mail, network designs, and protocols and had kept system tables for network host names and addresses, both local and over the ARPANET, up-to-date.

Because of the heavy loads we have been experiencing on the 2060, we made a detailed performance study of the system dynamics and usage patterns. Among the observations of this study was that the usage of the 2060 was fairly evenly spread between research work (Lisp program development) and text-processing, communications (electronic mail, bulletin boards, etc.), Executive utilities, and system servers (printing, networks, etc.). The research usage has been much higher in the past but has already migrated significantly to personal workstations, while these other applications have not because comparable tools do not exist yet. In this study, we did not find any significant areas of inefficiency in the system -- simply that our user load is very high for the machine resources available.

*UNIX Development and Support*

We run UNIX on our shared VAX 11/780 and on our 11/750 file servers. This system has been used pretty much as distributed by the University of California at Berkeley, except for local network support modifications, such as for ChaosNet protocols. The local VAX user community is small, so we have not expended much system effort beyond staying current with operating system releases and with useful UNIX community developments.

**Workstation System Development and Support**

Lisp workstations represent the major new direction for system development at SUMEX-AIM because these machines offer high performance Lisp engines, large address spaces required for sophisticated AI systems, flexible graphics interfaces for users, state-of-the-art program development and debugging tools, and a modularity that promises to be the vehicle for disseminating AI systems into user environments. Accordingly, we have invested a large part of our system effort in developing selected workstations and the related networking environments for effective use in the SUMEX-AIM community. In the transition to workstations as *computing* environments suitable for AI applications work, not just as programming environments, much system development remains to be done, as illustrated below.

*Filing*

In general, each vendor has addressed the file storage needs of their particular workstation in a way that is incompatible with most other workstations, making support difficult in a highly heterogeneous environment such as the SUMEX-AIM community. The resources necessary to maintain many distinct families of filing conventions and protocols on specialized hardware, all meeting the performance needs of a demanding research community, is prohibitive. Thus we have decided to attempt a compromise.

There is active systems research on distributed file service issues and the results are not clear enough yet to guide long range design decisions. So, we have tentatively decided to adopt a variant of the NFILE file access protocol developed by Symbolics, Inc. A *file access protocol* is intermediate between a *remote file system* and a *file transfer protocol*. A remote file system imposes many constraints upon a potential server machine by specifying features of the file system such as pathname syntax, data block size, character set, protection mechanisms, etc. This makes such protocols very difficult to implement on arbitrary machines as many of these attributes are integrated into operating systems at quite a low level. Conversely, file transfer protocols are specified to allow *copying* an entire file from one machine to another -- a very primitive form of access to the files. A file access protocol can be designed to exist with many

different operating systems, each with its own idiosyncrasies in its file system, but still allowing remote users flexible access to the data stored in the file systems by providing features such as random access, well defined file directory listing, file property manipulation facilities, and asynchronous error recovery.

We decided on NFILE for the following reasons:

- Most significantly, NFILE is built upon abstract interfaces to network streams and host operating systems. It can be easily built upon any reliable byte stream protocol, allowing us to use the results of on-going network development without reimplementing filing protocols. It also is careful not to specify host-dependent information such as pathname syntax or storage format, while providing mechanisms for manipulating file system entities such as directories, files, links, and file attributes. Many file attributes, such as BLOCK-SIZE and CREATION-DATE are included in the protocol, but any can be added as needed, and none are required by the protocol itself, lending flexibility that should make it easy to implement on a variety of operating systems.

- NFILE is a public domain protocol. No licensing is needed to implement or run it.

- An implementation already exists on the Symbolics machine, which can be used for testing and debugging.

- We can implement it to run efficiently in the UNIX kernel, providing the performance the research community needs with inexpensive equipment.

- NFILE can be implemented fairly easily on all of the systems in use by the SUMEX-AIM community since it need not draw on internal operating system features, difficult modifications to existing software should not be needed. Then, as alternative, potentially better techniques become known, NFILE can be abandoned and replaced without having consumed significant resources.

- Many of the options specified by NFILE are derived from the CommonLisp specification and so provide for a significant part of our needs without extension.

*Electronic Mail*

Electronic mail has become a primary means of communication for the widely spread SUMEX-AIM community. The advent of workstations is forcing a significant rethinking of the mechanisms employed to manage such mail. With mainframes, each user tends to receive and processes mail at the computer he uses most of the time, his *primary host*. The first inclination of many users when an independent workstation is placed in front of them is to begin receiving mail at the workstation, and, in fact, many vendors have implemented facilities to do this. However, this approach has several disadvantages:

- Workstations (especially Lisp workstations) have a software design that gives full control of all aspects of the system to the user at the console. As a result, background tasks, like receiving mail, could well be kept from running for long periods of time either because the user is asking to all of the machine's resources, or because, in the course of working, the user has (perhaps accidentally) manipulated the environment in such a way as to prevent mail reception. This could lead to repeated failed delivery attempts by outside agents.

- The hardware failure of a single workstation could keep its user "off the air" for a considerable time since repair of individual workstation units might delayed. Given the growing number of workstations spread throughout office environments, quick repair would not be assured, whereas a centralized mainframe is generally repaired very soon after failure.

- It is more difficult to keep track of mailing addresses when each person is associated with a distinct machine. Consider the difficulty in keeping track of postal addresses or phone numbers if each person you knew lived in a different city. On the other hand, remembering a name and one of several "hosts" is fairly simple, though not perfect.

- It is very difficult to keep a multitude of heterogeneous workstations working properly with complex mailing protocols, making it difficult to move forward as progress is made in electronic communication and as new standards emerge. Each system has to worry about receiving incoming mail, routing and delivering outgoing mail, formatting, storing, and providing for the stability of mailboxes over a variety of possible filing and mailing protocols.

Thus, we are investigating the alternative strategy of having a *mail server* machine which handles *mail transactions*. Because this machine would be isolated from direct user manipulation, it could achieve high software reliability easily, and, as a shared resource, it could achieve high hardware reliability, perhaps through redundancy. The mail server could be used from arbitrary locations, allowing users to be freed from their console to read mail across campus, town, or country without need of expensive machinery.

The mail server acts as an interface among *users*, *data storage*, and *other mailers*. Users employ a *mail access protocol* to retrieve messages, access and change properties of messages, manage mailboxes, and send mail. This protocol should be simple enough to implement on relatively simple, inexpensive machines so that mail can be read remotely easily. This is somewhat distinct from some previous approaches since the mail access protocol is used for all message manipulations, isolating the user from all knowledge of how the data storage is used. This means the the mail server can utilize the data storage in whatever way is most efficient to organize the mail. The data storage could be anything from conventional magnetic disk file system to a highly specialized mail filing system built on optical disks, since it is abstracted from other elements in the mail system. The other mailers constitute the mail server's (and thus the users') link to the outside world. The mail server would use various *mail transport protocols* (e.g., SMTP) to exchange mail with other mail hosts.

We have been investigating user mail interface issues for workstations, as well as issues for the mail access protocol itself. We are examining several related projects, including MIT's PCMAIL, the public parts of Xerox's Grapevine and NSMail, and work on Stanford's V system. We have implemented an interim mail access protocol and have begun implementing user interfaces that make use of it on Xerox D-machines and Texas Instruments Explorers.

*Xerox D-Machines*

Much of the SUMEX-AIM community uses InterLisp and has moved naturally to the Xerox D-machines -- initially the Dolphin (1100), then the Dandelion (1108), Dandetiger (1109), and Dorado (1132), and now the DayBreak (1186). Much work has gone into hardware installation and networking support but we have also developed numerous software packages to help make the machines more effective for users and to ease our own problems in managing the distributed workstation environment.

The number and utility of "lispusers packages" has again increased significantly over the past year. Although too numerous to detail (there are approximately 550 packages currently, up from about 240 last year), packages receiving heavy use for the first time in the past year were Sketch, FileBrowser, TEdit, Manager, Impress, Hash, Helpsys, and Spy. Many of these packages were beta-tested at SUMEX and/or patches and updates were distributed via the Info-1100 and Bug-1100 discussion lists we maintain. Other AIM sites and research groups around the world were able to share in our progress and benefit from our experience by participating in the discussion lists. We, in turn, as subscribers to the same discussion lists were able to benefit from the experiences and expertise of others.

The past year saw Interlisp's device-independent graphics mechanism ("image streams") mature and grow a good deal. We participated in the design changes, ensuring that the specification was sufficiently device-independent that our Impress laser printer protocol package would be integrated into the system as easily and well as the Xerox-authored Press protocol and Interpress protocol packages. An ImageStream driver was developed to support a Hewlett-Packard color plotter. The development of the driver helped to explore the issues of color in the ImageStream specification as well as test how it applied to analog devices. For the first time we were able to generate color hardcopy output from the both the black & white Interlisp workstations and, with some additional conversion software, from the color Iris workstation.

The Impress package was extended to include almost all of the operators in the new specification. The Impress package is sufficiently complete that HARDCOPYW, DISPLAYGRAPH, TEdit, and Sketch all produce output of quality comparable to that of the more-expensive Xerox laser printers. In most cases the output is generated faster than for those printers and more compactly. We are presently participating in another round of improvements to the specification.

In support of the expanding number of ImageStream drivers, a self-scaling graphics command set was implemented on top of the standard graphics command set that allowed software to be written without regard to the scaling requirements of a particular output device. This allowed graphics output to be easily redirected to varying printing devices without modification of the source program and/or without adding additional scaling routines to every program.

Implementation of an Interlisp-based Ethernet boot file server for the Xerox workstations was completed this past year. This server made it possible to obtain workstation installation and diagnostic utilities via the network as an alternative to floppy disks. Much later, the Interlisp-based boot file server was replaced with a Xerox product Ethernet boot server which extended our network installation and diagnostic capabilities. The addition of network boot file service has led to improvements in software installation procedures by allowing us to move away from our previous dependence on floppy disks. This has become increasingly important as the newer Xerox 1186 hardware supports a smaller capacity floppy disk drive and would require use of over a dozen floppy disks if Ethernet installation were not available.

Initial exploration into distributed systems and remote workstation access was started. An experimental XNS-based TELNET server was built to allow access into a workstation remotely via the Ethernet. This experimental server uncovered numerous problems with the workstation software and initiated discussions that led to a complete TELNET/GAP (Xerox XNS Gateway Access Protocol) server for the workstations (pending the next Xerox software release). The workstation "executive" part of the experimental TELNET server was extracted and generalized and put to use in the TCP-based Ethernet virtual graphics work.

We developed a system called IMEDIT. This program allows users to break apart Impress files and also to merge in other Impress files. Merging Impress files is an

important feature since SCRIBE cannot do this. SCRIBE can merge in a picture but any text in the picture will be printed in the wrong font. IMEDIT gets around this problem by manipulating the fonts so that fonts in a merged file don't conflict with fonts in the base file. IMEDIT can also generate an ASCII file showing the Impress commands and their arguments in an Impress file. This feature is invaluable for those who need to understand the Impress language.

Currently we are working on the TEdit text editor, initially to facilitate simple document types like memos. We have implemented an ImageObject that allows users to select the logo they prefer and are working on others for document features like the return address. Eventually users will be able to interactively choose what they want from standard menus. Such systems are essential to allow users to move work from the 2060 to workstations.

We have worked closely with many other sites, including the Center for Study of Language and Information at Stanford, the Stanford Campus Networking group, Rutgers University, Ohio State University, the University of Pittsburgh, Cornell, Maryland, and industrial research groups such as Xerox Palo Alto Research Center, SRI, Teknowledge, IntelliCorp, and Schlumberger-Doll Research. We have been the maintainers for the international electronic mail network of users for research D-machines, which have upwards of 300 readers, and the interchange of ideas and problems among this group has been of great service to all users.

*ZetaLisp Workstations*

The complement of ZetaLisp-based workstations has grown to include twenty Texas Instruments Explorers and ten Symbolics 3600-class machines. The acquisition of these machines was driven by three primary factors. First, many of the research projects are attempting to become independent of any particular machine, and so are moving development to the CommonLisp standard language. These machines are among the first to offer production quality support of CommonLisp. Second, some application systems require substantial performance in terms of processing speed and address space in order to complete in a reasonable amount of time. These machines were among the highest performing Lisp machines available at the time. Finally, there are researchers who prefer the MacLisp/Emacs derived programming environment.

The Explorer and the 3600 are both built on MIT's ZetaLisp software, and so continue to share much functionality. Therefore, many projects have been undertaken simultaneously on both machines. In order to facilitate this interoperability, two compatibility packages have been built, one for each type of machine. The packages contain code to add functionality to each machine to bring it closer to the specification of the other machine where possible, without blocking the native functionality of the system it is running on. There are also lists of features which do not exist and could not be easily duplicated, as well as suggested workarounds where appropriate. These compatibility packages have been made available to the ARPANET community.

We found that users of these machines spent a considerable amount of time redoing work that had already been done since there was no adequate library of user-written tools to draw from. Thus we have undertaken to provide such a facility and to gather as many tools as possible. The *TOOLS* system allows a user to select those tools that he wishes to load either by giving a list of their names (in an initialization file, for instance), or by selecting them from a menu. The menu can also be used to obtain on-line documentation about each tool, and so provides a convenient way to browse the tools. The following is a list of the tools that have so far been implemented or collected:

*FS-TO-FS-BACKUP* -- Functions that can copy unbacked-up files from the Lisp

machine's file system to another file system which is then backed up to tape. This obviates the need to do backups to expensive and slow cartridge tapes.

*SYSTEM-MANAGER* -- Provides for shared access to hierarchically structured sets of files, allowing multiple people to work on development of a single system or subsystem simultaneously.

*DYNAMIC-SYSTEM-MENU* -- Attempts to facilitate managing the screen so that the size and position of windows can be easily tailored to the task at hand.

*NET-IMAGEN* -- Allows printing on network based Imagen printers using the Impress document formatting language. (Symbolics implementation from MIT)

*TCP-FINGER* -- (Needed only on Explorer) Implements the popular FINGER person lookup protocol for TCP/IP.

*WHO-LINE* -- (Explorer only) Shows percentage-wise progress through editor buffers during lengthy operations such as compilation.

*VERTICALLY-ORDERED-MENU-ITEMS* -- Allows multi-column menus to be displayed with the items split into columns first rather than rows first.

*SMALL-FONTS* -- Changes all standard windows to use a smaller font, allowing more data to be displayed at a time.

*SCREEN-EDIT-MIXIN* -- Allows selected windows to be moved or reshaped by clicking on small boxes in the margins of the windows.

*MOUSE-SELECTABLE-PANE-MIXIN* -- Allows constraint frame panes to be mouse selectable.

*MAKE-INTO-SCRIBE-FILE* -- Converts a file with Lisp machine special characters into Scribe format so that the special characters will be correctly printed on Imagen printers.

*INSPECT-HASH-TABLES* -- (Needed on Explorer only) Causes the inspector to display hashing data structures in a more readable Key/Value format.

*GENERAL-NAMED-STRUCTURE-MESSAGE-HANDLER* -- Causes selected structures to error instead of returning NIL when they receive messages they do not handle, facilitating debugging.

*FILTER-WINDOW-DEBUGGER* -- Allows specifying functions that will not be displayed in the window debugger, eliminating the clutter of system functions so that user is only presented with "interesting" stack frames.

*DEFSTRUCT-TYPE-CHECKING* -- An addition to DEFSTRUCT that causes the access functions of selected structures to check their arguments, facilitating debugging.

*DEBUG-STACK-GROUP* -- (Explorer only) Allows users entering the debugger to examine a particular stack group from the window debugger.

*BATCH-PROCESSOR* -- Facility for "running" command files overnight.

*CHOOSE-VARIABLE-VALUES-MACROS* -- Alternate interface to the CHOOSE-

VARIABLE-VALUES facility which does not require the user to specify and manipulate specials.

All of these tools except for parts of NET-IMAGEN were developed at Stanford. We are currently working with the Lisp machine vendors on licensing that will make it possible to distribute these tools via the ARPANET. (Both Symbolics and TI are much more restrictive in their software-sharing policies than is Xerox).

A great deal of work has been done in installing the Explorers in the SUMEX-AIM research environment. This is one of the first installations of a large number of Explorers, and we have participated actively in the "shaking down" of the Explorer, by being a beta test site for release 2 of the Explorer system software, and release 1 of the Explorer TCP/IP software. In the course of the testing, we submitted 56 written software problem reports, and over 25 verbal reports, many of which included solutions. As a result, the Explorer is now a well-integrated part of the research environment, allowing many researchers to actively pursue their work by putting powerful development tools on their desks.

*Virtual Workstation Graphics*

We have done a number of experiments with the remote connection of bitmapped displays to hosts and workstations. Generally, the displays on Lisp machines are tethered through a high bandwidth cable to their processors. This limits the flexibility with which users can move from one Lisp machine to another (one must move physically to another machine) and loses the ability of researchers to work from home over telephone lines. A way of providing a more flexible display to processor connection is to use a virtual graphics protocol, such as the V Kernel system developed by Lantz [4]. This allows efficient communication of the contents of a display window to be compactly represented, transmitted over a communication network, and reconstructed on a remote bitmapped screen.

In order to more fully understand the integration of remote virtual graphics access to workstation, a nearly complete implementation of a client interface of the Lantz Virtual Graphics Protocol (VGP) was done within the Xerox Interlisp-D environment. This implementation was done in such a way as to make the fact that the VGP was in the system transparent to the Interlisp programmer. Several key steps were involved:

- Since the access to the workstation was to be done remotely on some kind of network, it was necessary to write an IP/TCP/TELNET server which handled the peculiarities of the TELNET protocol, and provided the usual input and output data streams to a virtual graphics stream executive. This executive then did remote login authentification and called the remote LISP evaluator.

- In parallel to the LISP evaluation, a mechanism was necessary to interface the client VGP into Interlisp-D. This was done using the IMAGEOP objects which are associated with each stream within Interlisp-D. As a consequence all reads and writes on the "standard" input and output streams were defaulted to the VGP input and output streams when the workstation was accessed remotely in this manner.

- As a consequence when one is connected to such a workstation through a VGP server, the graphics engine was driven by standard calls to graphics functions on the Interlisp-D workstation. Thus, the functionality of windows, menus, text within windows, mouse interaction, and a suite of drawing functions were all translated to the VGP and done remotely on the user's workstation.

This feasibility experiment proved that remote access to a LISP workstation using virtual graphics protocols was practical. This paves the way for additional work to allow researchers to take advantage of powerful user/graphics environments on Lisp machines, even if not physically near the machine.

## Network Services

A highly important aspect of the SUMEX system is effective communication within our growing distributed computing environment and with remote users. In addition to the economic arguments for terminal access, networking offers other advantages for shared computing. These include improved inter-user communications, more effective software sharing, uniform user access to multiple machines and special purpose resources, convenient file transfers, more effective backup, and co-processing between remote machines. Networks are crucial for maintaining the collaborative scientific and software contacts within the SUMEX-AIM community.

### *Remote Networks*

In addition to continuing our connection to TYMNET, we have implemented an experimental connection to UNINET this past year in an attempt to improve services for remote users. As reported last time, we have had serious difficulties getting needed service from TYMNET for debugging network problems and users away from major cities have problems with echo response times. The opinions of about 15 of our heaviest TYMNET users were sought concerning the performance of TYMNET. Though many were quite pleased, several with experience on a variety of other such networks recommended a change. The TYMNET hardware interface itself has been quite dependable in the past year.

Discussions were held with CompuServe and UNINET concerning alternative service. The UNINET connection was finally installed after a period of considerable review, based on technical evaluations, cost analyses, and the experience of other network customers with similar systems (e.g., BIONET). Requests for office phone locations went out to 70 of our TYMNET users. Responses from 50 of them revealed only one who would be unable to reach a UNINET node with a local phone call. The capability of KERMIT data transfers was reviewed as was the capability for using the text editors, EMACS and TVEDIT. On the whole, UNINET seems much more responsive for current users, although evaluation is still underway. Both TYMNET and UNINET services are purchased jointly with the Rutgers Computers in Biomedicine resource to maximize our volume usage price break.

We also continue our extremely advantageous connection to the Department of Defense's ARPANET, managed by the Defense Communications Agency (DCA). This connection has been possible because of the long-standing basic research effort in AI within the Knowledge Systems Laboratory that is funded by DARPA. ARPANET is the primary link between SUMEX and other machine resources such as Rutgers-AIM and the large AI computer science community supported by DARPA. We are also attempting to establish a link to the DARPA wideband satellite network to facilitate the rapid transfer of large amounts of data such as are involved with projects like our Concurrent Symbolic Computing Architectures project.

### *Local Area Networks*

For many years now, we have been developing our local area networking systems to enhance the facilities available to researchers. Much of this work has centered on the effective integration of distributed computing resources in the form of mainframes, workstations, and servers. Network gateways and terminal interface processors (TIP's)

were developed and extended to link our environment together and are now the standard system used in the campus-wide Stanford University network. We are developing gateways to interface other equipment as needed too. A diagram of our local area network system is shown in Figure 6 and the following summarizes our LAN-related development work.

*Ethernet Gateways* -- In our heterogeneous network environment, in order to provide workstation access to file servers, mail servers, and other computers within the network, it is necessary to able to route multiple networking protocols through the network gateways. Over the past year, support for both the Xerox NS and Symbolics/Texas-Instrument CHAOSNET protocols were added to the SUMEX gateways. This support not only provides the routers necessary to move such packets within this topology, but also other miscellaneous services such as time, name/address lookup, host statistics, address resolution, and routing table broadcast and query information. As a consequence, the SUMEX gateways now support these protocols as well as the PUP, and IP protocols. These services are unique within the SUMEX-AIM portion of the Stanford University network, and give our researchers a networking environment that is flexible, of high bandwidth, and extremely dependable.

*Remote Ethernet* -- Some preliminary design was done on a "home Ethernet connection" to facilitate virtual graphics access and other network connections from home workstations. The feasibility of this device was investigated to evaluate its cost versus a similar device manufactured by Bridge Systems at a cost of $5,000.00. It is believed that a less expensive device can be built that will conform to our remote Ethernet needs. Although this device will communicate via modems and hence be much slower than the 10 MBit/sec Ethernet bandwidth, the fact that each remote station will act as an Ethernet host without the RS-232 overhead, should improve file transfer significantly.

*Network Bootstrap* -- Over the past year, SUMEX has participated in the definition of a remote workstation bootstrap protocol which can flexibly load systems over networks, even through gateway links to remote servers. The details of the protocol are documented in RFC951, put out by the ARPANET development group. Implementation of the BOOTP protocols required developing a new programmable read-only memory (PROM) monitor for our workstations (MC 68000-based) with a more extensive command structure to facilitate specification of remote boot file pathnames. If the user specifies enough information (server address, workstation address, and file name), then the PROM bypasses the BOOTP phase entirely and directly enters the transparent FTP phase. This can be useful for manually booting from arbitrary internet hosts not running BOOTP servers. The PROM code currently contains drivers for the 3COM 3C400 interface (at 4 possible multibus board addresses) and the Interlan NI3210 (also at 4 addresses).

The BOOTP/TFTP bootstrap uses a global structure located at the end of memory during its operation. This structure is left intact after the booted program gets control. In some cases a program (such as an EtherTIP) may want to fetch a configuration file listing its addresses and options before starting up. With the mechanism provided by this structure, that program can call the PROM resident TFTP code to fetch the desired configuration file.

Network gateway modifications necessary to route BOOTP requests have been made and installed in all Stanford gateways.

**Laser Printing Services**

Since the first Xerox laser printers were developed in the mid-1970's, a number of companies have produced computer-driven systems, such as Imagen and Adobe. These systems have become essential components of the work environment of the SUMEX-AIM community with applications ranging from scientific publications to hardcopy graphics output for ONCOCIN chemotherapy protocol patient charts. We have done much systems work to integrate laser printers into the SUMEX network environment so they would be routinely accessible from hosts and workstations alike.

Over the past year, we purchased 2 new Imagen 12/300's, upgraded an 8/300 to a 12/300, and converted an old Hewlett-Packard 2688A to a 12/300 laser printer for the SUMEX-AIM community. These enhancements were funded by DARPA. The move to 12/300's was motivated primarily by the ruggedness of the Ricoh LP-4120 print engine used in those printers. Whereas the Canon LBP-CX print engine used in the 8/300 has an expected lifetime of 70,000 pages, the Ricoh LP-4120 has an expected lifetime of 700,000 pages. Since the KSL printed roughly 250,000 pages on laser printers last year we decided it was time to move to a sturdier printing workhorse. Other beneficial side-effects of the upgrade were: (1) higher print rate (12 pages-per-minute), (2) bigger paper tray (half a ream), (3) blacker and more solid print, (4) crisper print, and (5) cheaper supplies (half the price per page compared to the 8/300).

We have also acquired an Apple Laser Writer which interprets the PostScript page description language. Within a few months of its introduction, the Apple Laser Writer has become the most common laser printer on campus and around the world. Economies of scale have made it possible for us to acquire this printer for under $4000. SUMEX AppleNet/Ethernet expertise will make it possible for us to attach the Laser Writer to the high-bandwidth campus internet and operate the printer at the high-end of its 8 page-per-minute capacity. (The vast majority of laboratory-owned Laser Writers in the U.S. are driven over a low-bandwidth RS-232 line yielding only 3 pages-per-minute throughput and typically greater latency.) The PostScript page description language is already the standard of choice at university and DARPA sites (judging by traffic on the Laser-Lovers discussion group). It is generally agreed upon in these communities that PostScript is among the easiest-to-generate and most expressive of the page description languages in use today and reconciles these traits much more effectively than other languages do.

Although anyone with $4000 can benefit from the advantages of owning a Laser Writer, SUMEX users at Stanford have access to a Linotronics 300P typesetter owned by the university. This printer interprets PostScript files identical to those which can be printed on a Laser Writer, but renders its output on photographic paper up to 11" x 17" in size at a resolution of 1200 scans-per-inch. (At present, most of our printers image at 300 spi and our finest printer is the aging Xerox Alto-Raven which images at 384 spi.) To exploit the special capabilities of this printer and to take advantage of the economical Apple Laser Writer, we have begun an Interlisp implementation of an "image stream" driver for PostScript. Unilogic has already added Postscript support to Scribe and Adobe has implemented Postscript support for TeX.

**General User Software**

We have continued to assemble (develop where necessary) and maintain a broad range of user support software. These include such tools as language systems, statistics packages, vendor-supplied programs, text editors, text search programs, file space management programs, graphics support, a batch program execution monitor, text formatting and justification assistance, magnetic tape conversion aids, and user information/help assistance programs.

A particularly important area of user software for our community effort is a set of tools for inter-user communications. We have built up a group of programs to facilitate many aspects of communications including interpersonal electronic mail, a "bulletin board" system for various special interest groups to bridge the gap between private mail and formal system documents, and tools for terminal connections and file transfers between SUMEX and various external hosts. Examples of work on these sorts of programs have already been mentioned in earlier sections on operating systems and networking.

At SUMEX-AIM we are committed to importing rather than reinventing software where possible. As noted above, a number of the packages we have brought up are from outside groups. Many avenues exist for sharing between the system staff, various user projects, other facilities, and vendors. The availability of fast and convenient communication facilities coupling communities of computer facilities has made possible effective intergroup cooperation and decentralized maintenance of software packages. The many operating system and system software interest groups (e.g., TOPS-20, UNIX, D-Machines, network protocols, etc.) that have grown up by means of the ARPANET have been a good model for this kind of exchange. The other major advantage is that as a by-product of the constant communication about particular software, personal connections between staff members of the various sites develop. These connections serve to pass general information about software tools and to encourage the exchange of ideas among the sites and even vendors as appropriate to our research mission. We continue to import significant amounts of system software from other ARPANET sites, reciprocating with our own local developments. Interactions have included mutual backup support, experience with various hardware configurations, experience with new types of computers and operating systems, designs for local networks, operating system enhancements, utility or language software, and user project collaborations. We have assisted groups that have interacted with SUMEX user projects get access to software available in our community (for more details, see the section on Dissemination on page 89).

**Operations and Support**

The diverse computing environment that SUMEX-AIM provides requires a significant effort at operations and support to keep the resource responsive to community project needs. This includes the planning and management of physical facilities such as machine rooms and communications, system operations routine to backup and retrieve user files in a timely manner, and user support for communications, systems, and software advice. Of course, the move of our groups to new space in the Medical School Office Building has required major planning and care to ensure minimum downtime for our computing environment and much systems and electronics work to outfit the new space.

We use students for much of our operations and related systems programming work. We spend significant time on new product review and evaluation such as Lisp workstations, terminals, communications equipment, network equipment, microprocessor systems, mainframe developments, and peripheral equipment. We also pay close attention to available video production and projection equipment, which has proved so useful in our dissemination efforts involving video tapes of our work.

# III.A.3.4. Core AI Research

We have maintained a strong core AI research effort in the SUMEX-AIM resource aimed at developing information resources, basic AI research, and tools of general interest to the SUMEX-AIM community. It should be noted that the SUMEX resource grant from NIH supports much of the computing environment for this core AI work[1] but NIH supports only a small part of the manpower and other support for core AI. Substantial additional support for the personnel costs of our core AI research (roughly comparable to the NIH investment in computing resources) comes from DARPA, ONR, NSF, NASA, and several industrial basic research contracts to the Knowledge Systems Laboratory.

The following summary reports progress on the basic or core research activities within the KSL. The development of the ONCOCIN system (under Professor Shortliffe) is an important part of our core research proposal for the renewal period. Progress on that work is reported separately in Section IV.A.3, however, because its efforts have been supported as a collaborative and resource-related research project up until now. Together, this work explores a broad range of basic research ideas in many application settings, all of which contribute in the long term to improved knowledge based systems in biomedicine.

**Rationale**

Our core AI research work has long been the mainstay on which our extensive list of applications projects are based. Medical information -- both medical data and medical knowledge -- is the key to progress in research and excellence in biomedical science and clinical practice. As the rapid explosion of information continues, clinicians and biomedical scientists must turn to computers for help in managing the information, and in applying it to complex situations.

Artificial Intelligence (AI) methods are particularly appropriate for aiding in the management and application of knowledge because they apply to information represented symbolically, as well as numerically, and to reasoning with judgmental rules as well as logical ones. They have been focused on medical and biological problems for over two decades with considerable success. This is because, of all the computing methods known, AI methods are the only ones that deal explicitly with symbolic information and problem solving and with knowledge that is heuristic (experiential) as well as factual.

Expert systems are one important class of applications of AI to complex problems -- in medicine, science, engineering, and elsewhere. An expert system is one whose performance level rivals that of a human expert because it has extensive domain knowledge (usually derived from a human expert); it can reason about its knowledge to solve difficult problems in the domain; it can explain its line of reasoning much as a human expert can; and it is flexible enough to incorporate new knowledge without reprogramming. Expert Systems draw on the current stock of ideas in AI, for example, about representing and using knowledge. They are adequate for capturing problem-solving expertise for many bounded problem areas. Numerous high-performance, expert systems have resulted from this work in such diverse fields as analytical chemistry, medical diagnosis, cancer chemotherapy management, VLSI design, machine fault diagnosis, and molecular biology. Some of these programs rival human experts in solving problems in particular domains and some are being adapted for commercial use.

---

[1] DARPA funds have also helped substantially in upgrading our mainframe systems and in the purchase of community Lisp workstations

Other core research projects have developed generalized software tools for representing and utilizing knowledge (e.g., EMYCIN, UNITS, AGE, MRS, GLISP) as well as comprehensive publications such as the three-volume *Handbook of Artificial Intelligence* and books summarizing lessons learned in the DENDRAL and MYCIN research projects.

There is considerable power in the current stock of techniques, as exemplified by the rate of transfer of ideas from the research laboratory to commercial practice. But it is also clear that today's technology needs to be augmented to deal with the complexity of medical information processing.

Our core research goals, as outlined in the next section, are to analyze the limitations of current techniques and to investigate the nature of methods for overcoming them. Long-term success of computer-based aids in medicine and biology depend on improving the programming methods available for representing and using domain knowledge. That knowledge is inherently complex: it contains mixtures of symbolic and numeric facts and relations, many of them uncertain; it contains knowledge at different levels of abstraction and in seemingly inconsistent frameworks; and it links examples and exception clauses with rules of thumb as well as with theoretical principles. Current techniques have been successful only insofar as they severely limit this complexity. As the applications become more far-reaching, computer programs will have to deal more effectively with richer expressions and much more voluminous amounts of knowledge.

**Highlights of Progress**

In the last year, research has progressed on several fundamental issues of AI. As in the past, our research methodology is experimental; we believe it is most fruitful at this stage of AI research to raise questions, examine issues, and test hypotheses in the context of specific problems such as management of patients with Hodgkin's disease. Thus, within the KSL we build systems that implement our ideas for answering (or shedding some light on) fundamental questions; we experiment with those systems to determine the strengths and limits of the ideas; we redesign and test more; we attempt to generalize the ideas from the domain of implementation to other domains; and we publish details of the experiments. Many of these specific problem domains are medical or biological. In this way we believe the KSL has made substantial contributions to core research problems of interest not just to the AIM community but to AI in general.

In addition to the technical reports listed below, the following survey articles were published during this year. These are of central interest to AI researchers and of direct relevance to the mission of the SUMEX-AIM resource.

*SURVEY ARTICLES:* KSL-85-19, KSL-85-27, KSL-85-28, KSL-85-37, KSL-85-54, KSL-86-17, KSL-86-32

Progress is reported below under each of the major topics of our work. Citations are to KSL technical reports listed in the publications section.

1. *Knowledge representation*: How can the knowledge necessary for complex problem solving be represented for its most effective use in automatic inference processes? Often, the knowledge obtained from experts is heuristic knowledge, gained from many years of experience. How can this knowledge, with its inherent vagueness and uncertainty, be represented and applied?

   Work continues on BB1, with its explicit representation of control knowledge, as reported last year. In addition, part of our research on

NEOMYCIN is focused on using a flexible, rich representation of control knowledge so that we can model problem solving at the strategic level as well as at the tactical level.

[See KSL technical memos KSL-85-16, KSL-85-17, KSL-85-31, KSL-86-11, KSL-86-27.]

2. *Blackboard Architectures and Control:* How can we design flexible control structures for powerful problem solving programs?

   We have continued to develop the BB1 blackboard architecture for systems that reason about -- control, explain, and learn about -- their own actions. We have developed domain-independent control knowledge sources for refining abstract control plans. We have developed capabilities for explaining problem-solving actions by incrementally elaborating the control plan underlying the decisions to perform them. We have developed capabilities for acquiring new control knowledge from domain experts automatically.

   Our most innovative work on BB1 focused on the idea that reasoning effectively about action requires knowledge about action. In particular, it requires knowledge of: the hierarchy of action types; the patterns of formal parameters defining all actions types; the network of concepts for instantiating formal parameters; the modifiers that can restrict the scopes of defined concept types; the translations of terminal action patterns into executable code; and the partial matches between patterns defining different action types. We developed a body of such knowledge (the ACCORD framework discussed below) for the actions involved in assembling arrangements of objects under constraints. We used the PROTEAN system, which is implemented in BB1, to demonstrate the power of task-specific action knowledge to enhance control, explanation, and learning capabilities. We have also begun to investigate the applicability of this knowledge to another design problem, site layout.

   In addition to the two applications mentioned above, several other scientists at Stanford and at other research and industrial laboratories have begun developing application systems in BB1.

   [See KSL technical memos KSL 84-16, KSL 85-2, KSL 85-35, KSL 86-38.]

3. *Advanced Architectures*: What kinds of software tools and system architectures can provide orders of magnitude speedup in the performance of expert systems? The Advanced Architectures Project is a long-range project with two related goals:

   - To realize a new generation of software system architectures using parallelism to achieve high-speed computation in artificial intelligence applications.
   - To specify multiprocessor hardware system architectures that support those parallel computations.

   The basic problem we are addressing is to increase the speed of execution of expert systems through the use of parallel computations on a multiprocessor computer system. Part of the effectiveness of expert systems, particularly for real-time applications such as continuous signal data understanding, lies in the speed of execution, or throughput rate. However, for many

significant applications of this type, projected performance limits of uniprocessors fall short of the speed required by as much as several orders of magnitude. Multiprocessor parallel computing must be used to attain the necessary levels of performance.

The anticipated computational requirements for the next decade cannot be realized by just using parallelism at only one particular level of computation (for example, parallel left-hand-side rule matching in rule-based systems). To understand the effectiveness of parallel implementations of expert systems, we must study both the programming problems and the performance issues at all levels of the computational hierarchy:

- The application level.
- The problem-solving framework level.
- The programming language level.
- The hardware system architecture level.

Our research emphasis is therefore on overall software and hardware system architectures for the parallel execution of expert systems.

During the past year, with principal support from DARPA under the Strategic Computing Program, we have demonstrated significant progress at each of the levels. We have also completed the first of a series of "vertical slice" experiments, in which a choice is made at each design level and a simulated execution of the resulting system is analyzed.

Application level

The methodology employed in this project is to select an application and use it as the driver that determines the requirements at the underlying system design levels. That is, the application, or class of applications, should determine the architecture rather than the other way around. Consequently, it is necessary to choose applications very carefully with respect to their complexity, generality and potential for significant speedup.

During the past year we defined and started the development of a new application, within the area of signal understanding, information fusion and situation assessment. The new application, called AIRTRAC, concerns the classification and tracking of light aircraft, some of whom are deliberately trying to evade detection (e.g., smugglers). The sensor data include both acoustic and radar data from distributed sources. Other data include flight plans and intelligence reports. This application has many desirable characteristics, including:

- Multiple sources of input, including both "low-level" (radar, acoustic) and "high-level" (intelligence reports) data;
- Need for both data-driven and model-driven (e.g., using knowledge of intentions or expectations of future behavior) reasoning.

Problem-solving framework level

We have completed the first-pass development of two parallel blackboard framework systems, CAGE and POLIGON. A third framework, CAOS, is complete and has been used for the first vertical slice experiment.

CAGE is an extension of the AGE system that contains concurrency primitives for building parallel constructs. In contrast with POLIGON, CAGE represents a conservative, incremental approach to building parallel systems. CAGE will run both on QLAMBDA and CAOS/CARE simulators, or can be run serially. POLIGON is a demon-driven blackboard system in which all blackboard nodes are active agents. Changes in the blackboard nodes trigger rules to be fired. POLIGON will run on the CAOS/CARE simulator. CAOS ("Concurrent Asynchronous Object System") is a set of language extensions to Lisp for multiprocessor systems. CAOS allows the user to express process and data locality and interprocess communication, organized in an object-oriented manner.

Programming Language Level

We have experimented with Lisp-based languages. One is QLAMBDA (renamed QLISP), an extension to Lisp for a multi-processor, shared memory architecture. (See R. P. Gabriel and J. McCarthy, "Queue-based Multi-processing Lisp," in *Proc. of the 1984 Symposium on Lisp and Functional Programming*, August 1984.) We have also designed and partially implemented a concurrent Lisp for the CARE distributed-memory family of multiprocessor architectures. CAREL (CARE Lisp) is a distributed-memory variant of QLISP. CAREL supports features (like MultipLisp), truly parallel LET binding (like QLISP), active objects with locality and state (like OIL), programmer or automatic specification of locality of computations (like para-functional programming or Flat Concurrent Prolog), and both static assignment of process to processor and dynamic spread of recursive computations through the network via remote function call (like V).

Hardware System Architecture Level

Our activity at this level has been focused on the testing and refinement of CARE, a set of executable system component specifications that can be used to specify a parameterized family of hardware system architectures. The architecture consists of a number of sites interconnected under some specified communication topology. Each site consists of (1) an evaluator (of Lisp forms), (2) an operator that performs message handling, process scheduling, process creation and process synchronization, (3) network ports for message routing, (4) FIFO queues that tie these components together, and (5) a memory -- hence a family of distributed memory machines with a processor of parameterized capability at each node. By specializing nodes to consist only of subsets of the above components, shared memory systems can also be simulated. CARE also permits specification of a suite of instrument probes and display panels that can be used to monitor the simulation of the hardware system.

Vertical Slice Experiment

We completed the first set of comprehensive experiments in implementing and executing knowledge-based expert systems on multiprocessor machines. This experiment consists of running the ELINT application, written in ZetaLisp on CAOS running on the CARE simulator. ELINT is a prototype passive radar signal understanding system that was originally implemented in AGE.

The experiment has two objectives. The first is to investigate the quality of solution and the amount of communication as a function of various degrees of inter-process control. In particular, we are investigating the types and amounts of serialization required to assure an acceptable level of solution quality for ELINT. The second investigates overall execution speedup (or "speeddown") as a function of the number of processors. In particular, a version of ELINT with control adequate to assure satisfactory solution quality was run on simulated CARE arrays ranging in size from four to sixty four processors. The major relations under investigation are execution time and communication behavior as a function of array size. Results of these experiments are currently being analyzed and documented.

[See KSL technical memos KSL-85-24, KSL-86-10, KSL-86-19, KSL-86-20, KSL-86-22, KSL-86-31, KSL-86-41.]

4. *Knowledge Acquisition*: How is knowledge acquired most efficiently from human experts, from observed data, from experience, and from discovery? How can a program discover inconsistencies and incompleteness in its knowledge base? How can the knowledge base be augmented without perturbing the established knowledge base?

   Several parallel lines of research on machine learning are in progress, representing a broad spectrum of possibilities for aiding in the construction of new knowledge bases for expert systems. Of these, significant progress was made on two aspects of learning by induction from examples. These two are documented in PhD dissertations by Li-Min Fu and Thomas Dietterich.

   Fu's dissertation investigates methods of induction in the context of learning rules and meta-rules for diagnosing cases of jaundice. The program, called RL, uses a rough model, or half-order theory, of the domain in order to guide a systematic search through a space of plausible concept definitions and associations. Experiments show that the quality of rules learned in this fashion are as good as rules derived from texts and physicians through knowledge engineering.

   Dietterich's dissertation explores another important problem in theory formation: interpreting observed data in the first place. In the case that an emerging, partially formed theory is used to interpret the data, there is ample opportunity for erroneous extensions to the theory that is being developed to explain the data. We have defined a method for "theory-driven data interpretation" that propagates constraints in order to determine a consistent interpretation of the data. This has been implemented in a program called PRE. The domain in which PRE operates is learning descriptions of UNIX file commands from examples of I/O behavior of a UNIX system in use.

   In addition, we have completed a prototype program that serves as a learning

apprentice for systems developed under NEOMYCIN. The model is general, and the program is being tested in NEOMYCIN's domain of medical diagnosis. Its purpose is to "watch" the interaction of an expert diagnosing a difficult case and to build a set of knowledge structures that will allow NEOMYCIN to diagnose similar cases in the same way.

"Chunking" is a learning mechanism that acquires rules from goal-based experience. SOAR is a general problem-solving architecture with a rule-based memory that can use the learning capabilities of chunking for the acquisition and use of macro-operators. Rosenbloom *et al.* are investigating chunking in SOAR and find that chunking obtains extra scope and generality from its intimate connection with the sophisticated problem solver (SOAR) and the memory organization of the production system.

Two MSAI theses (Hewett and Harvey) address learning from human experts. Hewett's program, MARCK, interviews a domain expert to determine why the expert prefers problem-solving actions not chosen by the application system. Harvey's program, WATCH (which is not completely implemented yet), abstracts a domain expert's control heuristics by observing the his or her problem-solving actions. Both of these programs operate in the context of application systems implemented in the BB1 architecture.

[Preliminary results have been published in KSL-85-11, KSL-85-20, KSL-85-26, KSL-85-30, KSL-85-32, KSL-85-34, KSL-85-35, KSL-85-36, KSL-85-38, KSL-85-42, KSL-85-43, KSL-85-44, KSL-85-51, KSL-85-53, KSL-85-56, KSL-86-1, KSL-86-6, KSL-86-7, KSL-86-35, KSL-86-38.]

5. *Knowledge Utilization*: By what inference methods can many sources of knowledge of diverse types be made to contribute jointly and efficiently toward solutions? How can knowledge be used intelligently, especially in systems with large knowledge bases, so that it is applied in an appropriate manner at the appropriate time?

   A PhD dissertation by Greg Cooper has been completed in which a model of inexact reasoning is proposed and demonstrated using both probabilistic and causal knowledge. The key idea is that estimates of probability ranges can be modified by using knowledge of causal relations.

   [See KSL technical memos KSL-85-14, KSL-85-18, KSL-85-23, KSL-85-25, KSL-85-35, KSL-85-40, KSL-85-41, KSL-85-46, KSL-86-26, KSL-86-30.]

6. *Software Tools*: How can specific programs that solve specific problems be generalized to more widely useful tools to aid in the development of other programs of the same class?

   We have continued the development of new software tools for expert system construction and the distribution of packages that are reliable enough and documented so that other laboratories can use them. These include the old rule-based EMYCIN system, MRS, and AGE.

   We have continued our development and refinement of BB1, including the following new capabilities: generic control knowledge sources for refining abstract control plans; graphical display of the dynamic control plan and other system-state information; strategic explanation of problem-solving actions; two programs for automatically learning control heuristics from experts; general knowledge-base facilities; and "mouse-controlled" interfaces for all system functions.

We also have extended BB1 to exploit any user-defined "framework" embodying task-specific action knowledge (see "Progress" above). We have implemented the ACCORD framework for systems that assemble arrangements of objects under constraints. We have released BB1 to approximately fifteen research groups outside of Stanford.

The ACCORD language allows definition of knowledge sources at a high level of description and represents a significant improvement in clarity and ease of definition of knowledge sources. We plan to release ACCORD sometime during the summer of 1986.

[See KSL technical memos KSL-85-12, KSL-85-15, KSL-86-38.]

7. *Explanation and Tutoring*: How can the knowledge base and the line of reasoning used in solving a particular problem be explained to users? What constitutes a sufficient or an acceptable explanation for different classes of users? How can knowledge in a system be transferred effectively to students and trainees?

   We have been concerned for years about the understandability of expert systems. We are currently focusing on the high resolution bit-mapped displays on Lisp workstations as a desirable mode of explaining the contents of knowledge bases. A prototype program, called GUIDON-WATCH, has been written and documented that provides easily understood windows into the problem solving activities of NEOMYCIN.

   A knowledge-based system must not only be able to recommend an action but also must provide an explanation as to why that action is the most desirable and what task or subtask the proposed action will accomplish. Control knowledge in BB1 has a hierarchical structure of *heuristics*, a current *focus*, and one or more levels of *strategy* for solving the problem. Since BB1's control knowledge is explicitly represented as knowledge sources, an explanation of the problem solving can be constructed at each level of abstraction. BB1 offers several ways for the user to determine the rationale behind its recommendations, including the "Explain", "Describe", and "Why" commands.

   Jeff Harvey designed and built the "Why" facility to run under BB1. It differs from those found in rule based systems because it explains why it recommends an action, rather than just explaining why the system is asking a question. When first asked "Why", the system describes the heuristics used to evaluate feasible actions. On additional "Why" queries, the system describes control decisions in more generality, continuing until the highest level of abstraction is reached. The "Describe" command is similar to "Why", but all of the control decisions are explained at the present level of abstraction.

   "Explain" gives the rationale for why the decision is a good thing to do, in terms of the ratings of that action by each of the active heuristics. The action with the highest rating is the one recommended by BB1.

   [See KSL technical memos KSL-85-39, KSL-86-2, KSL-86-15, KSL-86-34.]

8. *Planning and Design*: What are reasonable and effective methods for planning and design? How can symbolic knowledge be coupled with numerical constraints? How are constraints propagated in design problems?

We have made significant progress in this last year in merging AI and decision-analytic methods in developing plans. This is largely reflected in the program named ONYX, which "backs up" the reasoning in ONCOCIN with planning at a more fundamental level.

[See KSL technical memos KSL-85-10, KSL-85-52, KSL-85-55. ]

9. *Diagnosis and Therapy Management*: How can we build a diagnostic system that reflects any of several diagnostic strategies? How can we use knowledge at different levels of abstraction in the diagnostic process? How can a rational therapy plan be devised that is tailored to the specifics of an individual case?

   ONCOCIN (see separate section on ONCOCIN) is the primary vehicle for studying therapy management, and substantial progress has been made in representing and using therapy plans.

   [See KSL technical reports: KSL-85-21, KSL-85-22, KSL-85-29, KSL-85-33, KSL-85-50, KSL-86-3, KSL-86-4, KSL-86-5, KSL-86-9. ]

**Relevant Publications**


**KSL 85-10** Curtis Langlotz, Lawrence Fagan, Samson Tu, John Williams, and Branimir Sikic; **ONYX: An Architecture for Planning in Uncertain Environments,** May 1985. **7 pages**

**KSL 85-11** Shoko Tsuji, Edward H. Shortliffe; **Graphics for Knowledge Engineers: A Window on Knowledge Base Management,** April 1985. **23 pages**

**KSL 85-12** Stuart J. Russell; **The Compleat Guide to MRS,** June 1985. **121 pages**

**KSL 85-14** Matthew L. Ginsberg; **Implementing Probabilistic Reasoning,** April 1985. **12 pages**

**KSL 85-15** Lane, Differding, and Shortliffe; **Graphical Access to Medical Expert Systems: II. Design of an Interface for Physicians,** July 1985. **22 pages**

**KSL 85-16** **(Working Paper)** William J. Clancey; **Representing Control Knowledge as Abstract Tasks and Metarules,** April 1985. **56 pages**

**KSL 85-17** Mark Musen, Curtis Langlotz, Lawrence Fagan, and Edward Shortliffe; **Rationale for Knowledge Base Redesign in a Medical Advice System,** April 1985. **6 pages**

**KSL 85-18** Vineet Singh, and Michael Genesereth; **PM: A Parallel Execution Model for Backward-Chaining Deductions,** August 1985. Submitted for publication to: *Future Computing Systems,* 1985. **26 pages**

**KSL 85-19** Mark H. Richer; **An Evaluation of Knowledge-Based Software Tools,** May 1985. **21 pages**

**KSL 85-20** **STAN-CS-85-1068.** Mark H. Richer and William J. Clancey; **GUIDON-WATCH: A Graphic Interface for Browsing and Viewing a Knowledge Based System,** September 1985. *IEEE Computer Graphics and Applications,* Vol. 4, No. 11, November 1985, pp. 51-64. **33 pages**

**KSL 85-21** D.H. Hickam, E.H. Shortliffe, M.B. Bischoff, A.C. Scott, C.D. Jacobs; **A Study of the Treatment Advice of a Computer-Based Cancer Chemotherapy Protocol Advisor,** July 1985. To appear in the: *Annals of Internal Medicine,* 1985. **34 pages**

KSL 85-22 D.L. Kent, E.H. Shortliffe, R.W. Carlson, M.B. Bischoff, C.D. Jacobs; **Improvements in Data Collection Through Physician Use of a Computer-Based Chemotherapy Treatment Consultant,** May 1985. **20 pages**

KSL 85-23 **(Working Paper)** David Heckerman; **Probabilistic Interpretations for MYCIN's Certainty Factors,** May 1985. **28 pages**

KSL 85-24 Penny Nii; **Research on Blackboard Architectures at the Heuristic Programming Project,** May 1985. **11 pages**

KSL 85-25 Matthew L. Ginsberg; **Decision Procedures,** September 1985. **22 pages**

KSL 85-26 David C. Wilkins, William J. Clancey, & Bruce G. Buchanan; **An Overview of the Odysseus Learning Apprentice,** August 1985. *Machine Learning: A Guide to Current Research, Academic Press,* 1986. **4 pages**

KSL 85-27 (Working Paper)] G.D. Rennels, E.H. Shortliffe; **Medical Advice Systems.** May 1985. To appear in: *Encyclopedia of Artificial Intelligence,* published by John Wiley & Sons. **16 pages**

KSL 85-28 **(Working Paper)** E.H. Shortliffe; **The State of the Art of the Science.** *Proceedings of the Conference on Medical Information Sciences,* U. of Texas Health Science Center at San Antonio, July 1985. **10 pages**

KSL 85-29 Michael G. Walker, Robert Blum, and Lawrence M. Fagan; **MINIMYCIN: A Miniature Rule-Based System.** Published in: *Clinical Computing,* Vol. 2, No. 4, 1985. **8 pages**

KSL 85-30 Differing, Combs, Musen, Lane, Fagan, & Shortliffe; **Graphical Access to Medical Expert Systems: III Design of a Knowledge Aquisition Environment,** August 1985.

KSL 85-31 William J. Clancey; **Review of Sowa's "Conceptual Structures",** August 1985. To appear in the: *Journal of Artificial Intelligence,* 1985. **12 pages**

KSL 85-32 **(Working Paper)** Timothy F. Thompson and William J. Clancey; **The CASTER System: An Experiment in Knowledge Acquisition Within a Generic Expert System Shell,** August 1985. **25 pages**

KSL 85-33 James F. Brinkley; **Knowledge Driven Ultrasonic Three-Dimensional Organ Modelling,** August 1985. Published in: *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-7, No. 4, pp 431-441. July 1985. **11 pages**

KSL 85-34 John Laird, Paul Rosenbloom, & Allen Newell; **Chunking in SOAR: The Anatomy of a General Learning Mechanism,** September 1985. In: *Machine Learning,* Vol. 1, No. 1, 1986. Also appears as **Xerox PARC ISL-13** and **CMU-CS-85-154. 34 pages**

KSL 85-35 Barbara Hayes-Roth, Bruce Buchanan, Olivier Lichtarge, Mike Hewett, Russ Altman, James Brinkley, Craig Cornelius, Bruce Duncan, and Oleg Jardetzky; **Elucidating Protein Structure from Constraints in PROTEAN,** October 1985. Submitted for publication to: *Insight Series in Applied AI.* **29 pages**

KSL 85-36 **(Working Paper)** Peter Karp; **Thesis Proposal: Qualitative Simulation and Discovery in Molecular Biology,** September 1985. **31 pages**

KSL 85-37 Bruce G. Buchanan; **EXPERT SYSTEMS: Working Systems and the Research Literature,** December 1985. Appears in: *Expert Systems.* **55 pages**

**KSL 85-38** Bruce G. Buchanan; **Some Approaches to Knowledge Acquisition,** October 1985. Appears in: *Proceedings of the Third Int. Workshop on Machine Learning.* **5 pages**

**KSL 85-39** **(Working Paper)** Glenn D. Rennels, Edward H. Shortliffe and Perry L. Miller; **A Model of Choice and Explanation in Medical Management,** October 1985. Appears in: *Computers and Biomedical Research.* **17 pages**

**KSL 85-40** **(Thesis)** Jeffrey S. Rosenschein; **Rational Interaction: Cooperation Among Intelligent Agents,** October 1985. **133 pages**

**KSL 85-41** Buchanan, B.G., Hayes-Roth, B., Lichtarge, O., Altman, A., Brinkley, J., Hewitt, M., Cornelius, C., Duncan, B., and Jardetzky, O.: **The Heuristic Refinement Method for Deriving Solution Structures of Proteins,** March 1986. In: *Proceedings of the National Academy of Science.* **10 pages**

**KSL 85-42** Li-Min Fu and Bruce G. Buchanan; **Inductive Knowledge Acquisition for Rule-Based Expert Systems,** October 1985. Submitted for publication: *Artificial Intelligence.* **34 pages**

**KSL 85-43** **(Working Paper)** Robert L. Blum; **Two Stage Regression: Application to a Time-Oriented Clinical Database,** October 1985. To appear in: *Statistics in Medicine.* **27 pages**

**KSL 85-44** **(Thesis)** Li-Min Fu; **Learning Object-Level and Meta-Level Knowledge in Expert Systems,** November 1985. **229 pages**

**KSL 85-46** **(Working Paper)** Richard Treitel and Michael R. Genesereth; **Choosing Directions for Rules,** March 1986. Accepted for publication by *AAAI-86.* Submitted for publication to: *Journal of Automated Reasoning.* **36 pages**

**KSL 85-50** **(Journal Memo)** G.D. Rennels, E.H. Shortliffe, F.E. Stockdale and P.L. Miller; **Reasoning from the Clinical Literature: a "Distance" Metric,** November 1985. To appear in: *Proceedings of AAMSI Congress 86,* Anaheim, CA, May 1986. **6 pages**

**KSL 85-51** **(Journal Memo)** M.A. Musen, J.A. Rohn, L.M. Fagan and E.H. Shortliffe; **Knowledge Engineering for a Clinical Trial Advice System: Uncovering Errors in Protocol Specification,** November 1985. To appear in: *Proceedings of AAMSI Congress 86,* Anaheim, CA, May 7-10, 1985. **5 pages**

**KSL 85-52** **(Journal Memo)** C.P. Langlotz, L.M. Fagan and E.H. Shortliffe; **Overcoming Limitations of Artificial Intelligence Planning Techniques,** November 1985. To appear in: *Proceedings of AAMSI Congress 86,* Anaheim, CA, May 8-10, 1986. **5 pages**

**KSL 85-53** **(Journal Memo)** M.A. Musen, L.M. Fagan and E.H. Shortliffe; **Graphical Specification of Procedural Knowledge for an Expert System,** December 1985. To be presented at: *Second IEEE Computer Society Workshop on Visual Languages,* Dallas, TX, June 1986. **18 pages**

**KSL 85-54** Devika Subramanian and Bruce G. Buchanan; **A General Reading List for Artificial Intelligence,** December 1985. **63 pages**

**KSL 85-55** **(Working Paper)** C.P. Langlotz, L.M. Fagan, S.W. Tu and B.I. Sikic; **An Architecture for Planning under Uncertainty,** December 1985. **19 pages**

**KSL 85-56** **(Journal Memo)** D.M. Combs, M.A. Musen, L.M. Fagan and E.H. Shortliffe; **Graphical Entry of Procedural and Inferential Knowledge,**

December 1985. To appear in: *Proceedings of AAMSI Congress 1986* **5 pages**

**KSL 86-1** **(Journal Memo)** M.A. Musen, L.M. Fagan, D.M. Combs and E.H. Shortliffe; **Facilitating Knowledge Entry for an Oncology Therapy Advisor Using a Model of the Application Area,** January 1986. Accepted for publication in: *Proceedings, MEDINFO 86.* **9 pages**

**KSL 86-2** **(Journal Memo)** E. Horvitz, D. Heckerman, B. Nathwani and L. Fagan; **The Use of a Heuristic Problem-Solving Hierarchy to Facilitate the Explanation of Hypothesis-Directed Reasoning,** January 1986. Accepted for publication in: *Proceedings, MEDINFO 86.* **5 pages**

**KSL 86-3** **(Journal Memo)** C.P. Langlotz, L.M. Fagan, S.W. Tu, B.I. Sikic and E.H. Shortliffe; **Combining Artificial Intelligence and Decision Analysis for Automated Therapy Planning Assistance,** January 1986. Accepted for publication in: *Proceedings, MEDINFO 86.* **5 pages**

**KSL 86-4** **(Journal Memo)** M.G. Kahn, L.M. Fagan and E.H. Shortliffe; **Context-Specific Interpretation of Patient Records for a Therapy Advice System,** January 1986. Accepted for publication in: *Proceedings, MEDINFO 86.* **5 pages**

**KSL 86-5** **(Journal Memo)** G.D. Rennels, E.H. Shortliffe, F.E. Stockdale and P.L. Miller; **Reasoning from the Clinical Literature: The Roundsman System,** January 1986. Accepted for publication in: *Proceedings, MEDINFO 86.* **5 pages**

**KSL 86-6** **(Journal Memo)** S.M. Downs, M.G. Walker and R.L. Blum; **Automated Summarization of On-Line Medical Records,** January 1986. Accepted for publication in: *Proceedings, MEDINFO 86.* **5 pages**

**KSL 86-7** **(Journal Memo)** M.G. Walker and R.L. Blum; **Towards Automated Discovery from Clinical Databases: the RADIX Project,** January 1986. Accepted for publication in: *MEDINFO 86* **5 pages**

**KSL 86-9** **(Journal Memo)** Gregory F. Cooper; **A Diagnostic Method That Uses Casual Knowledge and Linear Programming in the Application of Bayes' Formula,** January 1986. Accepted for publication in: *Computer Methods and Programs in Biomedicine.* **24 pages**

**KSL 86-10** **(Working Paper)** James Rice; **The Poligon User's Manual,** February 1986. **68 pages**

**KSL 86-11** **(Working Paper)** William J. Clancey; **From Guidon to Neomycin and Heracles in Twenty Short Lessons,** February 1986. **48 pages**

**KSL 86-15** **(Working Paper)** William J. Clancey; **Qualitative Student Models,** February 1986. Submitted for publication to: *First Annual Review of Computer Science.* **88 pages**

**KSL 86-17** Robert S. Engelmore & Craig W. Cornelius; **Heuristic Programming Project, October 1982 - September 1985, Final Report,** February 1986. **16 pages**

**KSL 86-19** J. P. Rice; **Poligon, A System for Parallel Problem Solving,** April 1986. To appear in Proceedings of DARPA Workshop on Expert Systems Technology Base, Asilomar, April 1986. **19 pages**

**KSL 86-20** J.R. Delaney; **Multi-System Report Integration Using Blackboards,** March 1986. Accepted for publication in: *1986 American Control Conference.* **12 pages**

**KSL 86-22** Eric Schoen; **The CAOS System,** March 1986. To appear in Proceedings of DARPA Workshop on Expert Systems Technology Base, Asilomar, April 1986. **70 pages**

**KSL 86-26** **(Working Paper)** C.P. Langlotz, E.H. Shortliffe and L.M. Fagan; **Using Decision Theory to Justify Heuristics,** March 1986. Accepted for publication in: *AAAI-86* **15 pages**

**KSL 86-27** **(Working Paper)** William J. Clancey; **The Science and Engineering of Qualitative Models,** March 1986. Submitted for publication to: *AAAI-86, Science Track: Applications, Philosophical and Scientific Foundations.* **20 pages**

**KSL 86-30** **(Working Paper)** David C. Wilkins & Bruce G. Buchanan; **On Optimizing Rule Sets When Reasoning Under Uncertainty,** April 1986. Submitted for publication to: *AAAI-86.* **14 pages**

**KSL 86-31** Nelleke Aiello; **User-Directed Control of Parallelism: The CAGE System,** April 1986. To appear in Proceedings of DARPA Workshop on Expert Systems Technology Base, Asilomar, April 1986. **12 pages**

**KSL 86-32** **(Working Paper)** Peter D. Karp & David C. Wilkins; **An Analysis of the Deep/Shallow Distinction for Expert Systems,** April 1986. **18 pages**

**KSL 86-34** **(Working Paper)** William J. Clancey, Mark Richer, David C. Wilkins, Steve Barnhouse, Curt Kapsner, David Leserman, John Macias, Arif Merchant and Naomi Rodolitz; **Guidon-Debug: The student as knowledge engineer,** April 1986. **17 pages**

**KSL 86-35** **(Working Paper)** Michael G. Walker; **How Feasible is Automated Discovery?** April 1986. Submitted for publication to: *IEEE Expert.* **24 pages**

**KSL 86-41** H. Penny Nii; **CAGE and POLIGON: Two Frameworks for Blackboard-based Concurrent Problem Solving,** April 1986. To appear in Proceedings of DARPA Workshop on Expert Systems Technology Base, Asilomar, April 1986. **9 pages**

**Funding Support**

We are pursuing a broad core research program on basic AI research issues with support from not only SUMEX but also DARPA, NASA, NSF, and ONR. SUMEX provides some salary support for staff and students involved in core research and invaluable computing support for most of these efforts. Additional salary support comes from the sources listed below.

**Boeing Computing Service Company**
**Project Title:** Research on Representation Systems
**Principal Investigators:** Michael Genesereth
**Award Amount:** $75,000
**Period Covered:** 2/1/84-1/31/86

**Agency:** Boeing Computing Services Company
**Project Title:** Research on Blackboard Problem-Solving Systems
**Principal Investigators:** Edward A. Feigenbaum and Bruce G. Buchanan
**Amount:** $225,000
**Period Covered:** 2/1/85 - 3/31/86

**Agency:** Defense Advanced Research Projects Agency; N00039-83-C-0136

**Project Title:** Heuristic Programming Project
**Principal Investigators:** Edward A. Feigenbaum and Bruce G. Buchanan
**Amount:** $3,354,493
**Period Covered:** 10/1/82 - 9/30/85 (Note: New three-year contract in negotiation.)

**Agency:** Defense Advanced Research Projects Agency; MDA903-83-C-0188
**Project Title:** Research Computing Equipment Modernization
**Principal Investigator:** Edward A. Feigenbaum
**Amount:** $2,565,000
**Period Covered:** 6/1/83 - 5/31/86

**Agency:** Defense Advanced Research Projects Agency; F30602-85-C-0012
**Project Title:** Expert Systems on Multiprocessor Architecture
**Principal Investigator:** Edward A. Feigenbaum
**Award Amount:** $1,873,511
**Period Covered:** 3/14/85 - 3/13/87

**Agency:** Lawrence Livermore
**Project Title:** Research on Intelligent Budget Planning and Resource Management Systems
**Principal Investigator:** Bruce G. Buchanan
**Award Amount:** 124,905
**Period Covered:** 12/14/84 - 9/30/86

**Agency:** Josiah Macy, Jr. Foundation
**Project Title:** A Family of Intelligent Tutoring Programs for Medical Diagnosis
**Principal Investigator:** Bruce G. Buchanan
**Award Amount:** $503,415
**Period Covered:** 3/1/85 - 2/29/88

**Agency:** Martin-Marietta Corporation
**Project Title:** Intelligent Task Automation
**Principal Investigator:** Michael Genesereth
**Period Covered:** 1/1/85 - 12/31/85

**Agency:** NASA-Ames Research Center
**Project Title:** Research On Knowledge Representation
**Principal Investigator:** Bruce G. Buchanan
**Amount:** $343,144
**Period Covered:** 10/1/83 - 12/31/87

**Agency:** NASA-AMES Research Center; NCC 2-220, S1
**Project Title:** Research on Advanced Knowledge-based System Architectures
**Principal Investigator:** Edward A. Feigenbaum
**Award Amount:** $381,417
**Period Covered:** 10/1/82 - 1/31/87

**Agency:** National Science Foundation; MCS-8310236
**Project Title:** Applications of AI to Molecular Biology
**Principal Investigator:** Edward A. Feigenbaum
**Award Amount:** $405,836
**Period Covered:** 11/1/83 - 10/31/86

**Agency:** National Science Foundation