The Next Big Thing: Position Statements

Munindar P. Singh
Dept of Computer Science
North Carolina State University
Raleigh, NC 27695, USA
singh@ncsu.edu

Daniel G. Bobrow
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304, USA
bobrow@parc.xerox.com

Michael N. Huhns
Dept of Electrical & Computer Engg
University of South Carolina
Columbia, SC 29208, USA
huhns@sc.edu

Margaret King
ISSCO, University of Geneva
54 route des Acacias
CH-1227 Geneva, Switzerland
Margaret.King@issco.unige.ch

Hiroaki Kitano
Sony Computer Science Lab
3-14-13 Higashi-Gotanda, Shinagawa,
Tokyo 141 Japan
kitano@csl.sony.co.jp

Ray Reiter
Dept of Computer Science
University of Toronto
Toronto M5S 1A4, Canada
reiter@cs.toronto.edu

Abstract

This panel is a celebration of artificial intelligence (AI). Basing its claims to interest on the past accomplishments of AI, it highlights some of the new exciting concepts and technologies that compete for the title *The Next Big Thing.*

1 Introduction

It is well-known among artificial intelligence (AI) researchers that AI has made significant contributions to computing research and practice. This fact is not always recognized in the rest of the computing community. Indeed, many people believe that "what works isn't AI." This is only half true. AI has traditionally focused on challenging problems, and when a problem is about to be understood, AI researchers tend to move to something more interesting. Indeed, topics that were identified by AI researchers, but are considered mainstream today include, among others

- functional programming
- object-oriented programming
- agents.

This panel seeks to identify the next major AI contributions to the rest of computing, while they are still considered AI! The goals of this panel are twofold:

- to raise awareness within AI of applications that will be the most amenable to AI techniques
- to identify results coming down the pike that may be used to raise awareness of AI contributions.

The panelists were asked to address the following questions. The first two questions define the essence of this panel. The next two help relate AI research with the rest of computing. The last question is the most speculative.

1. What major contributions might we expect from your area in about a decade?
   - what external technical factors might affect these?
   - what are their main applications?
2. What is the likely *next* major contribution from your area?
   - what external technical factors might affect it?
   - what are their main applications?
3. From which part of computing are we learning the most?
4. Which part of computing will we affect the most?
5. What advice can you offer researchers and funders?

In the context of the above questions, this panel includes commentaries by distinguished researchers representing major technology areas of AI. The area of *AI Programming* is covered by Bobrow; *Distributed AI* by Huhns; *Language Technology & Interfaces* by King; *Learning & Evolutionary Computing* by Kitano; and *Knowledge Representation* by Reiter. The moderator's contribution may be classified under *Cooperative Information Systems.*

2 BOBROW: Concurrent Constraint Models: A Basis for Intelligent Computation

Models have been at the heart of much of what has been done in artificial intelligence. We have built models of problems, of places, of things. We have built programs to reason about them, diagnose them when they exhibit faulty behavior, and explain how they behave. For each of these reasoners, we have often built different kinds of models. We are now starting to understand how to build simple composable models based on a hybrid concurrent constraint programming language [Saraswat *et al.*, 1994] augmented with hybrid-time semantics. The "hybrid" nature of the language enables the description of both continuous changes over time that might be described by a differential equation, and discrete changes of values and regimes of operations [Gupta *et al.*, 1994]. The support for "concurrency" enables compositionality; individual components and processes can be defined separately; simple composition of the separately defined components, and the strong semantic definition underlying the constraint system enables the composed system to run as a simulation with appropriate "real" parallelism (not just choosing one of the possible orderings of events).

As a complete programming language, it enables both environments of a system, and a task for the system to do to be expressed in the same language. As a constraint-based system, after composition of the model, the sense of locality can change. In the usual object oriented systems, information between composed objects defined separately is communicated through a protocol of messages between the objects, but all constraints on the object's behavior stay implicit, and local to the object. With constraint-based models, partial evaluation and constraint propagation allow information to flow to where it might be needed. We will describe a simple example with respect to a scheduling problem shortly. This style of programming, where the domain is the focus, constraints are the way of capturing the domain, and reasoners are the interpreter will be a very strong influence on computer science over the next decade. Similar systems will be used to control autonomous robots and immobots [Williams & Nayak, 1996].

As an example of a use of this technology, Xerox has been building a constraint-based modeling framework to enable construction of software to schedule plug-and-play xerographic engines used for copying and printing [Fromherz & Saraswat, 1995]. The software is intended to schedule the machine to produce paper at the fastest possible rate, given the description of the input stream, and the desired output. What makes this particularly difficult is that the composition of the machine is not known at the time the software is constructed, since different customers want different configurations of feeders, sorters, finishers, etc. These are plugged together in the field. This is a problem because the timing of feeding sheets in the beginning of the machine can be affected by delays that a finisher might impose because, for example, it needs time to move a stack of paper to a stapler, and no sheets should be moved to the stacker at that time. Because each module that can be connected together has a constraint-based description, this information can be propagated at the time these are plugged together, and taken into account by the generic scheduling software, a form of anytime optimizing planner.

The same types of models are being used to evaluate new design features to determine whether they increase productivity. I expect to see these kind of applications much more ubiquitous in the near future. Taking into account a wider range of model-based tasks, one can think of a generic device model as one that has five interfaces that: define its *inputs*; its *outputs*; its *task/goal*; set its *internal parameters/behavioral modes*; and provide *temporal control*. Given such a model in an appropriate declarative form, one can provide some of these interfaces, and compute the plausible values for the others. For example, control code for a device to achieve a certain task can be created (and compiled to low level machine code).

What is liable to affect the use of this most is whether the tools we are using can be put into the context of widely used systems and languages such as Java, and integrated into larger frameworks. This requires us to build these systems as component software so that they can be added easily to any preexisting architecture, framework, or algorithm. One thing we have learned from computer science is that there is no silver bullet. No single good idea, or system is enough to solve all problems. And systems need to be tuned to the work practice of the individuals and communities who use them. Sometimes solutions are more to be found in changing or leveraging the social environment in which a system is embedded than in clever software techniques [O'Day *et al.*, 1996]. And what we have to teach computer science is the power of symbolic reasoning, deep modeling, explicit task description and task independent reasoners.

3 HUHNS: A New DAI-Based Software Paradigm

The field of software engineering has exhibited almost glacial progress over the last twenty years, and appears to be in a malaise. Programmers still produce approximately the same number of lines of tested and debugged code per day as they did in 1975, in spite of several "silver bullets," such as structured programming, declarative specifications, object-oriented programming, formal methods, and visual languages.

This should not be surprising, for three reasons:

1. Software systems are the most complicated artifacts people have ever attempted to construct
2. Software systems are (supposedly) guaranteed to work correctly only when all errors have been detected and removed, which is infeasible in light of the above complexity

3. The effect of an error is unrelated to its size, i.e., a single misplaced character out of millions can render a system useless or, worse, harmful.

Software engineering attempts to deal with these problems by considering both the process of producing software and the software that is produced. The major goal for the software is that it be correct, and the major goal for the process is that it be conducted efficiently. One fundamental approach to meeting these goals is to exploit modularity and reuse of code. The expectations are that small modules are easier to test, debug, and verify, and therefore more likely to be correct, that small modules will be more likely to be reused, and that reusing debugged modules is more efficient than coding them afresh. This is the major result of the series of programming paradigms that have evolved from the machine language of the 1950's:

1. Imperative paradigm: procedure-oriented programming
2. Declarative paradigm: functional and logic programming
3. Interactive paradigm: object-based and distributed programming.

However, software has not kept pace with the increased rate of performance for processors, communication infrastructure, and the computing industry in general [Lewis, 1996]. Whereas processor performance has been increasing at a 48% annual rate and network capacity at a 78% annual rate, software productivity has been growing at a 4.6% annual rate and the power of programming languages and tools has been growing at an 11% annual rate. CASE tools, meant to formalize and promote reuse, have not been widely adopted [Iivari, 1996]. In addition to these sluggish rates for software, there is a legacy of approximately 50 billion[1] lines of Cobol, representing roughly 80% of all software written since 1960. It is unlikely that we can replace it anytime soon, even though maintaining it is a $3 billion annual expense.

The current "hot" computing paradigm is based on Java, and the ability it provides for users to download the specific functionality they want at the moment they request that functionality. This is leading to the rise of a software-component industry, which will produce and then distribute on demand the components that have the users' unique desired functionality [Yourdon, 1996]. However, because of this uniqueness, how can component providers be confident that their components will behave properly? This is a problem that can be solved by agent-based components that actively cooperate with other components to realize the user's goals.

3.1 A New Software Paradigm

Therefore, it is time to consider a completely different approach to software systems. We propose one based on the (intentionally provocative) recognition that

- errors will always be a part of complex systems
- error-free code can at times be a disadvantage
- where systems interact with the complexities of the physical world, there is a concomitant power that can be exploited.

We suggest an open architecture consisting of multiple, redundant, agent-based modules interacting via a verified kernel. The appropriate analogy is that of a large, robust, natural system.

3.2 Requirements for New Applications

There are a new class of applications evolving thanks to ongoing advances in computer systems. The applications have characteristics that lead naturally to an agent-based approach to their development [Woelk *et al.*, 1995]:

- They solve a specific business problem by providing a user with seamless interaction with remote information, application, and human resources.
- The identities of the resources to be used are mostly unknown at the time the application is developed.
- The pattern of interaction (workflow) among the resources is a critical part of the application, but the pattern might be unknown at the time the application is developed and might vary over time.

The development of these new applications requires improved programming languages and improved system services. These are not alternatives to such capabilities as OMG CORBA and Microsoft DCOM, but rather advanced features implemented at a higher level of abstraction and useful across multiple heterogeneous distributed computing environments.

Because each application executes as a set of geographically distributed parts, a distributed active-object architecture is required. It is likely that the objects taking part in the application were developed in various languages and execute on various hardware platforms. A simple, powerful paradigm is needed for communications among these heterogeneous objects. Due to the distributed nature of the application, an object might not always be available when it is needed. For example, an object executing on a PDA might be out of physical communications with the rest of the application.

Because the identities of the resources are not known when the application is developed, there must be an infrastructure to enable the discovery of pertinent objects. Once an object has been discovered, the infrastructure must facilitate the establishment of constructive communication between the new object and existing objects in the application.

Because the pattern of interaction among the objects is a critical part of the application and may vary over time, it is important that this pattern (workflow) be explicitly represented and available to both the application and the user. When an object has been discovered to be relevant to an application, the language for interaction and the pattern of interaction with the object must be

[1] 10^9.

determined. This interaction becomes part of the larger set of object interactions that make up the application. The objects collaborate with each other to carry out the application task.

Fundamentally, most business software modules are intended to be models of some real object within the business. A problem is that these modules are passive, unlike most of the objects they represent.

3.3 Multiagent-Based Development

Multiagent-based interoperation is a new paradigm distinguished by the features that requests are specified in terms of "what" and not "how," agents can take an active role, monitoring conditions in their environment and reacting accordingly, and agents may be seen as holding beliefs about the world.

We have initiated development of this new paradigm—a cooperative paradigm—based on interacting agents, active objects, and active wrappers of legacy components. The resultant methodology and language, interaction-oriented programming, represent a fundamental extension of the earlier paradigms, with greater expressive power, different conceptual foundations, such as beliefs held by components, and new modeling techniques.

4 KING: Language Technology and Interfaces

I take the language technology and interfaces area to cover all of speech technology, written language technology and aspects of interfaces such as the inclusion of images and sound. My own expertise is in written language technology, and my knowledge of the other areas involved scanty, being especially weak on the nonlanguage areas. Consequently, my intervention here will concentrate mainly on language technology. I hope that this will not be interpreted to mean that I believe work outside that subdomain to be less important. Similarly, I ask my speech colleagues to forgive me any unintentional misrepresentation of their field.

Major contributions in about a decade

Before looking at the critical developments I see coming in language technology, I want to be negative (or realistic, depending on one's point of view) and say what I do not see coming. I see no signs of a breakthrough which would allow us finally to conquer the semantic barrier by direct frontal attack. By this I mean that I see no developing technology which would allow us, in the general case, to resolve the problem of rampant ambiguity in natural language. Thus, whilst I can easily imagine succesful applications in constrained domains, I cannot, for example, imagine the development of a machine translation system which will produce high quality translation of arbitrary input, or a question answering system that relies on being able to simulate human understanding of an arbitrary text.

What I do see is the development of less ambitious technologies that will radically change the lives of very many people.

On the written language side, many research workers are already involved in finding ways round the semantic barrier, often taking as their starting point the recently developed capacity to process very large amounts of preexisting text [Armstrong, 1994]. Thus we see, for example, suggestions that it might be possible to spot the spelling mistakes in "Sea the dove sore" by relying on probabilities determined through part of speech tagging of large quantities of text, or in "I bought the mink in the super market" through probability of collocation. I do not mean to suggest that statistics-based techniques on their own are the magic solution to all problems: we know from the past decade or so that this is not so. But they do provide a new weapon, which, combined in certain cases with rule-based approaches, can give us a better succes rate in solving old problems. And perhaps even more importantly, they encourage a way of thinking in which being inventive about how to deploy the technology available is as important as tackling intractable fundamental problems directly.

On the spoken language side, I think a major contribution will be the development of systems which allow reliable multispeaker recognition of continuous speech, albeit perhaps in limited domains. Quite impressive prototypes already exist, and I believe we can foresee the incorporation of the technology into products in the relatively near future.

External factors that may affect this. One factor which will affect these developments strongly is, quite simply, market forces. Some products of language technology have become parts of everyday life: few secretaries would give up their spelling checkers, anybody who uses the web uses a search engine which incorporates a modest use of language technology. Even more ambitious products are becoming more widely known and used: CompuServe, amongst others, offers machine translation on the web, translators' workbenches are selling more and more, dictation systems are doing well. As the general public becomes increasingly aware of what can already be done, an appetite is created for yet more, and the manufacturers thereby encouraged to create and market more advanced language technology based products.

Another external factor is the development of the WWW: communication links across geographically dispersed communities and across different language communities creates a need for tools which facilitate communication. The need is accentuated by the existence of users communicating in a language which is not their own, or needing to communicate across languages.

One major technological issue will be the develpoment of robust systems: systems for both written and spoken language that can operate in noisy environments with nonexpert users.

Main applications. A hint at some applications is already contained in the above: improved spelling and grammar checkers, more user friendly and more robust dictation systems, voice input to information retrieval systems, improved information management tools, as well as more futuristic applications based on, for example, multilingual information management and retrieval. The list is, of course, far from being exclusive!

The likely *next* major contribution

The next major contribution is the integration of speech and language work. Funding agencies have been pushing in this direction for the last few years, as witnessed by the German National Project, Verbmobil, which aims at interpretation carried out over the telephone, an earlier similar Japanese effort and parts of the European Union Fourth Framework Programme [Alexandersson *et al.*, 1997; Gibbon, 1996; McTear, 1997]. As a result of these and other initiatives, there are already a few interesting prototypes around, but no widely available products, and the full impact of integrated speech and language technology has not as yet been realised.

Integration is not, of course, limited to speech and written language. I think we will see an explosion of multimedia systems incorporating imagery and sounds as well as speech and written language.

External factors that might affect this. Two major factors are likely to affect the development of integrated speech and language technology, favourably or unfavourably.

The first is the obvious factor of funding agencies forming and maintaining a consistent policy favouring work in this direction. The classical danger here is that of agencies expecting concrete and commercialisable results very quickly. When the results do not appear within a short time-frame, the agencies have a tendency to become disppointed and switch their attention somewhere else. Especially when this is coupled with a policy of funding for the short term only, four years or so being the maximum that a project can expect, and two years being typical, the result can easily be stop-and-start encouragement for work in a particular area, with all the associated problems of multiplying overheads in repeated start-up time and effort, and diminution of return on investment towards the end of a project, when the participants are investing much of their effort in writing convincing reports and prospecting for continuation funding.

This problem is particularly acute in the European setting, where funding is normally given only to consortia including groups from at least three different countries, and consortia are encouraged to change their membership from one project to the next. It is consequently difficult to produce medium or long-term consistent effort devoted to tackling specific problems or problem areas. However, the problem of acquiring funding for more than short term efforts is not particular to the language technology area.

The second factor, more specifically tied to that area, is that of initiating and maintaining creative dialogue between the speech community and the written language community. The problem here is not primarily one of good will. It is more a question of learning to reach across established habits of thought and ingrained ways of talking. An example may be useful. When a speech person talks about a corpus, he typically means a relatively small, but extensively annotated collection of items. When a written language person talks about a corpus, he typically means an extremely large collection of items, not necessarily including more than minimal annotation. Put the two of them together to talk about problems of corpus collection, storage and management, and cross-purposes discussion is almost inevitable. (Note that even the interpretation of "small" and "large" is influenced by the community one comes form: small here means several hundred, maybe a few thousand items, large means one or several million items). It would not be too difficult to find an extensive list of such areas of potential noncommunication.

Main applications. The range of potential applications is enormous. It starts with fairly unexciting but extremely important applications like dictation systems allowing for multispeaker input and natural speech, backed up by intelligent spelling checkers and grammar checkers, through to multilingual document retrieval and management and multilingual communication systems, passing through such a wide range of other applications on the way that the imagination boggles. Indeed, it quite seriously seems to me that we are limited in our imagination by what we know of the current state of technology. Once a breech is created by enabling integration of different means of communication, I foresee a flood of applications which we cannot yet even envisage. This is reinforced by experience with the WWW: there can be very few of us who would have imagined fifteen years ago that if we wanted to track down a rare book or find a modest hotel in New York or find out where a friend we last saw twenty years ago was now living, we would start with a web search.

The part of computing from which we are learning the most

If this is rephrased as "are we benefitting from," the answer must surely be hardware developments. Only increased memory capacity and processing power has made the enabling technology on which both written and spoken language rely possible.

The part of computing we will affect the most

I am very unsure of my answer here, but I suspect that the answer may once again be hardware. If we really think in terms of multimedia information and communication systems, it is not hard to imagine that we are influencing a push towards the development of more sophisticated sound and vision equipment incorporated into standard computer technology.

Advice for researchers and funders

My advice to researchers follows from all the above: be inventive about ways to use the technologies that are becoming available, about ways to combine new technologies with what is already there, and look at the interfaces between speech and language. We also need to fuse the written language and the speech communities, in order to create a language technology community.

My advice to funders is also clear: adopt a funding strategy which allows at least medium term research to be undertaken by relatively stable groups, and at the same time try to encourage research aimed at the integration of different modalities of communication.

5 KITANO: Building and Understanding Adaptive Complex Systems

Fifty years have passed since the invention of computers, and over forty years have passed since the birth of artificial intelligence. Assuming a 50 year economic cycle—based on the lead time between the conceptualization of a new technology and its maturity—computer science and AI will face a major leap in a decade. The explosive use of the Internet across the globe signifies the first major wave of the penetration of computer science into society, which impacts global economy as well as personal life. In the AI community, the success of the Computer Chess Challenge epitomizes state-of-the-art AI technologies that can be immediately recognized by a general audience. However, we need to move forward.

In recent years, there have been several attempts to elucidate future direction of AI research, including panel sessions at IJCAI-93, "Grand Challenge AI Applications" [Kitano *et al.*, 1993] and at AAAI-96, "Challenge Problems for Artificial Intelligence" [Selman *et al.*, 1996].

Looking back at the history of AI, success has been limited to domains where the world can be described by symbolic representations, and complete information is accessible. Computer chess is a typical example. Most expert systems also fall within this category. I have classified problems that AI systems may face into four classes [Kitano, 1994]: linear decomposition, linear approximation, nonlinear, and nonequilibrium problems. If we are to engage in the collective and organized effort to bring about *The Next Big Thing*, task domains need to be carefully chosen. Scientifically, the next big thing is going to be *Adaptive Complex Systems*. The challenge is how to build such systems and understand them.

5.1 Building Adaptive Complex Systems

Contextual-thickness as a foundation of integrated systems

One of the dreams of AI is to build an autonomous agent which can handle a broad range of tasks intelligently. This involves the capabilities of understanding and using language and behaving in the physical real world for specific tasks. What is an essential principle, or a concept which is salient in a complete agent? Together with my colleagues, I organized a closed workshop in 1994 at Sony Computer Science Laboratory titled "Grand Breakthrough." The participants were leading AI researchers in Japan. The discussion highlighted a concept of "contextually-thick system." A contextually-thick system is a robust system which can behave reasonably well for tasks with different contexts. A natural language system which can carry out dialogue with humans and translate written texts can be viewed as contextually-thicker than a traditional machine translation system. An autonomous system which can translate language and play soccer game is very contextually-thick. With the growing interests in building integrated systems, the concept of contextual-thickness will be highlighted. However, it is a nontrivial task to build such a system. Thus, a systematic and comprehensive approach needs to be taken from both engineering and science.

Multiaspect learning

One example of contextual-thickness can be illustrated by taking an example of how players should "learn" in the RoboCup. To be successful in the RoboCup, individual players' skills, teamwork, and coach's coaching ability should all be learned. The players need to learn before the match, during the match, and after the match. Learning of teamwork play before the match is allowed a substantial amount of time. However, if the team needs to adapt to the opponent's strategy, this type of learning must be done within 5–10 minutes. Also, the result of the match needs to be reflected to the next match against different opponents as well as the same opponents. In addition, the strategy needs to be changed if one of the players or an opponent player was involved in an accident and is not able to take part in the match anymore—thus, the system must learn to detect anomalies and how to switch strategies. This example from the RoboCup makes it clear that there are multiple aspects in learning when we try to build an integrated system for a comprehensive task. While RoboCup itself restricts a context to soccer games, it still has multiple aspects which naive use of current learning theories cannot cope with. Variations and constraints imposed on learning in various situations has not been systematically investigated. Thorough investigation on aspects of learning would be a possible first step towards a contextually-thick system.

Evolvable systems

Another potential breakthrough can be found in evolutionary computation. First, the increasing availability of reconfigurable hardware such as Field Programmable Gate Array enables genetic search and optimization technique to be used for real-time applications, such as telecommunication traffic optimization and real-time control of mobile robots. Second, increasing demands for evolving more complex structures will promote research into sophisticated genotype-phenotype mapping. Both avenue of biologically-inspired methods and formal methematical methods will be explored.

	Chess	RoboCup
Environment	Static	Dynamic
State Change	Turn taking	Real time
Info. accessibility	Complete	Incomplete
Sensor Readings	Symbolic	Nonsymbolic
Control	Central	Distributed

Table 1: Comparison of Chess and RoboCup

Infrastructure

In order to promote research for *The Next Big Thing*, an infrastructure needs to be implemented. Examples follow:

Standardized robot components. As the need for real world applications of AI increases, AI research platforms will increasingly emphasize robots and systems that interact with the physical world. However, the lack of standard components, which can be purchased at affordable prices hampers many researchers from actually going into the physical world. A collective effort shall have to be made to define a standard for reliable, flexible, scalable, and affordable physical robot components. Such a component set should provide a mother board with a real-time operating system, and a basic development environment.

RoboCup. The RoboCup, for example, is one of few attempts to provide a long-range challenge for the AI community [Kitano *et al.*, 1997; 1997]. It can be a significant testbed for AI and robotics research, as well as a showcase for a wide range of AI technologies, such as motion reconstruction for three-dimensional visualization of simulator, automatic commentary system, and intelligent studio systems. Since the RoboCup itself has been extensively publicized, here I only provide a table which contrasts Chess and RoboCup in terms of their domain characteristics (Table 1).

5.2 Understanding Adaptive Complex Systems: Computer-Aided Biology and Neuroscience

In the area of basic research, computer science and biology can establish a stronger relationship in the long run. Computer-based simulation and analysis systems, empowered by various AI and simulation techniques, will revolutionalize the way biology research is conducted. At the same time, AI will benefit a lot from the recent achievements in biology, particularly in the area of molecular neurobiology and functional brain mapping.

From computer science to biology

Already there have been many contributions from computer science to biology concerning instrumentation systems, databases, and protein structure predictions. However, I would argue that much more can be achieved.

The central focus of current molecular biology is to identify genes and their functions. Most papers published in *Nature*, *Science*, and *Genes and Development* report identification of new genes with significant functions. In addition, major efforts are begin made on genome sequencing, such as the human genome project, *C. elegans* genome project, and *E. Coli* genome project. The problem, however, is that completion of DNA sequencing and identification of genes is not sufficient to really understand biological systems. It is crucial to identify how these genes interact and what kinds of regulatory mechanisms control the whole process. However, this is an extremely difficult task, due to the involvement of a large number of interacting components.

The opportunities for AI and computer science are abundant. There is a pressing need to understand and possibly predict complex interactions among a large number of components. Components are interconnected with certain nonlinear functions with noise and delay. The identification of a set of parameters would significantly help biologists in further investigating the molecular mechanisms of biologically significant phenomena. Thus, the integration of simulation technology, search in a very large parameter space, logical deduction and induction of possible genetic networks from gene cloning and mutant analysis data, and intelligent visualization of highly complex information spaces would be extremely helpful.

From biology to computer science

Recent progress in molecular biology, particularly molecular neurobiology, and functional MRI technologies enables us to directly observe and manipulate the activities of the brain at both the neural (or subneural) level and the macroscopic level. In the past, much of the functionalities of the brain and its cognitive role have been discussed largely based on speculation. Now, we will be able to actually measure activities at various grain-sizes.

For example, a mouse's place memory formed at CA-1 region of hippocampus was directly measured by microscopic electrobes, and the effects of specific signal transaction channels was identified using an induced knockout mouse [Tsien *et al.*, 1996; McHugh *et al.*, 1996]. The knockout mouse is a transgenic mouse whose specific gene has been knocked out. Induced knockout is a means to knockout a specific target gene epigenetically. Such experiments resolve disputes over the internal representation of memory and learning. From more computational aspects, the relationship between biological processes and reinforcement learning was reported in [Schultz *et al.*, 1997].

These studies greatly expand our knowledge of the brain and neural systems, which is the foundation of intelligence. I believe a massive inflow of knowledge from the rapidly developing areas of molecular neuroscience, functional brain mapping, and molecular biology in general will drastically change the directions and quality of research in the AI community.

6 REITER: Knowledge Representation

AI practitioners tend to view knowledge representation (KR) as a rarefied, theoretical side of their field, and they do not normally look to it for a source of applied research. But in the past, foundational work in KR has led quite directly to important practical benefits. Before splitting off into their respective subcommunities, logic programming, deductive databases, description logics for taxonomic reasoning, and agent programming all had intellectual and methodological origins in KR.

I have two candidates for *The Next Big Thing* to emerge in the coming decade from research in knowledge representation, both closely related to each other.

Dynamical systems

A good case can be made that one of the most pressing needs for AI these days is a solid theoretical and computational account of actions. It is a challenge to capture, in a single formal and computational framework, the full range of characteristics associated with dynamical systems: the frame, ramification and qualification problems, exogenous and natural events, chance events and the unpredictability of action effects, complex actions and procedures and the ability of an agent to perform such actions, time, concurrency, planning, hypothetical and counterfactual reasoning about action occurrences and time, perceptual actions and their effects on an agent's mental state, the complex relationships among reasoning, perception and action, belief revision in the presence of conflicting observations, etc. Almost every AI system needs such a story because, virtually without exception, the phenomena AI wishes to model have dynamic components. Consider: robotics, planning and scheduling, natural language communication and speech acts, qualitative physics, database transaction processing, diagnosis and repair of time-varying systems, simulation, etc.

The good news is that, largely as a result of substantial foundational work by the KR community over the past few years, we are well along in the development of such a theory. The frame, ramification and qualification problems have largely been solved (e.g., [Kartha & Lifschitz, 1994], [Reiter, 1991], [Shanahan, 1997], [Sandewall, 1994], and [Thielscher, 1996]). This means that dynamical systems can now be given purely logical characterizations, thereby eliminating the need for the often ad hoc, procedural and hybrid approaches to modeling dynamics that AI systems have employed in the past. I think it's safe to predict that the next 10 years will produce "off-the-shelf" general logics and implementations suitable for incorporation into specifications and programs for domain dependent applications. There are already signs of this in the well-developed axiomatizations for the temporal action languages of [Sandewall, 1994] and Kowalski-Sergot as extended by [Shanahan, 1997], for the family of $\mathcal{A}$ languages [Gelfond & Lifschitz, 1993], and for the situation calculus incorporating time, concurrency and continuous events [Pinto, 1994]. Moreover, various action-centered logic programming languages are now in use, for example the temporal logic-based MetateM [Fisher, 1994], and the situation calculus-based GOLOG family of languages [Levesque *et al.*, 1997; De Giacomo *et al.*, 1997].

With respect to "the next big thing" to emerge from these developments, I believe the most significant outcome is that they provide action-centered, purely logical programming languages. The computations of such programs are determined through their interactions with an axiomatic knowledge base describing actions (their preconditions and effects), as well the initial state of the world being modeled. This feature makes these languages of some interest beyond AI. In computer science, for example, software engineering, database transaction processing, discrete event simulation, computer animation, systems programming and programming language semantics are all areas where dynamics is central, and where "knowledge-based" specifications and programming can be profitably exploited. Further afield, the situation calculus can be seen as a generalization of classical discrete event control theory, providing for control systems *with* an explicitly represented knowledge base. This means that situation calculus-based programming languages, like GOLOG, are well suited to implementing control systems and their simulators, including "hybrid" control systems involving both discrete and continuous time and events.

Agent programming

There are many perspectives on what counts as agent programming. For the purposes of this presentation, I shall take it to concern the design of high level programming languages for (possibly distributed, possibly multi-agent) systems in which each agent is encapsulated in a program, communication among agents is possible, and each agent program respects suitable logical specifications involving the intensional states (knowledge, belief, desires, intentions) of itself and of other agents. Here, "knowledge and beliefs" are what you would expect, "desires" are the agent's goals, and "intentions" are those goals that the agent is currently committed to achieve. But of course, other intensional attitudes are possible, and sometimes desirable (permissions, obligations, etc). The key problem here is to capture in suitable programming languages, agent behaviors that emerge from the interactions among these intensional states, the agent's environment, and the agent's repertoire of basic actions that it can perform in its world, including perceptual actions that change its knowledge and belief states. Notice that these agents ultimately act in the world, so that all of the issues raised in my previous discussion of dynamical systems arise here as well (the frame, ramification and qualification problems, concurrency, temporal reasoning, etc). Moreover, they arise with a vengeance because they must be adapted to deal with intensional attitudes as well as ordinary actions [Scherl & Levesque, 1993].

For many years now, providing formal and compu-

tational foundations for intensional attitudes and belief change has been meat-and-potatoes research for KR, so it should come as no surprise that much current activity in agent programming is informed by this work. The basic logical framework for intentions stems from [Cohen & Levesque, 1990] and the influential BDI (beliefs, desires and intentions) framework for multiagent systems is due to [Rao & Georgeff, 1992]. During the past several years, Ch. Meyer and his group at Utrecht have been sytematically developing formal accounts of rational agents [van der Hoek, *et al.*, 1997]. There have been a few implementations of multiagent programming languages that were motivated by the above considerations, for example the Procedural Reasoning System [Georgeff & Ingrand, 1989] and its successors, and Shoham's Agent0 [Shoham, 1993], but currently there remains a large gap between the logical ideals characterized by the foundational research in KR and actual implementations. This situation will change, of course, and in the next 10 years we can expect extremely rich programming languages that represent, at a very high level of abstraction, the logical and procedural properties of agents with mental states.

The consequences of such developments for computer science, especially programming language theory, software engineering and databases are obvious, and I won't repeat the standard arguments. Nor will I rapture on about how the world will be a better place what with smart web browsers, personal assistants and so on. Certainly, someone will make a lot of money from agent programming, but alas, it probably won't be me.

7 SINGH: Interaction-Oriented Programming and Social Constructs

Open information environments are heterogeneous, distributed, dynamic, large, and frequently comprise autonomous components. For these reasons, they require solutions that marry artificial intelligence (AI) and traditional techniques to yield extensibility and flexibility. Agents are a result of this marriage. Unfortunately, many current agent approaches are "autistic" with all of the attendant limitations of centralization in open environments.

Multiagent systems require (potentially autonomous) agents to behave in a coordinated manner. Therefore, the designer of a multiagent system must handle not only the application-specific aspects of the various agents, but also their interactions with one another. However, constructing multiagent systems manually can lead to unnecessarily rigid or suboptimal designs, wasted development effort, and sometimes to the autonomy of the agents being violated. It is these difficulties that lead many to the centralized approaches.

We propose *interaction-oriented programming (IOP)* as a class of languages, techniques, and tools to develop multiagent systems. Briefly, IOP focuses on what is between, rather than within, agents. Interactions may conveniently be classified into three layers (lower to upper):

- *coordination*, which enables the agents to operate in a shared environment [Hewitt, 1977]
- *commitment*, which add coherence to the agents' actions [Castelfranchi, 1995; Singh, 1997]
- *collaboration*, which includes knowledge-level protocols on commitments and communications [Grosz & Kraus, 1996; Singh, 1994].

Informal concepts, such as competition, may be classified into different layers: auctions require only coordination, whereas commerce involves commitments, and negotiation involves sophisticated protocols.

Tenets. Our key tenets are as follows. We describe and defend these in some of our other work.

1. The openness, autonomy, and heterogeneity of modern systems are often sacrosanct.
2. Correctness or data integrity may be difficult to characterize, but *coherence* is still crucial.
3. Complex interactions greatly exacerbate the difficulties in developing robust multiagent systems.
4. *Customizable* approaches can yield productivity gains that far outweigh any performance penalties.

Interaction has been studied, albeit fragmentarily, in distributed computing (DC) [Agha *et al.*, 1993; Milner, 1993], databases (DB) [Gray & Reuter, 1993], and distributed AI (DAI). The DB and DC work focuses on narrower problems, and eschews high-level concepts. Thus it is less flexible, but more robust, than the DAI work. The challenge is in achieving rigor and flexibility.

It is instructive to evaluate the autonomy allowed by different DB approaches. Transactions publish results cautiously [Gray & Reuter, 1993]: they preserve autonomy of the consumers of those results, but violate autonomy of the subtransactions producing the results. Extended transaction models release results more liberally, and restrict the autonomy of their components in executing compensating subtransactions as needed [Elmagarmid, 1992]. Both of the above leave it to the application to handle errors or discrepancies that arise after a given transaction has completed. By contrast, Spheres of Control publish their results early, but require control over the activities consuming the results [Davies, 1978].

Toward an ontology of commitments. We believe *social commitment* is the key abstraction for supporting coherent interactions, while preserving autonomy. The DB approaches deal with passive objects, and view commitments as depending solely on the computation that "commits," not on the *interplay* between the interacting computations. Agents give us a handle on persistent activity, but without commitments they are quite limited. In order to formalize commitments, we observe that

1. agents can be structured, and are recursively composed of heterogeneous individuals or groups of agents [Singh, 1991]

2. agents are autonomous, but constrained by commitments—or we would have chaos!
3. social commitments cannot be reduced to internal commitments, which apply within an agent—the relationships among these concepts cannot be definitional
4. commitments are, in general, revocable; the clauses for revoking them are no less important than the conditions for satisfying them!
5. commitments arise, exist, are satisfied or revoked, all in a social context; commitments not only rely on the social structure of the groups in which they exist, but also help create that structure.

We believe that these observations, especially 4 and 5, have not received the attention they merit. Taking them seriously leads to a view of commitments as relating a *debtor*, a *creditor*, and a *context*. A number of natural operations can be defined on commitments, along with social policies that agents acquire when they (autonomously in some cases) adopt a role.

The above gives a flavor of IOP, and its social component. We invite the reader to join in its investigation.

Questions. We would suggest that the next result of IOP would be the osmosis of social constructs into mainstream computing, with tools for IOP being realized within a decade or so, for open applications. The key external factors will be advances in DC and the emergence of standards for agent interaction and "society management." IOP will benefit the most from DC and have the greatest influence on DB. If we may venture some advice to researchers it would be not to lose sight of principles when playing with the enticing technologies of today; our advice to funders would be the same!

References

[Agha *et al.*, 1993] Agha, Gul; Frølund, Svend; Kim, WooYoung; Panwar, Rajendra; Patterson, Anna; and Sturman, Daniel; 1993. Abstraction and modularity mechanisms for concurrent computing. *IEEE Parallel and Distributed Technology* 3–14.

[Alexandersson *et al.*, 1997] Alexandersson, Jan; Reithinger, Norbert; and Maier, Elisabeth; 1997. Insights into the Dialogue Processing of Verbmobil. In *Proceedings of the Conference on Applied Natural Language Processing.*

[Armstrong, 1994] Armstrong, Susan, editor; 1994 *Using Large Corpora.* MIT Press.

[Cohen & Levesque, 1990] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[Cole, 1997] Cole, A Ronald, editor; 1997. *Survey of the State of the Art in Human Language Technology.* Cambridge University Press, to appear. Also available at http://www.cse.ogi.edu/CSLU/HLTsurvey/ HLTsurvey.html

[Castelfranchi, 1995] Castelfranchi, Cristiano; 1995. Commitments: From individual intentions to groups and organizations. In *Proceedings of the International Conference on Multiagent Systems.*

[Davies, 1978] Davies, Charles T. Jr.; 1978. Data processing spheres of control. *IBM Systems Journal* 17(2):179–198.

[De Giacomo *et al.*, 1997] G. De Giacomo, Y. Lespérance, and H. J. Levesque. Reasoning about concurrent execution, prioritized interrupts, and exogenous actions in the situation calculus. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1997.

[Elmagarmid, 1992] Elmagarmid, Ahmed K., editor; 1992. *Database Transaction Models for Advanced Applications.* Morgan Kaufmann.

[Fisher, 1994] M. Fisher. A survey of concurrent MetateM - the language and its applications. In *Proc. First International Conference on Temporal Logic (ICTL)*, 1994. Lecture Notes in Computer Science, 827, Springer-Verlag.

[Fromherz & Saraswat, 1995] Markus P. J. Fromherz and Vijay A. Saraswat, "Model-Based Computing: Using Concurrent Constraint Programming for Modeling and Model Compilation." In U. Montanari and F. Rossi (eds.), *Principles and Practice of Constraint Programming - CP'95*, Springer-Verlag, LNCS 976, 629–635 Sept. 1995.

[Gelfond & Lifschitz, 1993] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

[Georgeff & Ingrand, 1989] M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, San Francisco, CA, 1989.

[Gibbon, 1996] Gibbon, Dafydd, editor; 1996. *Natural Language Processing and Speech Technology.* Mouton de Gruyter.

[Gray & Reuter, 1993] Gray, Jim and Reuter, Andreas; 1993. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann.

[Grosz & Kraus, 1996] Grosz, Barbara J. and Kraus, Sarit; 1996. Collaborative plans for complex group action. *Artificial Intelligence* 86(2):269–357.

[Gupta *et al.*, 1994] Vineet Gupta, Radha Jagadeesan, Vijay Saraswat, and Danny Bobrow, "Programming in Hybrid Constraint Languages." In Antsaklis, Kohn, Nerode and Sastry, editors, *Hybrid Systems II*, LNCS 999, Springer Verlag. Proceedings of the Hybrid Systems Workshop, Cornell, October 1994.

[Hewitt, 1977] Hewitt, Carl; 1977. Viewing control structures as patterns of passing messages. *Artificial Intelligence* 8(3):323–364.

[Iivari, 1996] Juhani Iivari, "Why Are CASE Tools Not Used?" *Communications of the ACM*, 39(10):94–103, October 1996.

[Kartha & Lifschitz, 1994] G. N. Kartha and V. Lifschitz. Actions with indirect effects (preliminary report). In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 341–350, 1994.

[Kitano *et al.*, 1997] Kitano, H., et al., "RoboCup: A Challenge AI Problem," *AI Magazine*, Spring, 1997.

[Kitano *et al.*, 1997] Kitano, H., et al., "RoboCup Challenge 97," *Proc. IJCAI*, 1997.

[Kitano *et al.*, 1993] Kitano, H., et al., "Grand Challenge AI Applications," *Proc. IJCAI*, Chambery, 1993.

[Kitano, 1994] Kitano, H., "The Challenge of Massive Parallelism," *Massively Parallel Artificial Intelligence*, The MIT Press, 1994.

[Levesque *et al.*, 1997] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: a logic programming language for dynamic domains. *J. of Logic Programming, Special Issue on Actions*, 31(1-3):59–83, 1997.

[Lewis, 1996] Ted Lewis, "The Next $10,000_2$ Years: Part II," *IEEE Computer*, May 1996, pages 78–86.

[McHugh *et al.*, 1996] McHugh, T., Blum, K., Tsien, J. Z., Tonegawa, S. and Wilson, M., "Impaired Hippocampal Representation of Space in CA1-Specific NMDAR1 Knockout Mice," *Cell*, 87:1339–1349, 1996.

[McTear, 1997] McTear, F. Michael; 1997. Spoken Dialogue Tehcnology: Enabling the Conversational User Interface. *Distributed at the DUS/ELSNET Bullet Course on Designing and Testing Spoken Dialogue Systems*, April.

[Milner, 1993] Milner, Robin; 1993. Elements of interaction. *Communications of the ACM* 36(1):78–89. Turing Award Lecture.

[O'Day *et al.*, 1996] Vicki O'Day, Daniel Bobrow, and Mark Shirley, "The Social-Technical Design Circle." In *Proceedings of the Conference on Computer-Supported Cooperative Work*, November 1996, ACM Press.

[Pinto, 1994] J. A. Pinto. *Temporal Reasoning in the Situation Calculus.* PhD thesis, University of Toronto, Department of Computer Science, 1994.

[Rao & Georgeff, 1992] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann, San Francisco, CA, 1992.

[Reiter, 1991] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, San Diego, CA, 1991.

[Sandewall, 1994] E. Sandewall. *Features and Fluents: The Representation of Knowledge about Dynamical Systems.* Oxford University Press, 1994.

[Saraswat *et al.*, 1994] Vijay Saraswat, Radha Jagadeesan, and Vineet Gupta, "Programming in Timed Concurrent Constraint Languages." In B Mayoh, E Tougu, and J Penjam, editors, *Constraint Programming*, Springer Verlag, 1994.

[Scherl & Levesque, 1993] R. Scherl and H. J. Levesque. The frame problem and knowledge producing actions. In *Proc. AAAI-93*, pages 689–695, Washington, DC, 1993.

[Schultz *et al.*, 1997] Schultz, W., Dayan, P. and Montague, P. R., "A Neural Substrate of Prediction and Reward," *Science*, 275:1593–1599, 1997.

[Selman *et al.*, 1996] Selman, B., Brooks, R., Dean, T., Horvitz, E., Mitchell, T. and Nilson, N., "Challenge Problems for Artificial Intelligence," *Proc. AAAI*, Portland, 1996.

[Shanahan, 1997] M. P. Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia.* MIT Press, 1997.

[Shoham, 1993] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[Singh, 1991] Singh, Munindar P.; 1991. Group ability and structure. In Demazeau, Y. and Müller, J.-P., editors, *Decentralized Artificial Intelligence, Volume 2.* Elsevier Science B.V. / North-Holland, Amsterdam. 127–145.

[Singh, 1994] Singh, Munindar P.; 1994. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications.* Springer Verlag, Heidelberg, Germany.

[Singh, 1997] Singh, Munindar P.; 1997. Commitments among autonomous agents in information-rich environments. In *Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW).*

[Thielscher, 1996] M. Thielscher. Causality and the qualification problem. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96)*, pages 51–62. Morgan Kaufmann, San Francisco, CA, 1996.

[Tsien *et al.*, 1996] Tsien, J. Z., Chen, D. F., Gerber, D., Tom, C., Mercer, E., Anderson, D., Mayford, M., Kandel, E. and Tonegawa, S., "Subregion- and Cell Type-Restricted gene Knockout in Mouse Brain," *Cell*, 87:1317–1326, 1996.

[van der Hoek, *et al.*, 1997] W. van der Hoek, B. van Linder, and J-J. Ch. Meyer: An integrated modal approach to rational agents. In *Proc. 2nd AISB Workshop on Practical Reasoning and Rationality*, pages 123–159, Manchester, United Kingdom, April 7-9, 1997.

[Williams & Nayak, 1996] Brian C. Williams and Pandurang Nayak. "Immobile Robots: AI in the New Millennium." *AI Magazine*, 17(3), Fall 1996.

[Woelk *et al.*, 1995] Darrell Woelk, Michael Huhns, and Christine Tomlinson, "Uncovering the Next Generation of Active Objects," *Object Magazine*, July-August 1995, pages 33–40.

[Yourdon, 1996] Edward Yourdon, "Java, the Web, and Software Development," *IEEE Computer*, August 1996, pages 25–30.

IJCAI-97

Proceedings of the Fifteenth International
Joint Conference on Artificial Intelligence

Nagoya, Japan
August 23–29, 1997

Volume 2

Sponsored by the
International Joint Conferences on Artificial Intelligence, Inc. (IJCAII)
Japanese Society for Artificial Intelligence