre of almost all areas of modern society and their computational abilities very often determine superiority of one product with respect to another. Traditional computers are able to deal with only finite numbers because arithmetic developed for infinite numbers leave undetermined many operations (e.g., ∞-∞) or represent infinite numbers by infinite sequences of finite numbers (e.g., non-standard analysis approaches). Thus, in spite of the key role of infinitesimals and infinite in Physics and Mathematics (e.g., derivatives, integrals, and differential equations), the parts of natural sciences related to infinite and infinitesimals remain purely theoretical fields.
The new unconventional approach proposed very recently by the author uses a new computational paradigm (that is not related to traditional non-standard analysis approaches) for creating a new revolutionary type of computer – Infinity Computer – able to store infinite, finite, and infinitesimal numbers and to execute arithmetical operations with all of them. The key methodological idea is usage of a new positional system with infinite radix allowing one to express finite, infinite, and infinitesimal numbers in a unique framework by a finite number of symbols.
The Infinity Computer will have a very high impact in Computer Science, Mathematics, Physics, and Economics because: it gives possibilities to execute calculations of a new type; leads to infinite accuracy of calculus; allows people to study complex systems where infinitesimal changes of parameters lead to finite and infinite changes in the output; simplifies scientific fields (and their teaching at schools and universities) where usage of infinity and infinitesimals is necessary.

Toward a new computing paradigm: Views of Microsoft Developer Div chief architect David Vaskevitch (Enterprise Systems Journal) July, 1995

We live in interesting times. As with any era of profound change, we find ourselves in need of a new paradigm -- a paradigm not just of computing but of human relationships on a global scale. But a paradigm is not something you simply define in advance; it is marked by a cultural shift in the shared mindset. Its pieces evolve slowly until it reaches a critical mass and points the way into the new era. We are at that critical point: Information technology, already undergoing tremendous change as a result of the business process reengineering movement, has entered into a new era.

The need for a new computer paradigm

The CHAOS-Based Systems
A New Computing Paradigm: (Chaos-Based System That "Evolves" Answers May Be Alternative to Current Computers) *Science Daily*

A revolutionary new computing technique that uses a network of chaotic elements to "evolve" its answers could provide an alternative to the digital computing systems widely used today. Described for the first time in the September 7 issue of Physical Review Letters this "dynamics-based computation" may be well suited for optical computing using ultra-fast chaotic lasers and computing with silicon/neural tissue hybrid circuitry.

The system has so far demonstrated an ability to handle a wide range of common operations, including addition and multiplication, as well as Boolean logic and more sophisticated operations such as finding the least common multiplier in a sequence of integers. Because it depends on interaction among its coupled elements, the system is naturally parallel.

"We have shown that this can be done, but we've only seen the tip of the iceberg," said Dr. William L. Ditto, professor of physics at the Georgia Institute of Technology. "This is a glimpse of how we can make common dynamic systems work for us in a way that's more like how we think the brain does computation. It's an entirely new computing paradigm."

For many years, scientists have observed the rich variety of behavioral patterns created by chaotic systems, including those found in living organisms. Ditto and collaborator Sudeshna Sinha of the Institute of Mathematical Sciences in Madras, India, reasoned that these natural chaotic systems should have been eliminated through evolution unless they served a purpose. Ditto and Sinha devised an experiment to see if a simple network of chaotic computer elements could be made to handle computations. They joined the chaotic elements into a lattice using an adaptive connecting mechanism that would open whenever an element exceeded a certain critical value. The mechanism was designed so that the researchers could set a wide range of critical values to vary the connection between elements.

The researchers then encoded values into the chaotic lattice using a variety of different techniques. In some cases, they chose patterns in the chaotic elements to represent numbers. In other cases, the numbers were represented by the amplitude of waves emitted by the chaotic elements or the frequency of "spiking" behavior.

After encoding the numbers, they stimulated the elements to begin interacting.

Elements containing values above the critical level triggered the connecting mechanism, allowing the excess value to "avalanche" into neighboring elements. That transfer then created additional avalanches in other connected elements. With additional stimulation, the domino effect continued until the imbalance was conducted out of the system -- as the answer to the mathematical problem.

"We have the elements interconnected so that they respond to their neighbors like the avalanching that occurs when you pile grains of sand onto a sandpile," Ditto explained. "You allow the elements to avalanche and the system to evolve chaotically, and then do the avalanching again until the system settles down to the right answer. It takes a couple of iterations for it to settle down."

In a simple example, values of three and four would be encoded into a system set with a critical value of one. The values would create an imbalance that would avalanche through the chaotic elements until it was conducted out of the system -- as the value of seven.

Chaotic elements are useful to this system because they can assume an infinite number of behaviors that can be used to represent different values or different systems such as logic gates. Because of this flexibility, altering the initial encoding and changing the connections between the chaotic elements allow a single generic system to perform a variety of computations using its inherent self organization. In conventional computing, systems are more specialized to perform certain operations.

"We are not really setting up rules in the same sense that digital computers are programmed," Ditto explained. "The system develops its own rules that we are simply manipulating. It is using pattern formation and self-organized criticality to organize toward an answer. We don't

micromanage the computing, but let the dynamics do the hard work of finding a pattern that performs the desired operation."
Just as the numbers can be encoded in a variety of ways, the answer also comes out in a variety of ways: a rate of change, amplitude, or a specific chaotic behavior.
"There are a surprisingly large number of ways that the system can perform the computations and give you the answer," he added. "By slightly changing the connectivity and the parameters of the chaotic system, we can have it multiply several different ways through the system's self organization."
Because this new system differs dramatically from existing digital computers, it is likely to have different strengths and weaknesses. "It might be better than digital computing for those activities that digital computing doesn't do very well -- such as pattern recognition or detecting the difference between two pieces of music," Ditto said.
He compared dynamics-based computation to DNA computing and quantum computing, both of which are new computing paradigms still in their early stages of development.
Ditto believes the new system would work particularly well in optical systems. He has done theoretical work applying dynamics-based computing to an ammonia laser system and hopes to see the system implemented experimentally.
"Potentially, we could stimulate a very fast system of coupled lasers to perform a highly complicated operation like very fast arithmetic operations, pattern detection and Fourier transforms" he said. "We have something that very naturally performs an operation in an optical system. This would provide an alternative to existing efforts, which try to make optical systems do operations more like transistors."
Beyond the systems they have tried, Ditto believes virtually any coupled dynamic system could be used to perform computation. "We hope that you can take any dynamical system, stimulate it in the correct way, and then get it to perform an operation for you," he said. "This would provide an alternative to engineering a system from the ground up."
Ditto acknowledges a number of engineering issues that may hamper development of a practical system based on this new computing paradigm. But he notes that in their early days, digital computers had to overcome a daunting set of obstacles to overtake earlier techniques.
Support for the work has come from the U.S. Office of Naval Research, and from Control Dynamics, Inc., a company partially owned by Ditto.
Note: This story has been adapted from a news release issued by Georgia Institute of Technology.

Introduction

Computer science is a soft science. That is, computer science, like mathematics is based of accepting the true of some basic theorems. The mathematical laws on which the conventional CS paradigm is based have served to guide CS during its history of incredible improvements of CPU speed and memory size, but like the relationship between the arrangements of the fingers on the hand and the layout of the QWERTY keyboard, the relationship between these laws and the real world of electrons, light beams and digital circuits is arbitrary. Absolute physical benchmarks used in the physical sciences such as Physics and Chemistry are ignored in the conventional CS paradigm. Instead, algorithm analysis depends on the nature of an algorithm in the limit. That is, even though computers are of a finite size and the speed of electricity is bounded, algorithm analysis is based on processing unlimited data sizes and efficiently utilizing increasingly insignificant CPU and memory costs while ignoring the impact of the complexity of algorithms on the increasingly dominate programming costs.

Unfortunately since the laws for the current paradigm are taught as absolute truths to every CS freshman, they are very difficult to challenge. Indeed, the perception is that they are infallible. The traditional relationship between mathematics and the sciences has been for mathematicians to develop abstract construct without regard to engineering or science and then, later, perhaps centuries later, discovered that they have a useful application. However, in the case of computer science, the abstract laws and theorems of mathematics have become the very basis of computer science with the consequence that any alternative paradigm can be proven to be inferior. The result is that new computer paradigms, architectures and languages, which could address many of today's issues such as the inefficiency of multitasking operating systems and hardware, have been overlooked, even suppressed. The wealth of alternative computer architectures, programming languages and alternative paradigms of earlier years, such as systolic arrays, data flow, data parallelism and associative computing, has been abandoned.

This point is illustrated by the Connection Machine fiasco of the late 80's and early 90's. The CM that was designed at MIT and promoted by DARPA was an attempt to develop alternative machine architecture. It failed because of inappropriate algorithm paradigms and programming techniques. That is, because of the mismatch between the accepted CS paradigm and the available programming languages and the reality of the machine, the programmers could not develop "cost effective" algorithms. The most obvious example of this mismatch is sorting versus searching. The need for sorting data for efficient retrieval is accepted without question in the conventional paradigm. To suggest that data can be efficiently retrieved without sorting is not only heresy, but can be proved to be wrong by any freshman computer scientist using conventional paradigm laws. Yet the CM, using data parallelism, could and did find any datum in memory in one step – O(1) time. This would be an existence proof in any other discipline, but it was not and still is not accepted by the CS community because the standard paradigm "proves" that it takes at least O(log n) time to find something (even conveniently ignoring the cost of sorting the data.) As a result, the massive searching approach to data selection and retrieval was not pursued. One of the results of the law that data need to be sorted in order to find them efficiently is the tree organizations. However, using a data parallel searching paradigm, there is no need for nested menu trees or any other type of sorted data. Data parallel searching, by eliminating the need for sorting, does not need linked lists and allows all data organizations – chronological, by recipient, by topical category – to be utilized simultaneously via associatively. That is, data parallelism and associative computing is one possible answer to simpler computing. This approach requires that you go back to the flat tables of yesteryear, but with gigabytes of memory and gigahertz of speed, flat tables are manageable, even preferable.

Moreover, the data parallel search based paradigm is well suited for high school level tabular data structures and natural language communication with your computer. Both features significantly reduce programming costs. But unfortunately, like the internal combustion engine, once a technology has progressed it a certain point, it is practically impossible to overcome it. However, as there is the possibility that the eventual depletion of our natural energy resources will cause economic changes in the design of car engines, the eventual depletion of "free" improvements in speed and memory may force a change in the conventional CS paradigm. Even today, super computers spend from 70 to 95% of their computing power in overcoming the limitations of the current paradigm so that only 5 to 30% of the actual speed improvements are obtained by the user. However, a change in paradigm is most likely to occur in the Laptop, PDA and cell phone environment not the super computer environment, because the lack of power is not the driving point, but the lack of physical space for keyboards, display screens, mice, etc. and the demand for easier modes of computing. Eventually, hopefully, computer scientists will come

to realize that, CS has been trapped by the von Neumann models success and that just as the laws of plain geometry work well over a small portion of the Earth's surface, but fail when applied to a much larger portion, so too, the laws of the current CS paradigm work well for the old world of "slow", few, expensive computers and cheap programmers, but are not well suited to the new world of fast, ubiquitous cheap computers and expensive programmers.

New Advances in Reconfigurable Computing and its Applications
J.UCS Special Issue by: (Miguel A. Vega-Rodríguez, Juan A. Gómez-Pulido, Juan M. Sánchez-Pérez Dept. Technologies of Computers and Communications, Univ. Extremadura Escuela Politécnica, Campus Universitario s/n. 10071 Cáceres, Spain

This is a survey of different papers about reconfigurable computing and its applications. These papers treat very different reconfigurable-computing applications: cryptography, computer vision, SOPC, microprocessor architecture, self-timed circuits, sensor systems, detection of ultrasonic emissions, FPGA compilation aspects (like data-dependent loops), and motion estimation. We can say that reconfigurable computing is becoming an increasingly important computing paradigm, being a good alternative for many real applications.

As the Reconfigurable Computing is becoming an increasingly important computing paradigm, more and more FPGA-based applications are appearing. FPGA devices are making it possible for thousands of computer engineers to have access to digital design technology in an easier way, obtaining a better performance with a similar flexibility to software. In addition, ASIC engineers are now "reconfiguring" themselves as FPGA engineers for economic reasons and adding to the growing legions of FPGA designers.
In conclusion, reconfiguration of circuitry at runtime to suit the application at hand has created a promising paradigm of computing that blurs traditional frontiers between software and hardware. At present, reconfigurable computing is a good alternative for many real applications in image and signal processing, multimedia, robotics, telecommunications, cryptography, networking and computation in general.
This Special Issue brings together high-quality state-of-the-art contributions about reconfigurable computing and its applications. Concretely, the special issue contains 7 papers that represent the diverse applications and designs being addressed today by the reconfigurable-computing research community. With authors from around the world, these articles bring us an international sampling of significant work.

New Computing Paradigm: A Chaos-Based System That Evolves Science and Engineering News (09/11/98)

San Diego, CA -- As John Toon reported for GIT, A revolutionary new computing technique that uses a network of chaotic elements to "evolve" its answers could provide an alternative to the digital computing systems widely used today. Described for the first time in the September 7 issue of Physical Review Letters this "dynamics-based computation" may be well suited for optical computing using ultra-fast chaotic lasers and computing with silicon/neural tissue hybrid circuitry.

The system has so far demonstrated an ability to handle a wide range of common operations, including addition and multiplication, as well as Boolean logic and more sophisticated operations such as finding the least common multiplier in a sequence of integers. Because it depends on

interaction among its coupled elements, the system is naturally parallel.

"We have shown that this can be done, but we've only seen the tip of the iceberg," said William L. Ditto, professor of physics at the Georgia Institute of Technology. "This is a glimpse of how we can make common dynamic systems work for us in a way that's more like how we think the brain does computation. It's an entirely new computing paradigm."

For many years, scientists have observed the rich variety of behavioral patterns created by chaotic systems, including those found in living organisms. Ditto and collaborator Sudeshna Sinha of the Institute of Mathematical Sciences in Madras, India, reasoned that these natural chaotic systems should have been eliminated through evolution unless they served a purpose.

Ditto and Sinha devised an experiment to see if a simple network of chaotic computer elements could be made to handle computations. They joined the chaotic elements into a lattice using an adaptive connecting mechanism that would open whenever an element exceeded a certain critical value. The mechanism was designed so that the researchers could set a wide range of critical values to vary the connection between elements.

The researchers then encoded values into the chaotic lattice using a variety of different techniques. In some cases, they chose patterns in the chaotic elements to represent numbers. In other cases, the numbers were represented by the amplitude of waves emitted by the chaotic elements or the frequency of "spiking" behavior.

After encoding the numbers, they stimulated the elements to begin interacting.

Elements containing values above the critical level triggered the connecting mechanism, allowing the excess value to "avalanche" into neighboring elements. That transfer then created additional avalanches in other connected elements. With additional stimulation, the domino effect continued until the imbalance was conducted out of the system -- as the answer to the mathematical problem.

"We have the elements interconnected so that they respond to their neighbors like the avalanching that occurs when you pile grains of sand onto a sandpile," Ditto explained. "You allow the elements to avalanche and the system to evolve chaotically, and then do the avalanching again until the system settles down to the right answer. It takes a couple of iterations for it to settle down."

In a simple example, values of three and four would be encoded into a system set with a critical value of one. The values would create an imbalance that would avalanche through the chaotic elements until it was conducted out of the system -- as the value of seven.

Chaotic elements are useful to this system because they can assume an infinite number of behaviors that can be used to represent different values or different systems such as logic gates. Because of this flexibility, altering the initial encoding and changing the connections between the chaotic elements allow a single generic system to perform a variety of computations using its inherent self organization. In conventional computing, systems are more specialized to perform certain operations.

"We are not really setting up rules in the same sense that digital computers are programmed,"

Ditto explained. "The system develops its own rules that we are simply manipulating. It is using pattern formation and self-organized criticality to organize toward an answer. We don't micromanage the computing, but let the dynamics do the hard work of finding a pattern that performs the desired operation."

Just as the numbers can be encoded in a variety of ways, the answer also comes out in a variety of ways: a rate of change, amplitude, or a specific chaotic behavior.

"There are a surprisingly large number of ways that the system can perform the computations and give you the answer," he added. "By slightly changing the connectivity and the parameters of the chaotic system, we can have it multiply several different ways through the system's self organization."

Because this new system differs dramatically from existing digital computers, it is likely to have different strengths and weaknesses. "It might be better than digital computing for those activities that digital computing doesn't do very well -- such as pattern recognition or detecting the difference between two pieces of music," Ditto said.

He compared dynamics-based computation to DNA computing and quantum computing, both of which are new computing paradigms still in their early stages of development.

Ditto believes the new system would work particularly well in optical systems. He has done theoretical work applying dynamics-based computing to an ammonia laser system and hopes to see the system implemented experimentally.

"Potentially, we could stimulate a very fast system of coupled lasers to perform a highly complicated operation like very fast arithmetic operations, pattern detection and Fourier transforms" he said. "We have something that very naturally performs an operation in an optical system. This would provide an alternative to existing efforts, which try to make optical systems do operations more like transistors."

Beyond the systems they have tried, Ditto believes virtually any coupled dynamic system could be used to perform computation. "We hope that you can take any dynamical system, stimulate it in the correct way, and then get it to perform an operation for you," he said. "This would provide an alternative to engineering a system from the ground up."

Ditto acknowledges a number of engineering issues that may hamper development of a practical system based on this new computing paradigm. But he notes that in their early days, digital computers had to overcome a daunting set of obstacles to overtake earlier techniques.

Support for the work has come from the U.S. Office of Naval Research, and from Control Dynamics, Inc., a company partially owned by Ditto.

Tuple Board: A New Distributed Computing Paradigm for Mobile Ad Hoc Networks
(Alan Kaminsky - Department of Computer Science Rochester Institute of Technology Rochester, NY, USA

The Tuple Board Paradigm

The tuple board distributed computing paradigm is derived from the tuple space paradigm. It first describes tuple space, and then introduces the tuple board and show how it differs from tuple space. It also describes how to design applications based on the tuple board. Section 4 describes how the tuple board is implemented.

Historical background

In 1985 Gelernter introduced the notion of tuple space and its associated distributed coordination language, Linda. Since then, tuple space has been implemented in many languages and platforms. Notable Java implementations include Sun Microsystems' JavaSpaces and IBM's TSpaces.

A tuple space based distributed application stores information in tuples. A tuple is a record of one or more fields, each field having a value of a certain data type. One process can write a tuple into tuple space. Another process can then take a tuple out of tuple space. To find a tuple to take, the taking process supplies a tuple used as a template. Each field of the template may be filled in with a specific value or be set to a "wildcard" value. The template is matched against all tuples in tuple space. A template and a tuple match if they have the same number of fields and the fields have the same data types and values; a wildcard matches any value. The taking process receives one matching tuple that was taken out of tuple space. If more than one tuple matches the template, one tuple is chosen arbitrarily to be taken; if no tuple matches the template, the taking process blocks until there is a match. A process can also read a tuple. Reading is the same as taking, except the reading process receives a copy of the matching tuple while the original tuple stays in tuple space.

Tuple space provides a distributed communication mechanism that decouples processes both "in space" and "in time." Processes need not reside on the same device to communicate. A process on one device can write a tuple and a process on another device can take or read the tuple; the processes thus have communicated through the intermediary of tuple space. Processes also need not be running at the same time to communicate. A process can write a tuple even if the process that will read or take the tuple is not running yet. The writing process can then go away, and the tuple will persist in tuple space until another process reads or takes it. Conversely, a process can read or take a tuple (and will block if necessary) even if the process that will write the tuple is not running yet.

Since mobile computing devices with wireless networking capabilities, such as laptop PCs, tablet PCs, and PDAs, are becoming prevalent, there is a need for distributed applications that run on groups of nearby devices. For example, people in a meeting would like to have their PDAs pool their individual calendars together to find a date and time everyone has free for the next meeting. With today's software this is typically impossible, since different people use different calendar software, different calendar server computers, and so on. Tuple space provides an alternative: Each PDA writes tuples with each person's open calendar slots, then all the PDAs read all the Tuples and find a common slot. Previous work, such as one world and Lime, has attempted to adapt tuple space to the mobile device environment. However, there are still difficulties. Since tuples are supposed to persist in tuple space even if the writing process goes away, tuple space is typically implemented on a separate, central server computer. However, central servers are unattractive for groups of mobile wireless devices, since the devices may not be in range of a central server. Although the devices are in range of each other, no device can act as a central server because devices can leave or turn off at any time. Without a central server, implementing tuple persistence is difficult and requires complicated software.

The notion of a tuple board was introduced as a modification of tuple space that is better suited for ad hoc networks of mobile wireless computing devices without central servers. A tuple board

is like a shared virtual bulletin board. One process can post a tuple on the tuple board. A process can withdraw a posted tuple; but a process can only withdraw the tuples the process itself has posted, not tuples any other process has posted. If a device leaves the network or turns off, all the tuples the device's processes had posted are implicitly withdrawn. Another process can read a tuple on the tuple board that matches a template, just as with tuple space. A process can set up an iterator over all tuples that match a template, and then repeatedly read a tuple from the iterator; a different tuple is returned each time. If all tuples matching the iterator's template have been read, the process reading the iterator blocks until a new matching tuple is posted. A process can set up a notifier for a template; the notifier will then inform the process whenever a matching tuple is posted or withdrawn.

The key difference between the tuple board and tuple space is that tuples do not persist on the tuple board if the posting process goes away. This greatly simplifies the tuple board implementation, since when a device turns off or leaves the network its posted tuples can simply disappear. Many useful distributed applications can be developed even without tuple persistence. In fact, the ability to detect when a device goes away – because a notifier reports that a tuple, which the device had previously posted, was withdrawn – is useful in its own right.

Applications Based on the Tuple Board

In this section we describe how distributed ad hoc collaborative applications are designed using the tuple board paradigm. Such applications are "collaborative" in that any number of nearby mobile computing devices can participate.

These applications are also "ad hoc" in that the devices do not need to be configured to know about each other ahead of time; instead, devices can come and go at any time, and the application runs on whichever devices happen to be nearby.

Information sharing applications of all kinds are easily implemented using the tuple board. Consider a digital photo sharing application. Each device with digital photos – PCs, PDAs, even cameras and cell phones – posts tuples for its pictures. Each tuple has, say, four fields: unique ID, thumbnail image, date taken, and description.

Any device can then obtain the other devices' pictures, as follows. To retrieve all the pictures, the device sets up a template where all four fields are wildcards; to retrieve only Disney World pictures, the template's description field is set to "Disney World" and the other fields are wildcards; and so on. The device uses an iterator to read all the tuples that match the template, assembles the tuples' thumbnail fields into an "album" of pictures, and displays the album on the device's own screen. If the user wants to see the full-size picture for some thumbnail, the user's device posts a tuple containing a full picture request for the picture's unique ID. Seeing that request posted, the device that has the picture with the given unique ID posts a tuple containing the full-size image.

The user's device reads that tuple and withdraws its request tuple, whereupon the other device withdraws its full-size image tuple. If a device joins the group, the new device merely starts posting tuples, and the other devices (responding to a report from a notifier) add the new device's thumbnails to their displays. If a device leaves the group, the remaining devices (again responding to a report from a notifier that tuples were withdrawn) remove the departed device's thumbnails from their displays. Thus, the application automatically adapts as devices arrive and depart, without needing to rely on a central server.

Other examples of ad hoc collaborative applications that can be designed in a similar way using a tuple board include file, music, and video sharing; groupware applications like shared whiteboard, shared document authoring, and the aforementioned shared calendar; and vendor information directories in shopping malls and avenues. Bondada built a conference information system demonstration using the tuple board collaborative applications based on the tuple board.

Mobile Computing Paradigm

Introduction

Mobile computing consistently fails to live up to expectations. Early adopters complain about the size and resolution of displays, awkward input devices and limited bandwidth. There is every reason to assume that the complaints will be exacerbated for mobile CSCW, since most collaborative computing is computationally intensive, often synchronous and visual. In addition, computer-mediated communications require minimal latency in order to be socially acceptable.

Many enthusiasts respond to this challenge by suggesting that new mobile technologies will be sufficiently powerful to meet the needs of CSCW. We assert, however, that expectations and requirements of the user community will proportionally increase by new advances in computing technology. The ante will, in a sense, be upped once more.

This paper provides an initial exploration of this problem, which represents a fascinating challenge given that today's mobile devices are immensely more powerful than the desktop computers of yesteryear, but at the same time they are a far cry from what users want today. We believe this challenge is fundamentally conceptual, rather than technical; today's mobile computing paradigm is simply not suited for handheld CSCW!

A research program has been launched, which aims to develop the conceptual foundations for mobile IT-use and CSCW more thoroughly. First, it is necessary to specify exactly how mobile IT-use differs from stationary computing. Next, a reference model is needed to guide fieldwork and provide a common vocabulary for designers.

Explaining the desktop 'bias' of mobile computing

Based on the premises offered above, the following argument can be given: Mobile IT is inferior to stationary computing in terms of performance and bandwidth. For example, bandwidth is limited, keyboards are awkward and there is no desk on top of which to put the devices when typing.

At the same time, mobile IT design is clearly stationary 'biased': de facto industrial standards have adopted the desktop metaphor and offer 'pocket' versions of familiar office applications. For example, there is Pocket Word (without styles) and Internet browsers (without support for Java and plug-ins), which, in addition, hardly display content but one line at a time.

It seems like this is a conceptual, rather than a technical *cul-de-sac*, since "users' needs", in these terms, will always exceed what mobile computing can offer. But as the following argument demonstrates, this can be conceived as a result of a naïve design paradigm.

Is advanced document management or internet-based multimedia publication purposeful operations for mobile workers? Are there mobile use contexts where a typewriter-metaphor based terminal with a connected keyboard and screen will be useful at all. Consider the example of electrical maintenance workers equipped with a desktop metaphor-based device. When operating the device as afforded by its design, i.e. sat down in from of the user in one arms length distance, the objects of work, which are switches and power cables in a roadside cabinet, cannot be reached. If the device, on the other hand, is put down on the only other flat surface, which is the top of the cabinet, then the display cannot be seen when squatting to reach the switches and cables.

In order to make the mobile computer support mobile work it has had to be turned into a desktop computer by connecting it to the stationary network and funnel its functionality through a hosting PC.

The desktop metaphor in mobile computing is, thus, a symptom of the confusion of operations and functionality in a stable use context. Only when the use context is constantly changing, like in mobile computing, it becomes apparent that this contributes to a problematic design paradigm. In order to open up an alternative design space (or at least a place to turn), an alternative conception of the relationship between computer-mediated services and continually changing modalities is needed. We are currently developing a model that responds to this challenge by explicating the nature of mobile IT-use.

Modeling mobile work

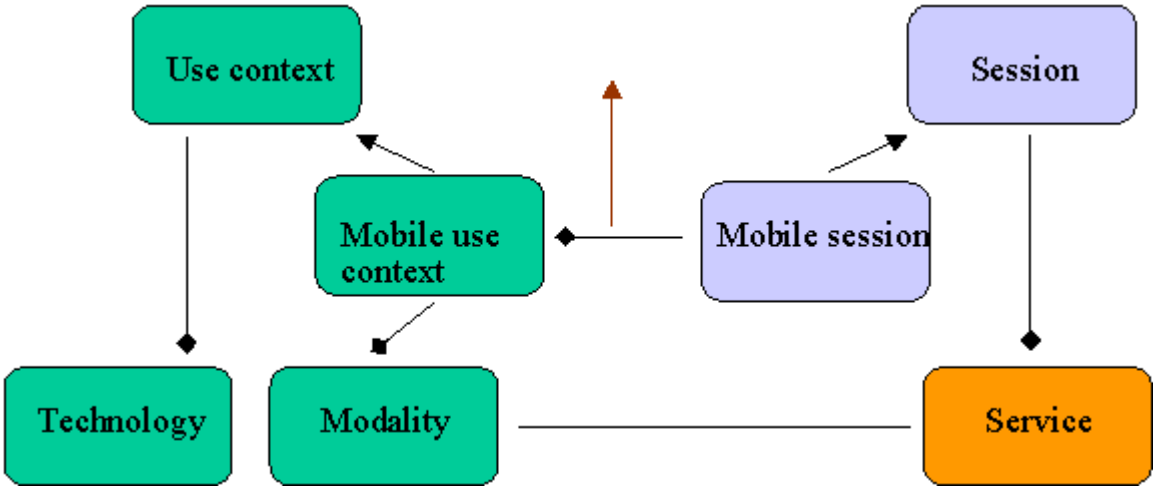
The reference model for mobile work builds on fieldwork and discussions. Its core concepts are:

Modality, which is a characterization of the physical relocation patterns of the mobile worker/mediating technology.

Technology, which can to varying degrees be adapted to mobile work, and
Service, which is the set of intended operations offered by the functionality of the mediating technology.

A *mobile session*, thus, simply consists of changing modalities and services.

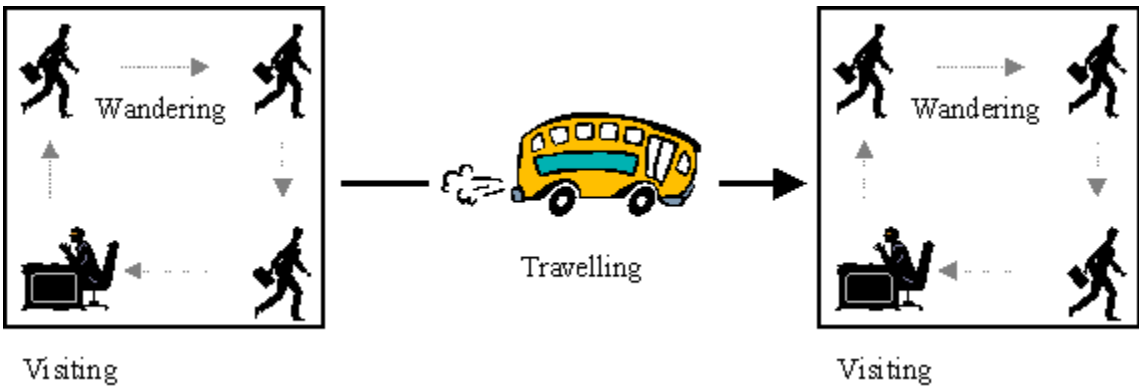
A *mobile setting* comprises at least one mobile session. The following figure summarizes the model in relaxed UML notation:



Modeling mobile work

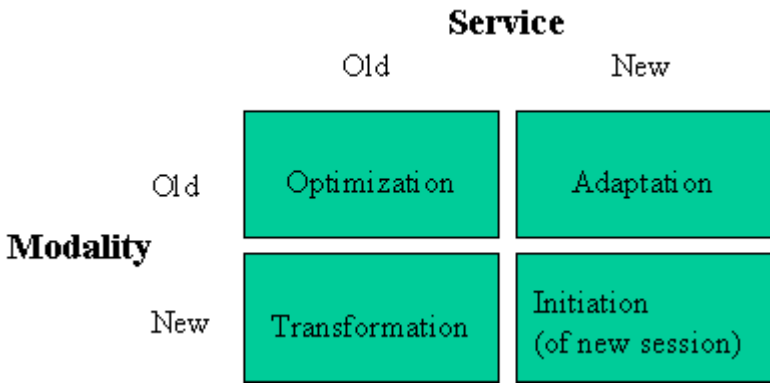
Three important *modalities* of mobile work are:
Visiting is working in different places for a significant period of time
Traveling is working while traveling in a vehicle, such as an airplane or a train
Wandering is working while being locally mobile

Certainly, these are only ideal types; nevertheless they represent a useful conceptual topology which distinguishes mobile work from stationary work and contributes to developing a new research agenda for handheld CSCW. The following figure illustrates the changing modalities of mobile work:



Modalities of mobile work

Services and modality constitute central components of a mobile session, which due to its mobility is far more contingent and dynamic than mobile work. Thus, it points in the direction of *new items for a mobile informatics research agenda* when these dimensions are combined, such as in the following figure:



Changing services and modalities

Implications for research
Building on the framework outlined above, the following items for a mobile computing research agenda can be outlined:

Initialization
Main objective: Develop models and systematic support for establishing sessions in a mobile setting.
Some selected research problems:
Browse, inspect and select services. One can assume that the availability of services from a mobile terminal will vary with geographical re-location.
Browse, inspect and select resources (such as documents, people, procedures, etc.), from within a service. It is integral to establishing sessions, especially involving other people in a synchronous setting, to be aware of who is available and what they are able/capable of contributing.
Negotiate access, quality and pricing. In a situation where mobile communications are more expensive then those offered by a stationary infrastructure, whilst at the same time furnishing limited performance and bandwidth, it becomes important to support the necessary trade-offs between cost in time, money and quality.

Programming sessions. In a mobile setting, user cannot be expected to be fully interactive, since mobility itself imposes demands on attention and mobile work is likely to be directed towards non-representable objects (otherwise, why go mobile?).

Transformation
Main objective: Develop models and systematic support for sustaining services between modalities in a mobile setting.
Introducing the notion of changing modalities enhances the mobile informatics research agenda with an interest in how people can change between modalities whilst remaining connected to current services.

Adaptation
Main objective: Develop models and systematic support for seamless use of new services within modalities in a mobile setting.

In this opposite of transformation, research interest is in how new services (for which the current modality may not be suited) can be adapted to function in a satisfactory manner.

- Negotiating access and performance profiles for users entering a new modality.
- Changing data from new services to make available programs interpret and refine such elements in a satisfactory manner.
- Adapting programs to deal with new types of data.
- Aggregation of service units into fully functioning services for the current modality.

Optimization
Main objective*:* Develop models and systematic support for improved performance of sessions in a mobile setting.

This item on the agenda is already well covered in the mobile computing area. There seems to be a tendency, however, of assuming that mobile technology will be almost as reliable as its fixed counterpart. We believe that more work is needed on how to complete business-critical tasks using mobile systems in case of unanticipated technological breakdowns.

We have selected one other area in which more work is definitely required: Simple input devices for mobile sessions.

New applications

The goal now is to combine empirical studies with the conceptual framework of the model to produce innovate, mobile-aware applications. By technical experimentation and empirical evaluation, such applications can inform re-design and improve the model.

Medical
Fostering the Biological Sciences in the School of Medicine (Rutter)
Now, with respect to the development of the biological sciences at UCSF, the Department of Biochemistry was a mini-school in itself because it covered all the most important scientific approaches in biology. We weren't just interested in molecular genetics. We also became a cell biology group because we were interested basically in the cell. There have been ebbs and flows. But it has been a department that has operated across the frontier of science. We always thought

we knew where the various fields were going and tried to get scientists who would help or lead the change. As the department flourished, of course we ran out of FTEs and space. So how to grow? Clearly, it was by colonization of the other departments.

Abstract: Single photon emission tomography allows the imaging of dynamic brain functioning. The use of cerebral activating procedures within the scan protocol enables investigation of the mechanisms involved in specific brain functions in health and disease. Activation studies involve the comparison of at least two data sets describing brain activity generated in conditions that differ for the specific function in question. When designing an activation study, decisions regarding methodology include: the nature of the activation regime, the tracer-ligand utilized, the SPET instrument and the manner of subsequent data analysis. These issues are discussed in this review, both theoretically and with reference to published studies. Means of activating particular cerebral structures and functions are reviewed, as are the limitations of the techniques with respect to temporal and spatial resolution and the potentially confounding nature of preconceived ideas regarding the mechanisms of brain function.

Simulators for biomolecular computing, (both in vitro and in silico), have come to play an important role in experimentation, analysis, and evaluation of the efficiency and scalability of DNA and biomolecule based computing. Simulation in silico of DNA computing is useful to support DNA-computing algorithm design and to reduce the cost and effort of lab experiments. Although many simulations have now been developed, there exists no standard for simulation software in this area. Reliability, performance benchmarks, user interfaces, and accessibility are arguably the most important criteria for development and wide spread use of simulation software for BMC. The requirements and evaluation of such software packages for DNA computing software are discussed, particularly questions about software development, appropriate user environments, standardization of benchmark data sets, and centrally available common repositories for software and/or data.

The Breast Cancer Working Group is a multidisciplinary team of researchers from the Baystate Medical Center and the University of Massachusetts Amherst. Working Group researchers represent a wide variety of scientific disciplines including surgery, pathology, nursing, epidemiology, statistics, molecular genetics, computer science, chemical engineering, and nanotechnology. The Working Group's laboratory-to-clinic approach expedites the translation of scientific discoveries into practical biomedical applications. In addition to university and hospital facilities, Working Group researchers have access to new, state-of-the-art laboratories at the Baystate-UMass Biomedical Research Institute in Springfield. Currently, the Group's collective effort is focused on understanding the nature of pre-cancerous breast lesions that can transition into malignant tumors. Dr. Richard Arenas, Chief of Surgical Oncology at Baystate, and Dr. Joe Jerry, UMass professor of Molecular Medicine, are two Working Group members whose collaboration with colleagues in the UMass Chemistry and Chemical Engineering departments is directed toward translating their medical and biological expertise into practical treatments and preventative therapies for breast cancer.

The Collaborative Approach
Collaboration fuels scientific innovation. In a 2001 report, the National Cancer Institute stated that, "multidisciplinary teams are needed to solve virtually all of the 'big' problems in cancer research." Productive collaboration among mathematicians, biologists, computer scientists, epidemiologists, imaging scientists, physicists, and clinicians is needed to effectively cure and

control cancer, the report said. Now, in western Massachusetts, just such a multidisciplinary research team exists. The Pioneer Valley's Breast Cancer Working Group, made up of M.D.s and Ph.D.s from the Baystate Medical Center in Springfield and the University of Massachusetts at Amherst, was formed to investigate the basic pathology of breast cancer and develop more effective interventions to treat and prevent the disease.

Collaboration across scientific disciplines has yielded many important achievements including the discovery of the structure of DNA. In his book on the process of scientific discovery, Francis Crick wrote, "In nature, hybrid species are usually sterile, but in science the reverse is often true. Hybrid subjects are often astonishingly fertile."

The Breast Cancer Working Group is a hybrid coalition of researchers, representing a wide variety of scientific disciplines ranging from pathology and surgical oncology to chemical engineering and nanotechnology that evolved out of a partnership between researchers at UMass Amherst and the Baystate Medical Center. They meet regularly to help steer the focus of the Group and to allow its direction to influence their own research programs and clinical practice.

At Risk for Cancer

Every year thousands of women are diagnosed with non-cancerous breast disease. Although, so-called benign breast disease is not cancerous, a certain form of the disease, called atypical hyperplasia, is associated with an increased risk of breast cancer. At least 10% of those women diagnosed with atypical breast lesions will develop cancer within one year.

At Baystate Medical Center, physicians frequently detect these atypical hyperplasias and, although they know that some of those patients will soon develop breast cancer, there is nothing to be done about it. Treating all women with atypical hyperplasia would be expensive and very invasive. However, inaction all but assures that 1 out of every 10 women diagnosed will be battling breast cancer within a year. This clinical dilemma, and the problem it poses for physicians and their patients, compelled the Breast Cancer Working Group to direct their focus toward investigating cancer associated with atypical, pre-malignant breast lesions.

Dr. Joe Jerry is a professor of Molecular Medicine at UMass Amherst and the designated leader of the Working Group. His lab uses animal models to study the molecular pathways that mediate susceptibility and resistance to breast cancer and to design targeted therapeutics to prevent it.

"Atypical doesn't sound good," Jerry says. "The physician has to tell the patient, 'go home and don't worry about it, but come back and have it checked out often, because you should worry about it.' That's pretty uncomfortable, and at some point people might die from worrying about the disease rather than from the disease itself."

Prevention is the Best Treatment

A primary goal of the Working Group is to harness the formidable scientific expertise of its members to achieve a better understanding of the basic cellular and molecular mechanisms that control how atypical hyperplasias transition into tumors. By accurately describing the pathology of tumorigenesis, new drugs can be designed and new methods developed to treat or even prevent breast cancer.

"Stopping the transition would be real prevention. It would be the ultimate prevention," says Dr. Richard Arenas, Chief of Surgical Oncology at Baystate Health Systems and an active researcher in the Breast Cancer Working Group. "If you know that, biologically, there is a link between atypia and the onset of actual breast cancer, presumably because you see it pathologically, then you've identified an actual pathway and there is a possibility of interrupting that pathway," he said.

Arenas has partnered with Jerry to explore how certain anti-cancer drugs affect tumor cells at a molecular level and with other Working Group members in the Chemical Engineering

department at UMass Amherst to develop more effective drug-delivery methods. They are interested in the effect that estrogen and prostaglandin-blocking drugs, called aromatase inhibitors and COX-2 inhibitors respectively, have on the activity a tumor-suppressor protein called p53. When working properly, p53 performs an indispensable function in the cellular reproduction process by preventing abnormal cell proliferation. Mutations in the p53 gene and non-mutational changes in p53 function may be associated with up to 45% of all breast cancers.

A Good Model

In depth experimental analysis in breast cancer is currently underway in Joe Jerry's laboratory at UMass Amherst. Jerry has developed a mouse model with a defective gene that consistently develops mammary tumors, which are very similar to those found in human breast tissue. Other animal models for breast cancer have previously been created, but none develop the true ductal hyperplasia that mimics human breast tumors.

"We're seeing a series of different structures in the carcinomas that are very much like humans," he says. This analogous system allows Jerry's research team to experiment with drugs that stimulate the surveillance function and boost its ability to prevent abnormal cell proliferation.

Almost nothing exciting about computing today has to do with data structures and algorithms

One of Alan's undergraduate degrees is in molecular biology. He can't understand it anymore despite having tried to review new developments every few years. That's not true in computer science. The basics are still mostly the same. If you go to most campuses, there is a single computer science department and the first course in computer science is almost indistinguishable from the first course in 1960. They're about data structures and algorithms despite the fact that almost nothing exciting about computing today has to do with data structures and algorithms.

The Internet is like the human body. It's replaced all of its atoms and bits at least twice since it started even though the Internet has never stopped working. Attacks on the 'Net aren't really attacks on the 'Net, they're attacks on machines on the 'Net. Very few software systems, maybe none, are built in ways that sustain operation in spite of being continually rebuilt and continually growing.

The future five years out is easy to predict because all of the forces acting on computer science are trying to keep it the same as it is now. Likely, the future will be more of what we have now.

Are computer science, software engineering, OOP, etc. oxymorons? Alan reminisce about Bob Barton, an early Utah professor. Bob said that "systems programmers are the high priests of a low cult" and "there are few things know about systems design, but the basic principle of recursive design is: make the parts of the same power as the whole. Bob Barton has a classic paper that contains seven of the most important things that people know about software today.

Another quote: "My job is to disabuse you of any firmly held notion you held when you came into this classroom." The best way to get people to think is to destroy their current thinking. Preconceived notions are largely reactions to what vendors are saying. Alan says that his course from Barton was the most valuable one he took in college because Barton garbage collected their minds.

"American's have no past and no future; they live in an extended present." This describes the state of computing. We live in the 80's extended into the 21st century. The only thing that's changed is the size. Windows XP has 70 million lines of code. It's impossible for Alan to believe that it has 70 million lines of content. Microsoft engineers don't dare prune it because they don't know what it all does. Cathedrals have 1 millionth the mass of pyramids. The difference was the arch. Architecture demands arches.

Computers are artifacts. in order to have a science of computing, we have to make them. This isn't unheard of. To have a science of bridge building, people had to start building them so they could be studied. He shows a clip of the Tacoma Narrows Bridge. After studying the failure, the bridge was rebuilt and hasn't come down. This reminds Alan of software systems. We're much better at building software systems than we are at predicting what they will do. There are no good models. If we were scientists, we'd be trying to build models.
"Science is not there to tell us about the Universe, but to tell us how to talk about the Universe." (Niels Bohr). Science helps us to be more reasonable about reasoning. It's set up so that we get better and better maps (abstractions) not perfect maps.
Alan uses John McCarthy and Lisp as an example of real science in computer science. He showed us that you can build a system that's also its own metasystem. Lisp is like Maxwell's equations. Many of the things that are wrong about Java is that it lacks a metasystem and that the metasystem that's been tacked onto it is missing key parts. To find the most interesting things about our field you have to go back 30 or 40 years.
Alan used McCarthy's method to design an object oriented system. He spent only a month implementing it because of the metasystem.
We build finite artifacts, but the degrees of freedom grow faster than we can reason about them. Thus, we're left with debugging.
We are creatures who live in a world of stories and we're always looking for simple stories o explain the world. He shows a clip, called Private Universe, of Harvard grads (at graduation) describing trying to explain the seasons. Almost everyone thought the seasons were caused by an elliptical orbit of the earth around the sun. It didn't matter whether or not the students were science majors or now. Interestingly, people know that the seasons are reversed between the northern and southern hemispheres. Their stories are inconsistent with what they know, yet they persist in believing them, even though they have the knowledge that contradicts their theory. When people react instantly, they're not thinking, they're doing a table lookup.
Engineering predates science by thousands and thousands of years because we don't have to understand things to engineer them. He uses one of my favorite quotes: an engineer is someone who can make for a dollar what any fool could make for two.
Making computing into a science means that we have to understand what to do about our beliefs. When we talk, we do nothing but tell stories that people will find interesting. There's danger in that because stories can create resonance without being scientific.

Distributed Systems.
The classical architecture of distributed systems relies on computer communication protocols. The main components of a distributed system are protocol entities that reside in different hosts and which exchange messages following a well defined communication protocol.
On one side, some distributed applications require an important deployment for testing and debugging; on the other side, users are not interested to use software if it has not been thoroughly tested and is not largely deployed:

Distributed operating systems use the exchange of special purpose messages for implementing non-local services. This imposes that a minimal set of protocols be explicitly wired into the kernel. These specialized protocols restrict the kernel's genericity and adaptability;

Distributed applications usually have two parts: one that is responsible for communication management i.e, implementing a protocol for moving data between the different hosts; the other part uses the communication facilities provided by the first one in order to implement the actual distributed algorithm.

Extending the above argument, we can argue that for the application programmer, developing a distributed application requires a different methodology from that of a centralized application. When developing a distributed application, the programmer must explicitly handle efficiently the data exchange in the system and it's processing whereas for a centralized application, only data processing is considered. We can thus argue that developing distributed applications that way is error prone; this is exacerbated by the difficulty to implement "correct" computer communication protocols. The ideal situation for the programmer would be to implement distributed applications the same way he/she implements centralized applications;
We present in this paper a new paradigm for performing computer communication, which we call "communication by messengers".

When two hosts communicate using this paradigm, they exchange programs called messengers". The messenger contains the appropriate code to instruct the recipient "what" it has to do next. A messenger contains both the data and the rules necessary to perform a given protocol.
This way of performing protocols does not require protocol entities be preconfigured in the communicating hosts.
It is the source host which completely determines which protocol is to be used for the data exchange; the destination host is not required to "know" the protocol being used. When used as the basis for distributed applications, the communication by messengers' paradigm brings more flexibility. It becomes possible for an application to explore the network and spread itself instead of having to rely on pre-configuration and installation of application specific software.
Another view of a messenger is that of a thread created in a remote host on behalf of the source host. From a programmer's viewpoint, sending a messenger to a remote host is similar to the creation of a local thread. This offers a uniform way for programming both centralized and distributed applications. Moreover, the communication by messengers paradigm can be used as a simple way to extend the remote procedure call paradigm.

The Messenger Paradigm
The messenger paradigm is born from a new way of thinking computer communications. Instead of the classical sender/receiver model based on protocol entities that exchange and *interpret* protocol specific messages, Tschudin proposed in a communication model based on the exchange of protocol unspecific programs, called the *messengers*. Hosts receiving messengers will then *execute* the messengers' code instead of interpreting them as messages.

The Messenger Platform
All hosts involved in a messenger based communication share a common representation of the messengers and provide a local execution environment, *the messenger platform*. A messenger is executed sequentially, but several messengers may execute in parallel inside the platform.

Platforms
```
arriving
messengers
~
...
~
x
y
123
```

'abc'
process
queues
channels
dictionary
messenger execution platform
msgr threads:
Messenger platform are connected through unreliable channels through which messengers are sent as simple messages or data packets.
A set of *channels* enables a messenger platform to exchange messengers with other platforms. The channels provide an unreliable datagram service which enables messengers either to reach entirely and correctly the neighboring platform or to be discarded or lost;

Integrally arriving messengers are turned into independent messenger processes or threads of execution, also called messengers for simplicity. The platform does not interfere with their execution except for programming errors or unavailable resources. Also do other messengers have no means to stop or kill another messenger process without its consent.
The executing threads are able to share common data inside a given platform by the means of a dictionary which contains pairs of keys and data values; Process queues are the low-level mechanism offered by a platform in order to allow messengers to implement basic concurrency control functionality such as synchronization of threads, mutual exclusion, etc. Process queues are FIFO queues of messenger processes. All threads of the queue are blocked except the thread at the head of the queue.
Platforms share
(1) a common messenger programming language, used to express messenger behavior; and
(2) a common external representation of the messenger, used for the physical exchange messengers.

Primitives of a Messenger Programming Language
Messengers are exchanged between two platforms as simple messages, using a common external representation. Arriving messengers are turned into threads of execution by the corresponding platforms. A messenger is then a sequence of instructions expressed in a *messenger programming language* common to all platforms. Beside general purpose instructions (arithmetic operations, logical operations, etc), a messenger programming language must offer a set of specialized primitives. Here we present an example of such a set of primitives: Process queues, channels as well as global variables in the dictionary of a given platform are accessed by a key. Keys are either generated by the platform (e.g., name of a channel) or can be constructed/chosen by the messenger processes using the `key ()` primitive. Global variables in the dictionary are accessed through their key using the `get(k:key)` and `set(k:key, v:value)` primitives; A messenger is sent through a channel to another platform by the `submit(k:key, m:msgr descr)` primitive. It is also possible to create a new local messenger process by submitting a messenger to a special "loop-back" channel – the new process is completely independent of its launching process; A messenger is able to replace its behavior by another behavior with the `chain(m:msgr descr)` primitive; Process queues play a central role in the synchronization of messenger processes. Primitives for handling process queues are: `enter(q:key)` which enables a messenger process to enter a queue, such a process will then be blocked until it reaches the head of the queue; `leave` which enables the messenger process at the head of the queue to remove itself from the queue; `stop(q:key)`

which stops the queue so that when the messenger process at the head of the queue leaves the queue, the next messenger process coming at the head of the queue will remain blocked; and `Start(q:key)` which resumes a queue previously stopped by the `stop` primitive. A referenced queue that does not yet exist is automatically created by the platform.
By the use of the `submit` and `chain` primitives, messengers are able to act as mobile entities that can propel themselves across the network to search for a given information or to perform a given task in a remote place. Alternatively, a messenger can continue its task in the local platform while the messengers it sent out will execute remotely before returning with the result of some sub-task. The possibility for a messenger process to replace its behavior with the `chain` primitive can be compared with the change of behavior of actors in the actor model of Agha. The difference resides in the fact that the behavior change is a fundamental part of the actor model: actors are seen as abstract machines dedicated to process one incoming communication. Actors always change explicitly or not their behavior before processing the next communication. On the other side, messengers are mobile entities that perform given tasks, which are not necessarily reactions to incoming communications.

Conclusion
The messenger paradigm is a communication model relying on instruction-passing instead of message-passing. Messengers are mobile threads of execution that can travel across a network. The basic requirements needed to achieve the messenger paradigm are:

(1) All hosts involved in the communication must be provided with the messenger platform,
(2) Al platforms are able to interpret the same messenger
language; and
(3) Unreliable messenger “channels” link the different platforms.

Distributed systems are built on top of a communication mechanism. The proposed communication by messenger paradigm is a low-level communication mechanism, which can be used at different levels of abstraction. We have shown that all kind of protocols can be implemented using this new paradigm: functionality of classical protocols, based on entities exchanging PDUs, can be realized without entities and without PDU exchange, using messengers. Distributed operating systems, which depend on communication protocols, can then be implemented using messengers.

Going further, the classical view of data traveling the network to be exchanged is no longer available according to the messenger paradigm where it is the code which travels the network to search for the data. Following these ideas, the well-known models of server/client and sender/receiver are no longer available, since a messenger based client will ask no request to a server, but goes itself to take the information and a messenger based server will not know which clients are taking information. Implementing distributed applications with messengers implies a new way of thinking: distributed applications will no more be seen as composed of static entities exchanging data, but as a set of mobile threads which can propel themselves across the network to look for resources.

Research pushes quantum spin technology toward real-world applications
Researchers have provided "proof of concept that quantum spin information can be locally manipulated using high-speed electrical circuits," according to an abstract of their paper being published on the "Science Express" website. The findings are significant because they

demonstrate a solid-state quantum logic gate (i.e., control mechanism) that works with gating technologies in today's electronics, today's computers. This research also moves esoteric spin-based technologies of spintronics and quantum computing from the futuristic closer to within reach of present-day possibilities. From the Barbara: Electrical Control of Electron Spin Steers Spin-Based Technologies Toward Real World

Santa Barbara, Calif. -- Researchers at the University of California at Santa Barbara (UCSB) and at the University of Pittsburgh have provided "proof of concept that quantum spin information can be locally manipulated using high-speed electrical circuits," according to the abstract of their paper being published expeditiously at 2:00 p.m. Jan. 23 on the "Science Express" website, Science Magazine's rapid portal for publication of significant research findings to appear subsequently in print in Science.

The findings are significant because they demonstrate a solid-state quantum logic gate (i.e., control mechanism) that works with gating technologies in today's electronics, today's computers. This research also moves esoteric spin-based technologies of spintronics and quantum computing from the futuristic closer to within reach of present-day possibilities.

The research was carried out in a joint venture between David Awschalom, Professor of Physics and Electrical and Computer Engineering at UCSB and Director of the Center for Spintronics and Quantum Computation (part of the California NanoSystems Institute [CNSI]), and Jeremy Levy, Associate Professor of Physics at the University of Pittsburgh and Director of the Center for Oxide-Semiconductor Materials for Quantum Computation.

A year ago at a program on Quantum Information held at the Kavli Institute for Theoretical Physics at UCSB, the two physicists fell into a conversation that led them to wonder how electron spins in semiconductors could be manipulated in all three dimensions.

The problem was an old one. So-called "spin resonance" techniques, used extensively for magnetic resonance imaging (MRI) and chemical identification, manipulate electron and nuclear spins in three dimensions using rapidly alternating magnetic fields. However, such magnetic fields are difficult to generate and control on a local scale. By contrast, local control over electric fields forms the basis of all of electronics, from CPUs to cell phones. The challenge was how to figure out a way to control electron spins using electric fields.

Awschalom and Levy realized that if a host for electrons could be designed for which the axis of spin rotation changed with an applied electric field, the spin direction could itself be controlled. That is, they could turn electric fields into effective magnetic fields.

The two researchers realized that semiconductor sandwiches made of aluminum gallium arsenide and gallium arsenide could provide just this sort of control. The material was built atomic layer by atomic layer in the Molecular Beam Epitaxy (MBE) laboratory of UCSB Materials Professor Art Gossard by Robert Myers and Dan Driscoll, graduate students jointly of Gossard and Awschalom. Myers and Driscoll guided the deposition of the material such that the parabolic quantum wells (PQWs) "are grown by varying the aluminum concentration x, ranging from 7% at the center to 40% at the barrier, to shape the conduction band into a parabolic potential," according to the paper "Gigahertz Electron Spin Manipulation Using Voltage Controlled g-Tensor Modulation."

The "Science Express" results provide just such a demonstration. Awschalom's physics graduate student, Yuichiro Kato, conducted the low temperature experiments at Santa Barbara. He used microfabrication techniques to construct the semiconductor devices and operate them, and experimental techniques developed by Awschalom and Levy to show that the researchers have indeed accomplished what they set out to accomplish.

Harkening back to that conversation of conceptual breakthrough between Levy and him, Awschalom said, "We realized that if we write down the equations of motion of the electron that are normally operated on by magnetic fields, we can replace the magnetic fields with electric

fields. Then we thought if that is the case, we could use miniature gates to operate on spins instead of magnetic fields for scalable architecture for computing."

Key to understanding the far-reaching implications of this proof of concept is the use of electrical fields, instead of magnetic fields, to control the electrons. Today's semiconductor, charge-based technologies (including computers) operate by control of electrical fields. Most researchers approaching the spin-based paradigm for spintronics and quantum computing technologies have assumed that the behavior of spins must be controlled by magnetic fields. The prospect of controlling 100 million magnets each independently on the equivalent of a chip has boggled the imagination of researchers. By contrast, controlling 100 million devices with electrical gates is what we already do in computers sitting on a multitude of desks throughout the world.

The parabolic quantum wells are grown with varying concentrations of aluminum gallium arsenide, sandwiched between gallium arsenide, flanked by metal plates, grown by deposition on the gallium arsenide. Think of a sandwich with Swiss cheese in the middle (aluminum gallium arsenide quantum wells) flanked by meat (gallium arsenide) in turn flanked by bread (the metal plates). The plates are the gates with one lead for the application of electrical current that in turn creates the electrical fields, which enable manipulation of the electrons through the material whose varying concentrations of aluminum govern the rotational speed of the electrons and the direction of their axes.

The result is "electron spin resonance (ESR) on a chip." This engineered nanostructure allows use of very small voltages in traditional gates to operate on electron spin in all three directions in which the axis can point without requiring fast alternating magnetic fields. "The experiments show that it is possible to build a very scalable array of quantum gates using semiconductors in a relatively straightforward manner," said Awschalom.

The experiments were conducted at low temperature and at a 50-micron scale, but designing them to operate at higher temperatures and smaller scales will likely not be difficult, according to Levy.

"This work describes and demonstrates how to replace magnetic with electrical fields for the control of spin information in semiconductors," said Levy. He described the findings as an "enabling technology" for spintronics and a "feasibility demonstration" for quantum information processing. For the latter, the size of the gates must be reduced so that the spin of a single electron can be precisely controlled. Such a feat would enable electron spins to be used as quantum bits or "qubits" for a quantum computer. "Ultimately," the researchers write at the end of their paper, "these electrical gates may be scaled down for operation on single spins and in quantum dots to form qubits."

In contrast to bits, the "1"s and "0"s of present-day computers, qubits can be in both the "1 state" and "0 state" at the same time, enabling a much richer and more powerful paradigm for computation. The orientation of an electron spin can be used to store one qubit of information. Quantum gates are then needed to reorient the electron spins and perform "quantum information processing."

And then there is the issue of spin-spin interactions, the next research hurdle to be overcome to enable another milestone in this line of progression toward full feasibility demonstration for quantum information processing. As the researchers write, "Control over single spin operations is sufficient for universal quantum gating, provided there is a 'backbone' of spin-spin interactions."

But, said Awschalom. "For the industrial sector looking at quantum information processing and asking whether there's a future, there is a trillion dollars of semiconductor technology for leveraging, and electrical control of spin makes the leveraging far more feasible and cost-effective than control with magnets. The only thing new here is the concept; the technology is

today's technology. There's nothing so special that would keep anybody anywhere from doing this experiment. Our hope is that people will do this much better in the next generation."
And, adds Awschalom the physicist in contradistinction to Awschalom the technologist, "This work points towards a way for potentially manipulating quantum states and entangling them to test some of the fundamental aspects of quantum mechanics with existing technology."
Funding for the work is being provided by the Defense Advanced Research Projects Agency (DARPA).

Proceedings of the 28th Annual Hawaii International Conference on System Sciences - 1995
DCCA : A Versatile Paradigm for the Description and Development f Concurrent Cmmunicating Systems (Sudhir Aggarwal * Sandeep Mitra Sanjay S. Jagdale
Department of Department of Department of Computer Science Computer Science Systems and Industrial Engg. SUNY-Binghamton SUNY-Brockport University of Arizona Binghamton, NY)

Abstract
Few methodologies exist that facilitate the formal specification and prototyping of distributed systems. In this paper, we describe certain features of the Dynamic Coordinated Concurrent Activities (DCCA) model. Any DCCA specification consists of a set of largely independent processes, each of which, however, needs to coordinate with several of its Upeers" in the course of its execution. Several diverse real world applications subscribe to such a paradigm - for example, a distributed control system for an automated factory, and a multiprocessor cache coherence system. DCCA is versatile enough to facilitate the specification of the protocols in both these systems on the same basis. Rapid prototyping and validation is also possible for DCCA models, as we describe in this paper. DCCA, and the attendant toolset could be of great use to a software engineer. 1 Introduction Advances in techniques for software development for distributed systems have not kept pace with those in hardware capabilities. Client-server models are still the norm for such systems, but these are inherently one-to-one mechanisms developed at the level of source code (e.g., in C). The networking code (e.g., socket establishment) is also explicitly written by the user. In recent years, therefore, there has been a trend towards developing alternative paradigms, and towards describing the behavior of systems at higher, more abstract levels. Formal Description Techniques (FDTs) are an example of this approach. *This research was 92-101-51 supported in part by NSF Grant CCRThe behavior of communicating systems can be systematically described using the mathematical basis of FDTs. This basis is usually state transition-oriented (as in Petri Nets (PNs) [l], ESTELLE [2]) or interaction sequence-oriented (e.g., LOTOS). Though ESTELLE and LOTOS have been standardized, few techniques exist that facilitate the description of systems that are inherently distributed and/or for which the target implementation environment is of a distributed nature. In the manufacturing domain, for example, PNs have been extensively used to model the system protocols ([4, 61). These PN models are, however, usually monolithic, and extremely complex, even for relatively small systems (for an example, refer [4]). Moreover, prototyping for a distributed environment is hard using PN descriptions. Models with a different basis are, therefore, needed. We propose DCCA as a state transition-oriented model that recognizes the distributed ness inherent in systems such as manufacturing controls and cache coherence. We discuss the features and versatility of DCCA here. Isis ([7]) is a similar paradigm.
1.1 A Control System for the Manufacturing Environment Flexible Manufacturing Systems (FMSs), which reflect the state-of-the-art in automated factories, are usually comprised of a

group of computerized numerically controlled (CNC) machine tools (e.g., lathes/mills/drills) interconnected by means of transportation agents (e.g., robots/conveyors). Flexible Manufacturing Cells (FMCs) refer to smaller groupings of machine tools. For example, a single FMC may consist of three machine tools handled by a single robot (that loads/unloads parts undergoing machining from these machines). Several such FMCs may be connected in a "star" configuration under the control of another single robot.

Report on the DNA/Biomolecular Computing Workshop

Arthur L. Delcher
Department of Computer Science
Lee Hood
Department of Molecular Biotechnology
University of Washington
Richard M. Karp
Department of Computer Science and Engineering
University of Washington

June 6-7, 1996

Abstract

This report is a summary of the DNA/Biomolecular Computing Workshop held on June 6-7, 1996 in Arlington, VA. The purpose of the workshop was to bring together researchers from both computer science and biotechnology to assess the current state of biomolecular computing, and to identify and explore the most promising approaches and most critical problems of this technology. Presentations by both computer scientists and biologists described the current state and future directions of biomolecular computing research and of competing computing technologies. The consensus of the participants was that although there is no clear road map toward achieving an efficient biomolecular computing device, there are many research directions that should be pursued that could lead to such a device.

Executive Summary

The current explosion of interest in molecular computing was sparked by Adleman's 1994 *Science* paper in which he showed how to use DNA to encode and solve a "toy" 7-city traveling salesperson problem. Although 7 cities is such a small problem that it can be solved at first glance by anyone, the method used could in principle be extended to much larger instances of the problem. This problem is particularly interesting since it is a member of an important class of problems known as *NP-complete* for which there are no known efficient algorithms on conventional computers. The reasons for optimism about this approach were the minute amounts of DNA and energy used and the large number of molecules that could be operated on concurrently. But questions remained about the scalability of the approach and the cost of materials for it.

Since then, much work has been done, both in assessing the theoretical computational issues of molecular computing as well as studying and improving the practical aspects of the biochemical systems. Molecular computing has been shown to be universal, meaning that it is theoretically capable of performing any computation that a conventional computer can. Several combinations of computational primitives, supported by a variety of biochemical reactions, have been proposed, and some have been tested in the laboratory.

The DNA/Biomolecular Computing Workshop was held on June 6-7, 1996 in Arlington, VA. It was designed to bring together researchers from both computer science and biotechnology to assess the current state of biomolecular computing, and to identify and explore the most promising approaches and most critical problems of this technology. The format of the workshop

consisted of informal presentations alternating with lively open discussions. On the last afternoon of the workshop, separate working groups of computer scientists and of biologists met and prepared lists of research areas to be pursued. In the final session these lists were presented and discussed.

The consensus view was that it is unlikely that a general-purpose biomolecular computer capable of outperforming a conventional electronic computer will be produced in the near future. This field is in its infancy, however, and more basic research needs to be done to assess its capabilities. There are many possibilities for molecular computing architectures that should be explored. Many of these, in addition to being computationally interesting, will almost certainly generate useful technologies for non-computational applications. Specific research areas that should be pursued include better characterization of the chemical reactions involved as computational primitives, improved or alternative techniques for performing input/output, and new computational models based on molecular components other than DNA in solution.

Introduction

A series of discoveries over the past fifty years have illuminated the extraordinary capabilities of living cells to store and process information. We have learned that genes encoded digitally as nucleotide sequences serve as a kind of instruction manual for the chemical processes within the cell and constitute the hereditary information that is passed from parents to their offspring. Information storage and processing within the cell is more efficient by many orders of magnitude than electronic digital computation, with respect to both information density and energy consumption.

The field of recombinant DNA technology has developed, based on procedures for synthesizing, cutting, splicing, copying, replicating and reading DNA molecules, and for separating and classifying them according to their size or content. These processes are fairly slow but highly parallel, operating on as many as 10^(16) molecules at a time. We have the ability to create designer genomes by splicing together selected fragments from different organisms, and cloning them by exploiting the self-replicating ability of bacterial cells. Recombinant DNA technology has also led to the creation of DNA hybridization arrays that can be used to analyze genomes and detect mutations associated with disease.

Recombinant DNA technology is an example in which the information processing capabilities of biological molecules are exploited for technological purposes. Another example is combinatorial chemistry, in which proteins with a specific activity, such as binding to a particular antigen, are synthesized by an iterative process which operates on a large pool of candidate proteins simultaneously *in vitro*, alternately selecting those proteins that are fittest with respect to the desired activity and then mutating them randomly to obtain a new pool of candidate proteins.

In the examples of recombinant DNA technology and combinatorial chemistry, processes inherent to living cells are used to analyze or modify biological molecules, and to select those with desirable properties. The field of biomolecular computing goes further by using these processes to perform information processing that has no intrinsic connection with the processing of biological molecules - in other words, biological molecules are used as a medium for general-purpose digital computation.

Biomolecular computing leapt into prominence in late 1994 through the work of Len Adleman, who performed a laboratory experiment in which a collection of DNA molecules was constructed representing the possible solutions to a toy combinatorial problem, and recombinant DNA techniques were used to sift through these molecules to select the correct solution. Subsequent work has introduced many refinements, but has for the most part stuck to Adleman's original generate-and-test approach, in which a large collection of candidate solutions is

generated, and tests are then performed in order to discard the incorrect solutions, until only the correct ones remain.
The original enthusiasm, based on the extraordinary energy efficiency and compactness of information storage afforded by the DNA medium, together with the extraordinary degree of parallelism of recombinant DNA procedures, is tempered by a number of sobering concerns. Among these are the following:

- It seems infeasible to operate on more than about l0^(16) DNA molecules at once. Thus the generate-and-test approach is limited to problems with at most l0^(16) candidate solutions. Typical problems of this size are easily dealt with by conventional methods of computation, leaving little advantage for DNA computing.

- DNA processing is slow, so that long sequences of steps must be avoided.

- DNA processing is error-prone. Error-correction techniques are available, but they multiply the numbers of steps required and the numbers of molecules processed in each step.

- In a large-scale DNA computation the cost of materials such as oligos, ligases, polymerases, restriction enzymes and hybridization enzymes may be prohibitive.

Because of these considerations DNA computation as we currently understand it may have a very limited range of application. Certainly no "killer application" has been identified yet. Nevertheless, biomolecular computing is still in its infancy. Our understanding of it, currently limited to a particular paradigm of DNA computing, will surely broaden as we gain a better understanding of information storage and processing in living cells, and the challenge of achieving cost-effective biomolecular computing will surely serve as a forcing function for advances in both biotechnology and computer science. Despite the short duration of our workshop, a number of promising, albeit speculative, directions for investigation were identified. We conclude the Introduction by mentioning some of them.
Information Processing Mechanisms in the Cell There is a great deal of science to be done in elucidating the mechanisms by which living cells store and process information and developing new biochemical tools and techniques based on these mechanisms. These new mechanisms will inevitably suggest new modes of biomolecular computing. Examples of such possibilities include:

- using artificial analogs of DNA and protein as computing media;
- using DNA to form self-assembling two- and three-dimensional structures analogous to cellular automata;
- exploiting the information inherent in the secondary and tertiary structure of DNA and proteins;
- using living cells as components in computing systems.

We should investigate Novel Computer Architectures, such as analog biomolecular computers and hybrid computers containing general-purpose electronic processors interacting with special-purpose biomolecular processors, as well as Special Applications such as sequence assembly or drug design, in which the information to be operated upon is inherently chemical or biochemical, so that costly input/output conversions between electronic and biochemical media can be dispensed with.

Departures from the Generate-and-Test Paradigm Combinatorial problems can be attacked by parallel local search algorithms, which maintain a large but not exhaustive pool of potential solutions, and gradually improve the quality of the pool by mechanisms of selection, crossover and replication analogous to those used in combinatorial drug design. DNA implementations of such procedures would not require the generation of all possible solutions, and could therefore be applied to larger problem instances.
Spin-offs The development of DNA computing will require the refinement and careful characterization of basic procedures for separating, ligating, cutting and amplifying DNA. The improved implementation and understanding of these procedures should find important applications in the biotech industry.

History and Current State of Biomolecular Computing
After opening remarks, the workshop began with a presentation via teleconference by Leonard Adleman, University of Southern California, on the history and current state of research in the area of DNA and biomolecular computing.
The historical origins of molecular computing can be found in the work of researchers from three academic arenas. Logicians Godel, Church, Kleene and Turing showed that universal computation could be accomplished using only memory and simple calculation steps. Biologists Watson and Crick discovered how genetic information is encoded digitally in DNA and how enzymes could manipulate this information. Finally physicist Richard Feynman argued that there were no inherent physical laws to prevent building extremely small computers.
The current explosion of interest in molecular computing was sparked by Adleman's 1994 *Science* paper in which he showed how to use DNA to encode and solve a "toy" 7-city traveling salesperson problem. Although 7 cities is such a small problem that it can be solved at first glance by anyone, the method used could in principle be extended to much larger instances of the problem.
The traveling salesperson problem consists of a collection of cities with a designated set of one-way connections from one city to another. The goal is to determine whether there exists a sequence of connections that visits every city without going to the same city twice. What makes this problem particularly interesting is that it is a member of an important class of problems known as *NP-complete* for which there are no known efficient algorithms on conventional computers.
Adleman solved the problem by designing a separate strand of DNA for each possible connection from one city to another. Moreover, the strands were created so that they would bind together exactly when the destination of one connection matched the source of another. The actual DNA was then fabricated and mixed together so that all possible bindings would occur. From the result, molecules that represented solutions to the problem could be isolated by extracting those molecules that had the correct length and which contained the code segments for all the cities.
There were several reasons to be optimistic about the potential of DNA computing based on the characteristics of Adleman's experiment. The volume of DNA used was very small, only 100 microliters. The speed of computation, specifically the rate at which molecules combined, was very fast, approximately 10^(14) operations/second (a pentium chip performs about l0^(8) operations/second while a parallel supercomputer may perform as many as 10^(12) operations/second). The energy used for the computation was extremely small, only 2 x 10^(19) operations/joule (within a factor of 17 of the theoretical optimum at room temperature). Finally the density of memory storage was very large, about 1 bit/cubic nanometer (about l0^(12) times denser than conventional videotape).

On the other hand, there were some clear difficulties to the approach. As implemented, Adleman's approach would require oceans of DNA to scale up to large enough problems to be of interest. The error rates in the molecular processes in the computation were high, especially in comparison to those of conventional computers. The cost of the materials in the experiment was high (some of the enzymes cost 10^(5) as much as gold). Finally, there was no clear computational problem that people wanted to solve for which the DNA approach appeared superior to conventional computers. In other words, there was no "killer application".

In the ensuing year further work has been done both in assessing the theoretical computational issues of molecular computing as well as studying and improving the practical aspects of the biomolecular systems themselves. In the theoretical domain it has been shown that molecular computing is universal, *i.e.*, it is capable of performing any computation that any other kind of computer can (but, of course, not necessarily as efficiently). In the biology domain, work has progressed in characterizing and improving the biochemical operations that can be performed, and in designing new architectures for biomolecular computers including different subsets of allowed operations and different methods of holding the DNA being used (attaching it to glass for example, rather than keeping it in solution). Work also has progressed in designing DNA sequences with desirable properties and in controlling the error rates inherent in the biochemical reactions.

One promising alternative molecular computing architecture is the *stickers* mode, currently being explored by Adleman. In this model, long memory strands of DNA are used together with shorter strands--stickers--which can anneal (attach) to a unique site on a long strand. An attached sticker represents a 1-bit, while the absence of a sticker represents a 0-bit. A test tube contains a collection of identical DNA memory strands, but with different stickers annealed.

There are three basic operations supported by the stickers model. The simplest is a merge operation in which the contents of two tubes are combined. The second is a separation operation based on the presence or absence of a sticker at a particular location. Two tubes are produced--one containing all complexes with the particular sticker attached, the other containing the remaining complexes. The third operation is a sticker-attach operation that anneals a specific sticker to all memory strands in a test tube.

The stickers model can be used to design a system to break Data Encryption Standard (DES) encryption. Specifically, given a message and its encrypted version, the system will compute which 56-bit key was used for the encryption. The proposed system would use a rack of test tubes containing a total of 1.4 grams of DNA, manipulated by 32 robots under microprocessor control. It would require separation error rates of less than 1 bad molecule for every 10,000 good ones, and would use no expensive enzymes. If stickers operations could be performed at the rate of 1 per hour, then DES could be broken in 7 months; at the rate of 1 operation per second, DES could be broken in 1.5 hours.

There are several reasons to be optimistic about DNA computing in the future. It realizes massive parallelism. It works with very low energy consumption. It can store huge amounts of memory in a very small volume. It has made rapid advances in a very brief time frame. Systems are now being proposed that do not use vast amounts of DNA, or require phenomenal error rates, or use expensive reagents. Finally, there are many potential architectures available, combining different biochemical technologies and computational models.

Reasons for pessimism still remain, however. The DNA reactions are slow, taking as much as an hour each. No practical application that fits this technology has been identified. Achieving the desired error rates in the chemical reactions is a major technical challenge. Finally, there is very strong competition from conventional electronic computers, in which phenomenal intellectual and financial investments have been made for more than 40 years.

Technologies

Competing Technologies

To be of practical significance, biomolecular computing systems will need to outperform other kinds of computing systems.

Conventional Electronic Computers Tilak Agerwala, IBM, gave a presentation describing the current state-of-the art in high-performance electronic parallel computing systems, as well as projections for where such systems will be in the next 15 years, based in large part on the 1995 Petaflops Workshop.

Although various proposals for innovative architectures exist, the mainstream view of the computer of the future is that it will utilize standard technology component microprocessors and operating systems with custom interconnection networks and judiciously chosen high-performance system services. Processor speeds currently are in the 1/4 to 1/2 gigaflop** range and are likely to increase by a factor of 20 in the next few years. Switch technology will improve at roughly the same rate. By 1998, multi-terafiop machines will be available at a cost of about $100 million. By the year 2015, petaflops systems with 5-10 thousand processors, each with up to 200 gigabytes of memory and 20 terabytes of on-line storage will be available. Of course, special-purpose dedicated computing systems with even better performance characteristics will be possible by then.

Richard Watson, Lawrence Livermore National Laboratory, discussed future trends and needs in high-performance mass storage systems. This arena has traditionally experienced a slower improvement rate in device performance compared to semiconductor processors and memories, and this trend is likely to continue in the future. Thus the performance bottleneck in future systems is likely to be input/output performance. Nevertheless, by the year 2000 systems storing as much as 1 petabyte of data transferred at up to 100 megabytes/sec will be available.

Other Technologies James Ellenbogen, Mitre Corporation, presented an overview of other technological approaches for building computing devices. Among the most revolutionary of these are molecular-scale nanomechanical computing systems and quantum computers based on interferences among coherent quantum waves. The best-developed technologies, however, are those connected with developing nanometer-scale electronic devices. Such devices include quantum dot cells, quantum-effect solid-state devices and molecular electronic circuit arrays. Quantum dots, for example, are small molecular "boxes' that can hold electrons. These dots can be used to make two-state cells that can be combined into chains, which can convey information without current flow. The cells can also be combined to form logic gates. Since such devices are electronic, they will be easier to integrate with conventional microelectronic devices and hence will be readily adopted and will receive large financial investments. Such technologies will pose stiff competition for biomolecular computing systems.

Biomolecular Technologies

Research in various biotechnologies may provide useful components in building molecular computers. Rob Lipshutz, Affymetrix described work being done to construct ordered oligonucleotide arrays using photolithography to deposit DNA on glass. With their technique they can create, on a single glass chip, a grid of cells (called features) in each of which are many copies of a particular DNA strand. They currently are working on feature sizes as small as 10 micrometers, allowing more than a million features to be stored on a single chip. Thus, for example, they can create an array containing every possible DNA 10-mer on one glass chip. The features can be detected using fluorizine and a scanning confocal microscope. In essence, they

have an addressable high-density storage array for DNA. Similar technology using inkjets instead of photolithography is being studied at the University of Washington.
Lloyd Smith, University of Wisconsin, discussed how solid-support chemistry can be used for DNA computing. The advantages of using solids compared to solutions are easy handling, reduced interference between oligonucleotides and easier purification. The disadvantages are limited surface area, decreased hybridization kinetics and surface-chemistry effects. They have designed a system that supports four computational primitives: mark a particular DNA strand (by hybridizing); unmark a strand (by dehybridizing); destroy a marked strand (using exonuclease); and append a sequence to the end of another sequence (using polymerase extension and ligation).
Erik Winfree and Nickolas Chelyapov, Laboratory for Molecular Science at USC, described their work in designing and implementing the above-mentioned stickers model of DNA computing. In order to increase the binding accuracy of the sticker strands they have used peptide nucleic acid (PNA) strands for the stickers. The major bottleneck in their system currently is achieving highly accurate separations.

Computational Issues
Richard Karp, University of Washington, presented a summary of the computational power of DNA computing. The basic approach of molecular computing to solving NP-hard problems has been to construct a DNA molecule for each potential solution and then use molecular operations to eliminate invalid solutions.
There are five basic operations used in DNA computing. The *extract* operation separates a tube T into two tubes: one with all molecules containing a particular substring; and another with the remaining molecules. The merge operation simply mixes two tubes. The *detect* operation simply checks if there are any strands in a tube. The *copy* operation amplifies all the strands in a tube. Finally, the *append* operation attaches a given string to the end of every molecule in a tube.
A length-n bit string can be encoded effectively as a DNA molecule. First assign two sequences: one to represent a 0-bit, the other to represent a 1-bit. Then concatenate these patterns separated by sequences that identify the position of each bit in the string. A tube containing all possible such sequences can be generated using copy, append and merge operations.
Lipton showed that using extract, merge and detect operations, the satisfiability of an *n*-variable Boolean formula of size *s* could be accomplished in *O(s)* operations using tubes with *O(2^n)* molecules. Adding the append operation allows testing the satisfiability of a size-*s* *n*-variable Boolean circuit within the same bounds. Boneh, Dunworth & Lipton showed that if the circuit represents a one-to-one function *f*, then in *O(s)* steps a tube can be created containing all strings of the form *(x, fx(z))*. To compute *f^(-1)(y)* one need only extract the sequence ending in *y* and then sequence it to determine *x*. This is the basis of breaking DES.
Two harder operations to perform are *intersecting* and *complement.* Intersect takes two tubes and creates a new tube containing the strings that were contained in both. Complement takes a tube and creates a tube containing the strings that are not in the first tube. If in addition either of these two operations is allowed, then any problem solvable by a conventional computer using a polynomial amount of storage can be solved in a polynomial number of molecular steps.
The strength of DNA computing is its high parallelism. The weaknesses are that the operations are slow and there is no communication within the tube. It is a hard problem, for example, to determine whether a tube contains two identical strands.
The limiting factor has been the volume of DNA that can be used. It takes 0.5 gram of DNA to make 2^(56) strands each of length 1000. To make 2^(70) strands of the same length takes 8 kilograms of DNA. Bach and Condon have given an approach that reduces the number of strands by building the potential solution strands progressively, eliminating impossible solutions before they are fully constructed. Another approach (not guaranteed to find the optimal solution) is to

generate only a sample of potential solution strands. Next identify the best strands, discarding the rest, and mutate some of the best strands. Then repeat.
Another limiting factor in DNA computing has been the problem of errors during the extract operation. Karp, Kenyon & Waarts have shown that to achieve separations with a desired error rate of delta using an extract operation with an intrinsic error rate of e requires and can be done with 0 (d'log~~J2) operations.

Conclusion and Recommendations

Near its conclusion, the workshop separated into two groups: one consisting of those most interested in computational issues, the other of those interested in biotechnology issues. In the final plenary session of the workshop, each group presented its observations and recommendations.

Issues of Interest to Computer Scientists

The computer science group prepared a long list of issues of interest to them, organized into five major areas:

Policy Issues: In terms of science policy, the main question is what level of investment should be made in these technologies, and in particular, which specific problems and areas are most worth pursuing now. An important issue will be to determine milestones in order to identify when the project is finished (either successfully or not).

Long-Term Goals: The first item on the list of long-term goals toward which biomolecular research should aim is, of course, the goal of solving a real-world problem better, either faster or more economically, than conventional electronic computers. Such a "killer application" does not seem to be achievable in the immediate future, but the preliminary work in breaking DES offers an example of a direction that might prove fruitful. Another goal of the project should be the development of technologies that will have uses in biotechnology other than just molecular computing. Such spinoffs will be a very significant benefit of this research. Other goals include obtaining a better understanding of natural computation, *i.e.*, characterizing more precisely what computations are being performed within living cells, and along with that using cells as components in computing systems.

Application Domains: The application areas toward which this technology should be targeted include, of course, virtually all aspects of molecular biology. Specific applications where effective molecular computing algorithms might be designed include DNA sequencing, molecular synthesis, drug design and related biological search problems. This research also will benefit the foundations of computer science and the development of new models of computation including those that occur naturally in living cells. Direct applications in information storage and retrieval and the development of new biochemical processes are also indicated.

Technical Issues: The most pressing technical issues to be pursued include:

- Characterizing existing biochemical operations as computational primitives -what biochemical steps can be performed and how efficient and reliable is each. This should include the creation of reliable simulation models of the chemical characteristics of these processes so that new experiments can be tested without expensive and time-consuming laboratory work.

- Designing new biochemical tools and techniques.

- Developing new models of computation.

- Developing error-correction techniques both for the final products of the reactions as well as in the biochemical processes themselves, just as living cells repair their defective molecules.

- Improving techniques for input/output to the biomolecular computer - better techniques to create/assemble the molecules that represent a problem instance and to detect/decipher the molecules that represent solutions to the problem.

- Developing hybrid computers with both biomolecular and conventional electronic components.

- Bionanofabrication - using nanotechnology to create and manipulate biocomputing molecules and processes.

- Designing asynchronous chemical computers where the operations/reactions occur continually as they pass through different stages of the system, much like a chemical refinery.

  Techniques: More specific steps and techniques that would aid the development of molecular computing include:

- Creating simple biomolecular computing components such as registers, adders, stacks, etc.

- Creating mechanisms for specific molecules to communicate with other specific molecules during reactions.

- Developing "garbage collection" techniques to remove unwanted molecular fragments.

- Using a molecular basis for the computation other than DNA, *e.g.*, oligosaccharides, artificial molecules, host-guest chemistry, marking/methylation.

- Developing local-search algorithms that are particularly suited to molecular computation.

- Designing useful subroutines with efficient biomolecular implementations that could be incorporated into larger algorithms.

- Designing analogue DNA computers.

Issues of Interest to Molecular Biologists
The biotechnology group identified the following areas of interest for further investigation:

- Enzymology. There is a need to characterize better the properties of enzymes that manipulate DNA, particularly their error rates. Better techniques for amplification also

need to be developed. This would be useful both for molecular computing and biological applications.

- Surface chemistry. Further efforts are needed to study the use of solid support techniques to interface with DNA.

- Separation. Significant improvements are needed in the accuracy of separating specific molecules from others in a test tube. The challenge here is so great that alternatives, such as the destroy operation, should be explored.

- Errors. Understanding and overcoming the error rates inherent in chemical reactions.

- Self-assembly. Designing reaction systems that would run without external intervention. The computational behavior of such systems, viewed as cellular automata, needs to be better understood.

- Compartmentalization of DNA computing systems. Living cells have specialized components for specific biomolecular tasks. It would be useful to design similar modularization into molecular computing devices.

- Secondary and tertiary structures. It may be possible to use these as part of the computing process, as opposed to current approaches that seek to avoid their effects.

- Biological applications of biomolecular computing. Rather than concentrating on building general-purpose molecular computing, the best payoff may come from designing special-purpose molecular computing processes for specific biological problems, *e.g., in vi*tro selection.

Conclusions

The consensus of the workshop was that biomolecular computing has great promise in its ability to process enormous quantities of information efficiently. In the very short time since Adleman's original paper, vast strides have been made in better understanding the capabilities of biomolecular computing systems and in overcoming technical difficulties. The notion of biomolecular computing has expanded from just a narrow view of DNA computing to encompass the entire concept of biological information processing. Although no general-purpose DNA computer that will be competitive with conventional electronic systems is likely in the near term, the wide variety of potential architectures for biomolecular systems make further research desirable. Moreover, work in this area will likely have additional impact on other biomedical applications.

Computer

Regular computations

Using the owner-computes rule, the compiler statically partitions regular computations across processors according to the selected data distribution and generates interprocessor communication for required nonlocal data. To avoid the overhead of computing ownership at runtime, static analysis is used to partition loops at compile time. In addition, several optimizations are employed to reduce the overhead of communication in Irregular computations. In many important applications, compile-time analysis is insufficient when communication patterns are data dependent and known only at runtime. In a subset of these applications, the

communication pattern repeats in several steps. Paradigm approaches these problems through a combination of flexible, irregular runtime support and compile-time analysis. Novel features in our approach are the exploitation of spatial locality and the overlapping of computation and communication.

Conclusion

Nowadays computers are in the centre of almost all areas of modern society and their computational abilities very often determine superiority of one product with respect to another. Since physical limits of traditional computers have been almost reached, current attention of the scientific and industrial communities is dedicated to new computational paradigms, e.g., quantum computing, DNA computing; and areas of their possible applications.

Problems related to the idea of infinite and infinitesimal are among the most fundamental in all natural sciences. Continuous efforts made to evolve existing numeral systems and to include infinite and infinitesimal numbers in them are explained on the one hand, by the necessity to study applications from the real world (it is sufficient to mention derivatives, integrals, and differential equations based on infinitesimals). On the other hand, by the necessity of Computer Science and Mathematics to create their foundations as solid as possible (starting from such a fundamental concept as number) and to use these foundations for creating new more and more powerful tools for applied calculus. For example, it is worthwhile to remind in this occasion that differential and integral calculus used nowadays practically everywhere in Science have been introduced by Newton and Leibniz using notions of infinitesimals.

There exist different ways to generalize traditional arithmetic for finite numbers to the case of infinite and infinitesimal numbers, however, traditional arithmetics developed for infinite numbers are quite different with respect to the finite arithmetic we are used to deal with. Moreover, very often they leave undetermined many operations where infinite numbers take part (for example, $\infty-\infty$, ∞/∞**,** sum of infinitely many items, etc.) or use representation of infinite numbers based on infinite sequences of finite numbers as it happens in many non-standard analysis approaches. In calculus, infinity leads to numerous difficulties. For instance, there exists a variety of viewpoints how indeterminate forms or divergent series should be treated. A canonical example is the series of alternating +1 and -1 for which Borel, Cesàro, Euler, Poincaré and many others propose different approaches leading to different answers.

These crucial difficulties lead to the current situation where parts of natural sciences related to infinity remain highly theoretical fields. In traditional computers, only arithmetical operations with finite numbers are realized. Fundamental difficulties described above precluded scientists from even trying to think about possibility to construct computers that would be able to work with infinite and infinitesimal numbers in the same manner as we are used to do with finite numbers. Consequently, the existing computers either give you simple *overflow* message or at maximum can return an answer that you have obtained ∞ as the result. At this point in the majority of cases you just stop your calculations or try to find some compactification technique (e.g., renormalization) and this is difficult and not always possible. It would be very important therefore, to be able to work with infinite values and indeterminate forms directly by a computer without necessity to return to re-formulation of theoretical models by humans.

The situation with infinitesimals is even worse. They are very often used in areas being foundations of the modern calculus: differentiation and integration but they cannot be expressed explicitly. Only qualitative descriptions like “let us take an infinitesimal δ” are used. But it is not clear how this δ can be obtained by operations +,-,/ and * from numbers 0 and 1 and, therefore, numerical calculus with infinitesimals (that could increase enormously accuracy of

computations) cannot be realised on a computer. Thus, if the passage from qualitative description of infinitesimals to their quantitative description could be realised, it would have an enormous impact in all the fields of practical numerical calculations from different applied areas.

For these reasons, a new unconventional approach was proposed for writing down finite, infinite, and infinitesimal numbers by utilizing only a finite number of symbols in a unique framework becomes very attractive. The new computational paradigm gives possibility to construct a new type of computer – Infinity Computer – able to store infinite, finite, and infinitesimal numbers and to execute arithmetical operations with all of them.

Finally, teaching Computer Science, Mathematics, and Physics at schools and universities becomes significantly simpler using the new computational paradigm (the Italian association of teachers of Mathematics has already expressed their high interest in this connection). Naturally, the necessity to use the Infinity Computer during studies opens wide business possibilities.

References

Alan Kaminsky and Hans-Peter Bischof. Many-to-Many Invocation: A new object oriented paradigm for ad hoc collaborative systems. 17th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA 2002), Onward! track, Seattle, Washington, USA, November 2002.

Alan Kaminsky and Hans-Peter Bischof. New architectures, protocols, and middleware for ad hoc collaborative computing. Middleware 2003 Workshop on Middleware for Pervasive and Ad Hoc Computing, Rio de Janeiro, Brazil, June 2003.

Chaithanya Bondada. Tuple Board: A new distributed computing paradigm for mobile ad hoc networks. Master's project, Rochester Institute of Technology, Department of Computer Science, January 2004.

David Gelernter. Generative communication in Linda. ACM Transactions on Programming Languages and Systems, 7 (1)80-112, January 1985.

Eric Freeman, Susanne Hupfer, and Ken Arnold. JavaSpaces Principles, Patterns, and Practice. Addison-Wesley, 1999.

G. Agha. *Actors: A model of Concurrent Computation in Distributed Systems*. MIT Press, 1986.

M. Betz. Omg's corba. *Dr. Dobb's Special Report*, (225):8–12, Winter 1994/1995.

Robert Grimm, Janet Davis, Eric Lemar, Adam MacBeth, Steven Swanson, Steven Gribble, Tom Anderson, Brian Bershad, Gaetano Borriello, and David Wetherall. Programming for pervasive computing environments. Technical Report UW-CSE-01-06-01, University of Washington, Department of Computer Science and Engineering, June 2001.

http://www.alphaworks.ibm.com/tech/tspaces. Retrieved

http://www.cs.rit.edu/~anhinga/publications/m2mi20020716.pdf.

http://www.cs.rit.edu/~anhinga/publications/mw2003cr.pdf.

http://www.theinfinitycomputer.com