PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT.

A REVIEW OF MCC'S ACCOMPLISHMENTS AND STRATEGIC OUTLOOK FOR KNOWLEDGE-BASED SYSTEMS

Edited by **Eugene Lowenthal, Vice President,**

Microelectronics and Computer Technology Corporation, Austin,Tx.

ABSTRACT

MCC's Advanced Computer Architecture (ACA) program is divided into three large laboratories and one small one. The large labs are tasked with continually assessing user/industry needs on the 5-to-10 year horizon and providing science and technology responsive to emerging requirements. In the most general terms, the Artificial Intelligence Laboratory is charged with advancing the **functionality** of Knowledge-based Sytems (KBS), while the mission of the Human Interface Laboratory is to enhance the **usability** of KBS, and that of the Systems Technology Laboratory is to provide KBS platforms with superior **performance and capacity.** In addition there is a recently formed laboratory called Experimental Systems which represents MCC's first publicly (DARPA) funded project. This is a three-year effort aimed at tools for rapid prototyping of alternative hardware architectural designs.

The ACA project managers summarize the current projects and past accomplishments of each of the laboratories. Finally, there is a brief description of an effort to establish new long range goals for the next decade.

1. Editor's Introduction

The organizers of FGCS'88 were very kind to invite me to submit a paper. Unlike the other papers offered to the Conference, however, this is not a scholarly contribution. Rather it is a manager's (admittedly proud) chronicle of the accomplishments and aspirations of a very talented ensemble of computer scientists — they take the credit for everything reported here.

I am director of a research program called Advanced Computer Architecture (ACA) which has been in business about five years. The name of our program (the largest of five at MCC) reflects the original charter which was to compete head-on with the ICOT-sponsored effort in fifth generation computer systems. As the program took shape and evolved over the past several years, however, it is clear that our "center of gravity" is now not so much computer architecture as knowledge-based systems. Certainly there is a strong overlap with the original charter, but we are emphasizing innovation in software much more than in hardware. Thus, for example, less energy is being put into novel parallel hardware and more into the language and software technologies required to exploit current and anticipated commercial offerings in parallel hardware.

The program's concentration on knowledge-based systems (KBS) should not be construed as an exclusive focus on artificial intelligence. On the contrary, we start from the notion that most software systems ("intelligent" or otherwise) should be knowledge-based, that a wide variety of benefits derive from extracting the logic of an application and representing it declaratively in a knowledge base. A clear example of how knowledge bases can be leveraged is our approach to human interfaces, which is described later. Throughout ACA we are concerned with knowledge representation — expressive languages with efficient compilation and execution.

ACA is divided into three large laboratories and one small one. The large labs are tasked with continually assessing user/industry needs on the 5-to-10 year horizon and providing science and technology responsive to emerging requirements.[1] In the most general terms, the Artificial Intelligence Laboratory is charged with advancing the **functionality** of KBS, while the mission of the Human Interface Laboratory is to enhance the **usability** of KBS, and that of the Systems Technology Laboratory is to provide KBS platforms with superior **performance and capacity.**

All of this work is jointly sponsored by member companies of the MCC research consortium; that is, the research is privately funded. In addition there is a recently formed laboratory called Experimental

[1] Nearer term, less risky research remains the province of MCC's industrial partners, as does "productization" of MCC-developed technology. Nonetheless, since MCC's inception there have been many commercially valuable results and there is now a steady pipeline of technology being transferred to the sponsoring companies.

Systems which represents MCC's first publicly (DARPA) funded project. This is a three-year effort aimed at tools for rapid prototyping of alternative hardware architectural designs.

The remaining sections summarize the current projects and past accomplishments of each of the laboratories. The final section deals with the difficult task of looking to the end of the century to establish new goals for ACA's long range research. From this point forward I have acted much more in the capacity of an editor than author. The substantive content of what follows was extracted from the annual plans and progress reports of ACA's outstanding research leadership team:

AI Laboratory: Douglas Lenat and Charles Petrie

Human Interface Laboratory: James Hollan and Elaine Rich

Systems Technology Laboratory: Haran Boral, Won Kim and Carlo Zaniolo

Experimental Systems Laboratory: Robert Smith.

2. Artificial Intelligence Laboratory

CYC: The goal of this ambitious, intensive effort is to encode a very large knowledge base of interrelated concepts encompassing the "common-sense" knowledge shared by modern humans. The potential impact of CYC is too great to detail in an overview paper. Suffice it to say that success (which is by no means assured) would have profound effects on progress in natural language processing, machine learning, and expert systems technology. It can be effectively argued that we are at a "plateau" in AI research and that a concentrated effort such as CYC to develop a common-sense knowledge base is an absolute prerequisite for moving to a significantly higher plateau.

Among the key problems facing CYC are: the development of an appropriate knowledge representation language for common-sense knowledge; determination of CYC's ontology — the fundamental structure of knowledge; and the creation of tools that will allow many individuals to introduce new knowledge in an efficient and coordinated manner.

The CYC project has made excellent progress on all these fronts, essentially staying on the original 10-year schedule laid down in 1984. A substantial core of common-sense knowledge, the basic foundation upon which the structure will grow, has been put in place. There now exist powerful editing and visual browsing tools geared to keeping multiple "knowledge enterers" productive while operating from the common ontological model.

A new knowledge representation language called CYCL, has emerged, not as a theory-driven effort, but as a function of the very difficult task of encoding common-sense knowledge in all of its richness. In addition to typical reasoning and representation features, CYCL/CYC necessarily addresses issues of time, space, causality, hypothetical worlds, beliefs/contradictions, guessing, plausibility, introspection (meta-level reasoning), causality, and much more. As CYCL has begun to stabilize, it is finding use in other parts of ACA and in carefully selected collaborative efforts outside of MCC.

The main thrust for the future is a progressively accelerating effort to manually encode more and more knowledge in the context of what has already been stored. In theory, if CYC is successful, it will ultimately learn how to acquire and incorporate new knowledge on its own, thereby eliminating our current dependence on hand-coding.

Throughout the project we will be evaluating CYC as it grows, to see if it gets better at disambiguating natural language segments, removing the "brittleness" from complex expert systems, facilitating cooperation among separately developed expert systems, and other anticipated uses.

Proteus: The Proteus expert system development tool was among the first technologies to be transferred to MCC shareholders, and it is the first one to result in a commercial product (viz. the NCR Design Advisor).

Proteus began in 1984 with two observations:

- The importance of defeasible reasoning in general, and default reasoning in particular, had been almost completely ignored.
- Among traditional expert system development environments, there was a severe trade-off between efficiency and conceptual simplicity, on the one hand, and expressiveness and functionality, on the other.

A key decision, based on the first observation above, was that a *truth maintenance system* (TMS) should be a cornerstone in the design of a knowledge-based application development system. This provides a mechanism for defeasible reasoning, the importance of which has been recognized for constructive tasks, especially design, which involves heuristic choices subject to iterative revision.

Proteus contains three major advances in truth maintenance technology: A *complete* algorithm for belief status labeling — if a solution is possible, Proteus will find it; A unique integration with rules and frames, including inheritance; and a novel method of representing domain knowledge to control dependency-directed backtracking.

Our approach to addressing the simplicity/expressiveness issue has been to accommodate multiple paradigms within a single system and language. However we are committed to the view that these paradigms must be integrated at the architectural level, rather than merely combined in a "toolkit". The design of Proteus includes a number of established reasoning and representation techniques: nonmonotonic truth maintenance with backtracking, restricted predicate logic, frames with multiple inheritance and metaclasses, forward and backward inference, and Lisp s-expressions.

Future releases of Proteus will introduce enhancements to the system's functionality, human interface, and especially performance. Work in progress includes a complete nonmonotonic TMS, support for multiple inheritance and metaclasses, temporal reasoning, hypothetical worlds, a graphics-oriented development environment, metareasoning to facilitate intelligent control of inference, and dramatic speedups for backward and forward inference. With respect to the latter, we have recently developed a rule compiler that achieves over 100K LIPS on a 68020-based Unix system, and we will be porting this technology to Proteus.

One of the interesting offshoots of Proteus was *Argo*, a modest effort carried out in cooperation with MCC's VLSI CAD project. Argo introduced basic learning and analogical reasoning techniques to Proteus. These capabilities were successfully demonstrated in a small circuit design application in which new rules were learned from training examples resulting in substantially faster execution times and improved design quality for similar circuit specifications subsequently presented to Argo.

Antares: Two of the major problems confronting the developers of KBS are that independently developed expert systems cannot in general be interconnected to work together and that there is no established means for several experts to develop an expert system jointly. Antares, (a new project) will utilize principles of distributed KBS to solve these problems. The result will be a set of methods for interconnecting separately developed KBS to enable them to cooperate in solving problems beyond the capabilities of any one of the KBS. This proposed system can be viewed as a new type of "shell" for the modular development of KBS. It will use the common-sense knowledge in CYC to provide a basis for globally consistent semantics among the KBS, but will not require globally consistent beliefs.

Among the advances required to achieve the Antares objective are: mechanisms for control that enable cooperative problem solving behavior among a set of KBS; representation of principles of "self-interest" within CYC that can be the basis of negotiation and cooperation among agents; and implementation of a general expert system whose expertise is the control and enforcement of cooperation among other expert systems.

Planning and Decision Making: The primary characteristic of current decision support tools is that they are numeric. They are typically based on utility theory and require significant quantification of variables by the user. This quantification task can be difficult or even meaningless. Only recently has work been done to derive appropriate numbers through user-supplied partial orderings. But knowledge acquisition is not the overwhelming disadvantage of the numeric approach. The real problem is that conclusions are not the result of symbolic reasoning.

Decision support methods based on symbolic reasoning are flexible, explicable, and can be revised intelligently. Numeric approaches are not. The semantics behind numeric methods are compiled and encoded into efficient but obscure algorithms. It is difficult to employ alternative computations or modify existing ones to fit the semantics of the problem. Explanations supporting decisions, including rejection of alternatives, are difficult to generate and usually not satisfactory. The traditional numeric approach is weakest in supporting revision of decisions and conclusions. But it is our hypothesis that *revision is a fundamental paradigm* for decision making. This is especially true for complex planning.

Planning and Decision Making is a new project which will build up the experience gained with defeasible reasoning in the Proteus project. The goal is to build complex tools which assist in the design of plans by allowing incremental development and revision of plans. Such systems will be "logical spreadsheets" that allow a user to minimally revise a plan given new data or hypothetical situation changes. The system will also allow users to interactively construct plans using simplifying assumptions and to support reasoned retraction of such assumptions when constraints are violated. Finally, the system should provide explanations for the current plan state.

It is unlikely that we will gain the requisite insights into understanding drafting and revising plans by armchair contemplation or by building toy problems. Our approach is to design a model of planning by iterative revision based on reasonably complex applications. We will work on a series of increasingly difficult planning systems. By intimately understanding each system, we will be able to derive the requisite insight to improve our model for use on the succeeding problem.

3. Human Interface Laboratory

Our work in the HI Lab is largely motivated by the belief that interfaces of the future will increasingly be

to KBS and will themselves be knowledge–based. This belief is based both on industry trends in KBS development and on the realization that the key to increasing people's productivity lies primarily in making interfaces more collaborative and allowing people to work closer to their conception of the task rather than requiring them to learn details irrelevant to the accomplishment of their real goals. The only way to provide users with this higher–level flexible access and to enable interfaces to be more cooperative and adaptive is to *represent* the user's task, the language of interaction, the application, and the user. This and the expected continued increase in the development of KBS motivates our focus on knowledge–based interfaces.

HITS: Because many powerful interfaces must rely on several interface capabilities, it is important that our various tools for providing these individual capabilities be designed to function in concert to produce a single, integrated, knowledge–based interface. As a result, we are working on the development of HITS (Human Interface Tools), which is an integration of the tools we are building throughout the laboratory. Our work on HITS as an integrated set of tools is intended both to guarantee that the integration of our tools is possible and to provide us a way of experimenting with such a toolset in order to refine our design. We will release versions of HITS each year. Each such release will incorporate both new results in the overall structure of HITS and new results from our work on the individual pieces of HITS. We expect it to evolve over time, providing shareholders with prototypes and demonstrations of concepts and serving us internally as an experimental vehicle for grounding, motivating, and coordinating our scientific and technological efforts.

Although the idea of a coordinated system for building interfaces is by now widely accepted (such systems are often called User Interface Management Systems), HITS is unique in that: it supports the construction of interfaces to KBS, for which it is often not possible to design unambiguous, humanly learnable input and output languages; the component tools of HITS are themselves knowledge–based; HITS fully supports the construction of integrated multimedia interfaces.

The various components of HITS are discussed in the following sections.

Graphical Interfaces: The approach we are taking in graphics is exemplified in our work on Pogo, a declarative representation system for graphics. We expect that future graphics systems will consist of two components: high–level declarative graphical descriptions and hardware specific interpreters of those descriptions. Separation into these two components has many advantages. Chief among them are the increased portability of code, the speed that will come from an increasing realization of specific interpreters in hardware and the ability to make use of specialized computational hardware, and the facility to efficiently provide views on multiple displays and form multiple conceptual perspectives.

Our plan, in keeping with our overall knowledge–based approach, is to work on the higher semantic levels of graphical representation and try to build on top of existing efforts for the lower levels of graphical representation. This has led us to focus our graphics work in three areas: high–level tools for graphical interface development, representation of graphical and design knowledge, and exploration of a novel interactive work surface interface paradigm.

The major problems that are being attacked are how to provide users with natural methods for specifying behaviors for new dynamic icons, how to have graphical editors automatically represent substantial portions of the knowledge needed to enable integrated multimodal interfaces and to make it possible for the tools to critique interfaces being constructed in terms of graphic design principles, and how to support paper and pencil kinds of interactions on an interactive worksurface.

Recent accomplishments include implementation of a number of experimental graphics tools such as the Pogo representation system and an editor for constructing "dynamic icons", i.e. their form and behavior. Excellent progress has been made on the Interactive Worksurface, the software and hardware for a system with a flat stylus – sensitive display that uses neural net technology to recognize sketches or annotations that are hand–drawn on the surface. Experimental versions of the IWS are now demonstrable.

Natural Language Interfaces: Our approach to the problem of providing natural language interfaces is to design, build, and evaluate a series of natural language understanding and generation programs that can be incorporated into HITS. Our goal is to produce systems that can be effective as components of complete interfaces even though the general problem of natural language use in unconstrained environments will likely remain intractable for decades.

There are three key problems that exist with current natural language systems and to which we are trying to find solutions in our work. The first such problem is that the linguistic knowledge in most existing systems is not portable and must be reconstructed for each natural language interface. Our work is attempting to represent this knowledge in a portable way. Rather than trying to approximate this knowledge in *ad hoc* structures, as is often done in natural language interfaces, we are attempting to

ensure portability by exploiting a linguistically sound theory of morphology and grammar.

A second problem is that rules that translate a natural language sentence into structures interpretable by application programs must be reconstructed for each interface. Although there is no way to avoid this reconstruction entirely, two approaches are being pursued to minimize this effort. The first is to exploit as much as possible the knowledge in the application program itself. Secondly, we are providing powerful tools so that the interface specific knowledge can be built as efficiently as possible.

A third problem with existing natural language interfaces is that they are often superficial in the sense that they do not build on detailed knowledge either about the task that is being performed or about the dialogue that is taking place. As a result, many sentences cannot be interpreted at all and many others are interpreted incorrectly. Our work is attempting to improve this situation by tying the natural language system more closely into the knowledge base of the application program and by providing a deeper analysis of the entire discourse as more than just a sequence of sentences.

The project has produced a series (Lucy) of increasingly sophisticated natural language understanding systems together with an editor (Luke) which can be used to associate linguistic information with domain objects in an application knowledge base.

Intelligent User Assistance: Although an important goal of the HI Laboratory's overall efforts is the design and implementation of interfaces that make the correct use of an application program as obvious to users as possible, in the foreseeable future we will not be able to build interfaces that make the complete functionality of the application and its interface immediately knowable. The goal of the Intelligent User Assistance (IUA) project is to support the development of advising and coaching systems by providing three key capabilities: a generalized architecture for advising and coaching; a knowledge base of advising and coaching strategies that can be exploited within the generalized architecture; and a set of tools for building the application-specific knowledge that each instance of the generalized architecture must also exploit.

Having conducted empirical studies of people performing tasks and of advisors helping them to do so, we have begun developing a series of prototype advisors for specific domains and to construct a set of tools that are useful in doing so. We have in mind a sequence of systems that range from intellectual amplifiers that themselves do relatively little problem solving (but that effectively augment the problem-solving abilities of their users) to intelligent assistants that can be given high-level problem descriptions and drive the collaboration required to reach a solution to that problem. There are several more specific dimensions that form the basis of this evolution. One is the extent to which it is necessary to incorporate a general purpose planning system. We are getting promising results from a restricted planning system in which solutions to individual problems are generated by referring to a hierarchical model of the way that experts solve problems in this domain.

Another dimension along which we can increase the power of our systems is the flexibility of the interface between the user and the advisor. One powerful idea is that of the *advice object*. Each advice object corresponds to an entity in the interface, such as the screen objects representing the application's data structures or procedures, or a piece of advice. Associated with each such object is a set of strategies and knowledge structures that can be accessed by the user by clicking on the object in the interface.

A third dimension is the locus of control. Control may reside entirely with the user who must ask specific questions to get advice. Alternatively, control may reside with the system, which may be able to volunteer advice under appropriate circumstances. We are exploring both of these approaches.

Last year we transferred the initial version of an experimental Interactive Development Environment for Advising (IDEA). IDEA allows developers to implement domain-independent and domain-specific advisory strategies which can be invoked either by the user on request, or by the system itself when it detects a situation that calls for advising. At run-time IDEA maintains a history of user interactions with the system at various levels of abstraction. It is able to compare this trace with an ideal model of the use of the application (as provided at development time by an expert) to produce efficient, cognitively appropriate advice.

4. Systems Technology Laboratory

The mission of the Systems Technology Laboratory is to develop system architectures and associated technologies that are commensurate with projected improvements in the functionality and usability of future computer applications. Within the broad charter of investigating high performance, high capacity platforms for symbolic computing, we are:

- covering both conventional (sequential) and parallel execution environments
- paying particular attention to the problem of providing very high speed access to, and manipulation of, very large knowledgebases.

Both the computational and data management demands of symbolic applications must be successfully addressed.

Logic–based Programming Languages: In the next two decades, the market for symbolic applications will experience a tremendous growth largely as a result of knowledge–based applications and expert system applications becoming widespread in the business world. Since these new applications will be based upon and extend the functionality of existing management information systems, there will be an acute need for programming languages and systems that are effective in both the domain of knowledge–based applications and in that of the more traditional applications, such as database management and retrieval.

The Logic Data Language (LDL) is designed to amalgamate the functionalities and enabling technologies of relational databases with the general purpose symbolic application development capabilities of logic programming. Thus, the LDL system supports rule based programming, pattern matching and inferencing, as in Prolog, along with the transaction management, recovery, integrity and schema based data definition facilities of relational databases. We are pursuing two experimental implementations of LDL: one is for a highly parallel database machine (Bubba), the other is for a single processor workstation environment. By the end of 1988 we will transfer a version of LDL running on Unix workstations, which will provide a highly portable and efficient demonstration vehicle for LDL. The execution speed of LDL is expected to be competitive with that of current procedural languages. The compiler includes an optimizer that automatically generates efficient execution plans for queries.

LIFE (a Logic of Inheritance, Functions and Equations) tackles the problem of extending logic programming with knowledge representation primitives such as generalization and inheritance. Moreover, it merges key features from functional programming and object oriented programming and also embeds functional and relational constraints and residuation. Performance is obtained by wiring–in the inheritance mechanisms in the unification algorithm. Our experience with LIFE applications suggests that the language represents a powerful tool for the development of ambitious symbolic applications such as CAD expert systems and natural language parsing. Furthermore, the LIFE experiment is teaching us important lessons on how to integrate different declarative languages, and to add knowledge representation primitives to such languages in a clean and efficient fashion. The LIFE interpreter is nearly completed and within a year we will have a compiler.

Future plans center on building upon the experience gained in LDL and LIFE to develop a single advanced language that combines their strengths and in addition can be used by non–programmers through visual programming techniques.

Bubba: Bubba is a highly parallel database machine designed to support a large mix of transactions and query programs of varying complexity all running concurrently against a large database. The market motivation for Bubba is the trend, brought about by relational technology, towards high–level interactive interfaces to operational databases. We believe this trend will lead to the need for large "information servers" providing support for:

- a large volume of transaction classes requiring immediate response and simple update transactions
- complex update transactions, representing a shift from batch processing to interactive processing
- a large number of query programs representing a spectrum of information needs -- from simple requests for stored information to requests that derive new information by applying complex transformations to the stored data (including deductive database management)

The goal of the Bubba project is the design of a dedicated scalable architecture that is dramatically superior in cost/performance compared to a mainframe–based system providing similar functionality in the early 1990s time frame. We envision using Bubba to support a variety of application classes.

The project began in 1984 and is currently in the midst of experimental implementation and modeling activities. Most of the research issues have already been addressed and resolved. The purpose of the implementation and modeling activities is to demonstrate our ideas and validate them with high confidence.

Bubba is undergoing implementation on a commercially available 40–node multiprocessor. We are collaborating with the Languages project to insure that LDL object code will execute efficiently on Bubba, as a demonstration of Bubba's ability to effectively support deductive database management as well as relational and transaction processing demands.

A series of working prototypes will be completed and transferred throughout the coming year. Beyond this the plan is to shift emphasis from parallel database management to parallel architectures for advanced KBS of the variety envisioned by the AI and HI Laboratories.

Orion: The increasing use of object–oriented languages and concepts has exposed the need for

augmenting object-oriented programming and application systems with database capabilities. At a minimum, object-oriented programming systems require objects to be persistent and sharable, so that objects generated during the execution of a program will be accessible to the subsequent invocation of the program, and they may further be accessible to a number of concurrently executing programs. Beyond this, within the application domains to which the object-oriented approach is well-suited (including CAD and AI), a number of complex tasks which application programmers have traditionally had to program should be offloaded to the database system; such tasks include version control, change notification, and long-duration transactions.

There were thus a number of major research problems that had to be solved. The impact of object-oriented concepts on the database system had to be fully understood (and vice-versa). The Orion database project was initiated in 1985 to address these research issues and to develop a database system that was well-matched to the unique requirements of object-oriented systems. A non-distributed version of Orion (Orion-1) was released in May of 1987. Since then we been focused on research, design, and prototyping of the Orion-2 homogeneous distributed object-oriented database system. In Orion-2, each workstation will have a full-function Orion which manages a private database. Further, each workstation Orion will participate in the access and management of a common shared database.

After Orion-2 is transferred to shareholders the emphasis of the project will shift to support for distributed heterogeneous databases, perhaps starting with a bridge between logic-oriented (LDL) systems and object-oriented (Orion) systems.

CODE: In conventional object-oriented languages and systems, objects are 'passive', in that they respond only to messages. However, message passing in these systems assumes a synchronous protocol, i.e., the sender of a message is blocked until receiving a reply from the receiver of the message. In contrast, concurrent objects are 'active' with a high degree of autonomous control, i.e., they have more knowledge and responsibility than passive objects to activate themselves and interact with other objects. Concurrent objects may be activated by any type of event, including messages, timer interrupts, and user-specified trigger conditions. The high degree of autonomy in concurrent objects implies an asynchronous communication among objects, and makes concurrent objects well-suited to modeling concurrent or distributed applications such as the scheduling and simulation in computer-integrated manufacturing. Further, successful execution of such applications on parallel/distributed hardware has the potential for dramatic improvement in performance.

CODE (Concurrent Object-oriented Design Environment) is a new project focused on fully exploiting the concurrent object concept. Among the research goals are:

- formalization of a model of concurrency (communication) which will allow maximum exploitation of parallelism in an application
- unification of a selected model of concurrency with the abstract object model
- augmentation of concurrent object-oriented programming with database support (building on the Orion experience)

Neural Networks: This is a brand new project whose long term goal is to develop a theory and practice of neural computation to enable its widespread use as a computing technology. Activities will include: examination of scaling properties of NN learning algorithms, and development of new learning algorithms; development of the mathematics and language to describe adaptive systems; investigation of the properties of the individual neuron as well as collections of neurons working together in biological systems; incorporation of time as a variable in neural computational models; and investigation of heterogeneous neural network models (i.e. incorporating several problem-solving methods, each specialized for a class of sub-problem).

Optical Computing: MCC got involved in optical computing in a small way a few years ago and we are now considering gearing up a significantly larger effort in this interesting technological domain. The past work was centered on investigating the potential of photorefractive crystals as a medium for mass storage, somewhere between RAM and disk in the storage hierarchy.

Our results have been sufficiently encouraging that we anticipate increasing our investment in this line of research. The proposal includes new work in such topics as optical switching networks and optical neural nets. Some of this work is funded by DARPA.

5. Experimental Systems Laboratory (DARPA-funded)

Rapid advancement of leading edge computing system technology requires a balanced mix of theoretical, analytical and experimental research. It is likely that recent thrusts into high performance parallel computation will require increasing emphasis on experimentation. Indeed, there is a clear worldwide trend toward exploratory prototyping of systems that embody innovations in hardware, system software and application technologies.

The recently launched MCC Experimental System Kit (ES-Kit) project may be viewed as a source of

1989–93 timeframe, related research is expected to evolve in directions that place more emphasis on exploitation of ES–Kits to conduct important experimental research at MCC. One could thus view the near–term work as foundation–building, with a longer term objective being use of ES–Kits to rapidly advance computing systems technology.

Two distinct research directions appear to be likely. The first will be software–oriented, typically involving the emulation of new systems layered onto ES–Kit configurations that in effect provide scalable and unusually configurable high performance parallel host systems that are impractical to obtain as commercial products.

The second major thrust will emphasize the development of experimental VLSI and subsystem hardware incorporating innovations that are impractical in some sense to study via software. The emergence of affordable rapid hardware prototyping technologies will tend to naturally accelerate the evolution of high performance hardware in most technical as well as business sectors of shareholder interest. It is therefore important to be near the leading edge of experimental hardware research.

A basic set of ES–Kit hardware and software building blocks are being developed during 1988–1989. Later research is expected to produce a sequence of more advanced hardware and software prototyping modules, which exploit emerging new technologies offering increased performance, capacity and functionality.

Collaboration with other research groups is expected to promote development of specialized modules and support tools, which are compatible with and further expand the applicable domain of ES–Kits. (Given vigorous government and industrial support, it is possible that within two or three years, the majority of the module types available to ES–Kit users could be developed elsewhere.)

Substantial research and development beyond the scope of the initial ES–Kit contract is expected to be launched in future years. Some of this work will undoubtedly involve task–specific development of modules, tools and related capabilities, under sponsorship of follow–on government contracts. Other experimental systems work producing proprietary technology could be sponsored by MCC shareholders.

6. ACA's Next Steps

Most of the research undertaken by ACA has been motivated by a mission and goals established at the time of MCC's inception. Even as we continue to work towards fulfillment of these goals, it is clear that new research must be motivated by an updated perspective on future competitive pressures. Thus we have found it appropriate to define new long range "beacons" predicated upon a collective vision of how people and institutions will use computers at the turn of the century. The task is to intersect that vision with an assessment of ACA's technical strengths and weaknesses to determine how the research agenda should evolve and how the organization must evolve to meet the challenge.

We are midstream in this exciting, difficult process, starting from the simple notion that information systems of the future – including very large, complex systems – will be knowledge–based. From the vantage point of the computer scientist this may seem to be a mundane prediction by now, but in practice there are myriad technical and organizational problems inhibiting implementation. But if we are moderately optimistic about the march of technology, and if we assume that the practical barriers are eventually overcome, then we can speculate about a powerful spectrum of systems whose end points are characterized by the two research directions shown in Figure 1.

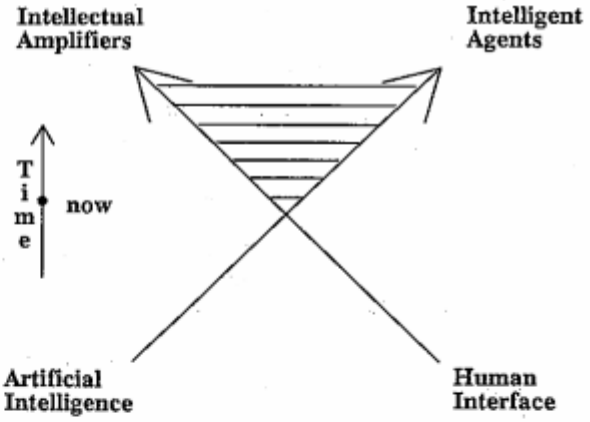

Figure 1: Directions for the Future

The first is towards the development of smarter and smarter systems. This direction has as its ultimate goal the creation of an *intelligent agent* similar to a person. Such an agent would be given a problem to solve. It would then solve it, with only minimal interaction with its user, and report back its result. The second direction is towards the development of *intellectual amplifiers*: systems that allow people to function in smarter ways by augmenting their perceptual, memory retrieval, and reasoning processes. These systems do not solve large problems autonomously. Instead, they interact with their users as the users solve the problems. The first of these directions is supported by the evolving body of research in artificial intelligence; the second by the

work in human interface techniques and cognitive science.

In their extreme forms, these goals are different and some of the technologies required to support them are different. But, as the figure attempts to convey by the shaded region between the two lines, there is actually substantial overlap between these two directions, both in the goals and in the necessary technologies.

Intellectual amplifiers can be more and more helpful as they become able to take on more and more complex tasks. Intelligent agents are only intelligent if they interact with their users when that is the best way to acquire the information that they need. Further, some of the best ways for amplifiers to interact, such as through natural language and image recognition, require substantial intelligence in their implementations. This overlap means that we can, by following a single, broadly based research strategy, provide to our shareholders the necessary technology for the creation of both classes of systems. This technology will be derived both from the continued progression of work in artificial intelligence and in human interface technology, as well as from research into the systems problems that must be solved in order to provide the platform upon which these other techniques can be delivered effectively to users.

We are now in the process of identifying some specific regions in the shaded area of figure 1 for further refinement and (eventually) development. Although we naturally want our research results to be as generically applicable as possible, we are interested in selecting a few particular targets to motivate our research, focus the work, and communicate its potential commercial benefits. The challenge is to choose opportunities that are at once intuitively profound in their market impact, feasible in the 10-year outlook, and yet well beyond the reach of today's technology.

Bibliography

Artificial Intelligence Laboratory

Lenat, D. and E. Feigenbaum, "The Knowledge Principle and the Breadth Hypothesis", *Proceedings IJCAI 87, Foundations of AI Workshop*, MIT.

Lenat, D., "CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks", *AI Magazine*, Vol. VI, No. 4, Winter 1986.

Ballou, N., H. Chou, F. Garza, W. Kim, C. Petrie, D. Russinoff, D. Steiner, and D. Woelk, "Coupling An Expert System Shell with an Object-Oriented Database System", *The Journal of Object-Oriented Programming*, SIGS Publications, Vol. 1, No. 2, June/July 1988.

Huhns, M. and R. Acosta, "Argo: A System for Design by Analogy", *IEEE Conference on AI Applications*, Summer 1988.

Huhns, M., L. Stephens, and D. Lenat, "Cooperation for DAI through Common Sense Knowledge", *Proceedings of the 8th Workshop on Distributed Artificial Intelligence*, May 1988.

Petrie, C., D. Russinoff, and D. Steiner, "PROTEUS: A Default Reasoning Perspective, *Proceedings of Fifth Generation Systems*, National Institute of Software, 1986.

Human Interface Laboratory

Tarleton, P. Nong and M. Tarleton, "POGO: A Declarative Representation for Graphics", to appear in *Object-Oriented Concepts, Applications, and Databases*, W. Kim and F. Lochovsky, editors, Addison-Wesley, 1988.

Avery, J., "Interactive Worksurface: An Interface Paradigm for Sketchable Things", *Telematics Workshop*, May 1988.

Rich, E. and S. Luper-Foy, "An Architecture for Anaphora Resolution", *Proceedings, Second Conference on Applied Natural Language Processing*, February 1988.

Wroblewski, D. and E. Rich, "Luke: An Experiment in the Early Integration of Natural Language Processing", *Proceedings, Second Conference on Applied Natural Language Processing*, February 1988.

McKendree, J. and J. Zaback, "Planning for Advising", *Proceedings of CHI 1988*, May 1988.

Systems Technology Laboratory

Zaniolo, C., "Design and Implementation of a Logic Based Language for Data Intensive Applications", invited paper, *Proceedings, 5th International Conference/Symposium on Logic Programming*, August, 1988

Boral, H., "Parallelism and Data Management", *Proceedings, 3rd International Israeli Conference on Data and Knowledge Bases*, June 1988.

Copeland, G., W. Alexander, E. Boughter and T. Keller, "Data Placement in Bubba", *Proceedings, SIGMOD '88*, June 1988.

Kim, W., N. Ballou, J. Banerjee, H. Chou, J. Garza and D. Woelk, "Integrating an Object-Oriented Programming System with a Database System", To appear in *Proceedings, 3rd Annual Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA)*, September 1988.

Peterson, C. and J. Anderson, "A Mean Field Theory Learning Algorithm for Neural Networks", Published in *Complex Systems I*, 1987.