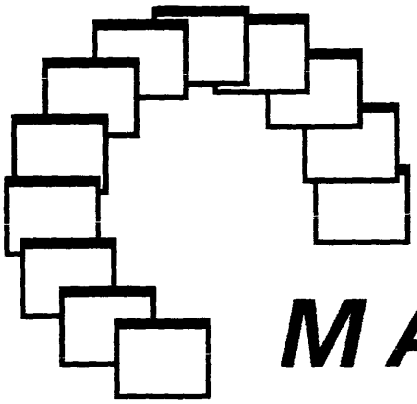
Masterscope (c) Xerox 1985



Volume 1, Number 2

April 1985

MASTERSCOPE

EDITORS' MESSAGE

This is the second issue of MASTERSCOPE, the Xerox Interlisp Users' Group newsletter. As stated in the inaugural issue, the objective of this newsletter is to provide a forum for you, the users, to discuss problems, solutions, and the projects that you are involved in. So far there has been a healthy response from the Loops community, but the response from the Interlisp-D community has not reached our expectations. This is **YOUR** newsletter and it will become an open channel between users only when users contribute items of common concern.

The Xerox AIS BU development staff is very interested in obtaining written feedback on your requests for new features or your wish list of items for forthcoming releases. Also, your comments and complaints about the software system, the organization and content of Masterscope are welcome.

At this time, we would like to announce the first Interlisp Users' Group meeting. It will be held at the August 1985 IJCAI in Los Angeles. Xerox will host the meeting, at which developers will give short talks, followed by a question and answer period. This first meeting will give you a chance to meet the management of Xerox AI Systems and other users, will get some of your questions answered, and will provide an opportunity for you to directly contribute input to future developments.

Final details and a sign-up sheet for the meeting will be included in the next issue of Masterscope.

We are looking forward to seeing you in August.

The Editors

Masterscope was generated primarily on a Xerox 1108 using TEdit and Sketch, and was printed on a Xerox 8044.

Interlisp-D

NOTECARDS

by

The Notecards Development Team

NoteCards is part of an ongoing research project in the Intelligent Systems Lab (ISL) at Xerox PARC to investigate "idea processing" tasks, such as interpreting textual information, structuring ideas, formulating arguments, and authoring complex documents. The NoteCards system, which is implemented in Interlisp-D and runs on the Xerox 1108 family of Lisp processors, provides an on-line environment for carrying out this research. Frank Halasz, Tom Moran, and Randy Trigg are the principal researchers involved in this project.

The System

NoteCards is intended primarily as an idea structuring tool, but can also be used as a fairly general database system for loosely structured information. The basic object in NoteCards is a note card, a small, idea-sized unit of text, graphics, images, or whatever. In general, note cards contain about same the amount of information as a typical 3x5 paper note card. However, the system imposes no constraints on the size of a card. Different kinds of note cards are defined in an inheritance hierarchy of note card types. The basic card types handle different substances such as text, graphics, images, etc. Specializations of these cards can be created to get cards that, for example, contain forms to be filled-in or, more interestingly, carry out a database search whenever they are retrieved.

Note cards and their associated links (see description of links below) are stored in a simple database called a NoteFile. On the screen, each note card is displayed in a window and can be edited using an editor appropriate to its substance type (i.e., text, diagrams, images, network structure, etc.). There is practically no limit to the number of cards that can be simultaneously displayed on the screen.

Individual note cards can be connected to other note cards by arbitrarily typed links, forming networks of related cards. At present, link types are simply labels attached to each link. Except for a few special system links, the type of link carries no semantics for the system. In ordinary use, it is up to each user to utilize the link types to organize the note card network. However, more specialized systems built on top of NoteCards could impose a semantics on the set of link types in a network.

Within a note card, a link is represented by a small, active icon. Clicking with the mouse in the icon, retrieves the target card and displays it on the screen. It is very easy to browse through networks of related cards simply by clicking at the link icons in each card. Under the user's control, each link icon can be made to display various pieces of information about the target card.

NoteCards includes a filing mechanism that can be used to manage large collections of cards. This mechanism is built around a special type of card called a FileBox. In each FileBox are filed (i.e., linked by a Filing link) zero or more note cards as well as zero or more other FileBoxes. The set of FileBox to FileBox

links must form a directed lattice. Thus, the FileBoxes serve as a kind of categorization hierarchy for filing note cards by "topic". In normal operation, the system "insists" that every note card be filed in at least one FileBox.

In addition to the local browsing enabled by the link icons, there are Browser cards that contain node-link diagrams, or maps, of arbitrary pieces of the note card network. Browsers are built by recursively following links of specified types emanating from a set of root cards. Each node in a Browser's node-link diagram is an active icon that can be used to retrieve the indicated card. Each link in the node-link diagram represents a link between two note cards in the NoteFile.

Spatially organized information is also available in NoteCards in the form of Sketch cards. A Sketch card allows the user to lay out line drawings, text and link icons in an arbitrary, zoomable 2-D space. It also provides the capability of drawing outline maps of any part of the world. By mixing maps and/or line drawings with link icons, the user can geographically or spatially organize sets of note cards, e.g., along a time line or at points on a map.

NoteCards is an environment that integrates several packages already available in the Interlisp-D system, e.g., TEdit, Grapher, and Sketch. NoteCards has a full programmer's interface. All of the functionality in NoteCards is accessible through a set of well-documented Lisp functions, allowing the user to create new types of note cards, develop programs that monitor or process the note card network, and/or integrate new Interlisp packages into the NoteCards environment.

Research directions

NoteCards was designed to be used primarily as a research vehicle. Following are some of the research topics that are being pursued, using the NoteCards system.

1. User tailorability -- we would like to develop a system description language that a non-programming user could edit in order to tailor the system to his or her task and/or interaction style.

2. Argumentation -- we are investigating the use of a "truth-maintenance" mechanism to help users develop and manipulate alternative argument structures within NoteCards. We also hope to investigate various general representations for arguments that could be incorporated into NoteCards structures and built into the semantics of tools running within NoteCards.

3. Psychological issues -- we are videotaping users interacting with the system and investigating the ways in which NoteCards does or does not support the real-world tasks they are performing.

4. Visual summaries of large networks -- our current Browser graphs are inefficient both in using the available screen space and in presenting information in cognitively appropriate forms. We are investigating other ways to display network maps, including fish-eye graphs, trimmed graphs, 3D graphs, indented outline, etc.

5. Multi-window management -- Screen real-estate tends to be a major problem in NoteCards since the user can have as many as 30 to 50 cards on the screen at any given time. We are investigating various

abstractions for building general multi-window management tools that will allow the user to take advantage of interdependencies between cards.

6. Querying networks of cards -- we are contemplating the design of a query language/query processor that would allow users to ask questions about the contents of individual note cards, as well as about the network structure relating note cards.

7. Multiple user, interlinked NoteFiles -- we are investigating the technical problems involved in providing distributed/shared NoteFiles with links between different NoteFiles.

8. Alternative documents -- we are developing tools for compiling linear documents from networks. We are also exploring alternative document concepts, such as guided tours, i.e., suggested paths through a network of cards.

9. Text retrieval -- we are investigating several methods for doing text retrieval based on full-text search. The most promising appears to be a statistical technique called N-gram analysis.

10. Object-oriented implementation -- we are investigating the possibility of rewriting NoteCards in Loops to take advantage of its object-oriented environment and user interface.

Concluding Remarks

NoteCards is a running, useable system with a community of about 2 dozen serious users in Xerox and a few outside organizations. We are attempting to keep the user community local and small so that we can closely observe how NoteCards is used in "field" situations. With input from our user community and from our research projects, we expect to make significant changes and improvements in the near future.

A technical paper on Notecards is in progress. For information about the research issues surrounding NoteCards contact Halasz.pa@Xerox or Trigg.pa@Xerox.

NoteCards is not at this time a Xerox product; i.e. it is not supported by 1100Support@Xerox. The system is being developed in conjunction with Xerox Special Information System's Vista Laboratories. Vista offers a limited licensing agreement aimed at distributing NoteCards to groups doing related research (Contact: Fisher.pasa@Xerox).

Notes, Cautions and Helpfull Hints

- How many times have you wanted to invoke some utility function by buttoning the background with the mouse? This functionallity has been provided in Harmony by the variable BACKGROUNDBUTTONEVENTFN. See page 31 of the Harmony release notes to use this, as well as other background functions.

- The function SETQ actually takes more than the 2 arguments which are documented in the Interlisp Reverence Manual. The extra arguments are evaluated and their values discarded. The value of the SETQ is always the value of the second argument. Thus, (SETQ AnAtom 5 (PRINT "Setting AnAtom to 5")) will print **"Setting AnAtom to 5"** and then give AnAtom the value 5 and return the value 5. This is so that forms, which are DWIMified such as (SETQ AnAtom 4 + 3), will work. In a case where the extra arguments are not *used* in constructing a value for the atom.

- The Record Package provides the WITH construct which allows one to define an expression inside of which record field names may be refered to as if they were variables (see page 3.4 of the Interlisp Reference Manual). A note at the top of page 3.5 points out that substitution in a WITH construct is *lexical*, and that it works by actually doing the substitution on the forms inside the WITH. This means that nested WITH constructs may not work when they reference field names which are shared between the two constructs. This is because when the outermost WITH construct is encountered all the record fields in the contained expression are substituted. This substitution is done without reguard to other WITH constructs.

- On Page 41 of the Harmony release notes, an extension to the AUTOBACKTRACEFLG is documented. If the AUTOBACKTRACEFLG is set to ALWAYS or ALWAYS! when a function is traced (using the TRACE function) an error break will occur. The solution to this problem is to use T or BT! for the value of AUTOBACKTRACEFLG rather than ALWAYS.

- In Intermezzo, the CAR or CDR of a non-list will not yet cause an error as announced on page 67 of the Harmony Release Notes. The variable CAR/CDRERR is now a user settable feature but as in the first issue we still recommend that it be left set to the default, NIL.

- As of Harmony the form of an expandable hash array has changed. In previous releases an expandable hash array was created by the form (LIST (HARRAY n)). In Harmony the form (HASHARRAY n) is equivalent, since the overflow handling data are stored as part of the hash array data type. A call to GETHASH can be up to 40% faster when supplied with a hash array rather than a list as an argument. All code using hash arrays should be updated to the new form. See pages 68-69 of the Harmony release notes for details on the new hash array functions.

- If you are using Harmony and have an Installation Floppy with no date stamped on it, do :

  DIR {FLOPPY}PROMETHEUS.SCRIPT CREATIONDATE

  It should print 30-Jan-85 as the date. If not call Software Support for help in updating to the correct Installation Floppy .

- In Harmony, directory listings made by DIR, FILEBROWSER, and DIRECTORY do not produce alphabetical listings of files stored on DSK. This is fixed in Intermezzo.

- The following function is compiled incorrectly by Harmony:

  (LAMBDA NIL (FTIMES x (DIFFERENCE y z].

Using both Floating-functions works, as does both non-Floating functions. This is fixed in Intermezzo. A simple patch for Harmony is:

(ADVISE 'COMP.NUMBERCALL 'BIND '(2FN].

- Because of the numerous releases, documentation for packages and system features may be found in several places. Documentation can be in any, or all, of the the package manuals, the Reference Manual, and any of the sets of Release Notes for Carol, Harmony, and Intermezzo.

LOOPS

A Shell for Intelligent Databases

by

Jon Sticklen

Ohio State University

As briefly described in the previous issue, the CSRL language/environment supports the Ohio State University view of diagnosis. However, as earlier group efforts by Sanjay Mittal (now at the Xerox Palo Alto Research Center) has shown, there is a need in full blown diagnostic systems for a "database component". To fully support diagnostic problem solving, we are in the process of building a language/environment that will support "data directed inference": the activity that we ascribe to an intelligent database.

The ASIM (A Shell for Intelligent Medical Databases) project is aimed at developing high level support for the construction and use of intelligent databases, especially in the medical domain.

The ground work for understanding database reasoning was the PATREC database assistant in the original MDX implementation built by Sanjay Mittal. PATREC provided both a data abstraction function and a course grained inference function for the diagnostic system, and was capable of temporal reasoning and reasoning about medical units. For example, PATREC was able to determine whether the patient's white blood count was normal, elevated, very low, etc. on the basis of the actual white blood count, and was able to infer that anesthetics were administered to a patient, given that the patient recently had major surgery.

This kind of reasoning is essential for medical diagnosis, but is appropriately embedded in a knowledge structure which is organized around different concepts than for classificatory diagnosis. The inferences are not diagnostic in nature, i.e., they do not relate data to diagnostic hypotheses. Also, it would be redundant to embed these inferences around each diagnostic hypothesis that requires them.

PATREC was coded in a local implementation of FRL, with most of the "demons" written directly in UCI LISP. Thus most of the inferences of PATREC could not be altered except by expert LISP programmers. In an attempt to allow access to a broader user community, the ASIM project was initiated.

ASIM is being implemented on XEROX 1108 workstations. By making full use of the graphic display tools provided by the 1108s and the LOOPS object oriented language, ASIM will provide a database language as well as an environment for the construction and updating of intelligent medical databases.

RED: a Red-Cell Antibody Identification Expert

by
J.W. Smith, John Josephson, B. Chandrasekaran
John Svirbely, Mike Tanner, Charles Evans
Ohio State University

In the AI Group at Ohio State, we have had experience over the last decade in designing and building AI systems in the medical arena. One of our systems currently in a state of refinement is the RED system. The system is designed as a consultant system for the medical blood bank expert. The basic problem being dealt with in RED is to make sure that a patient is not given a blood transfusion that will be detrimental.

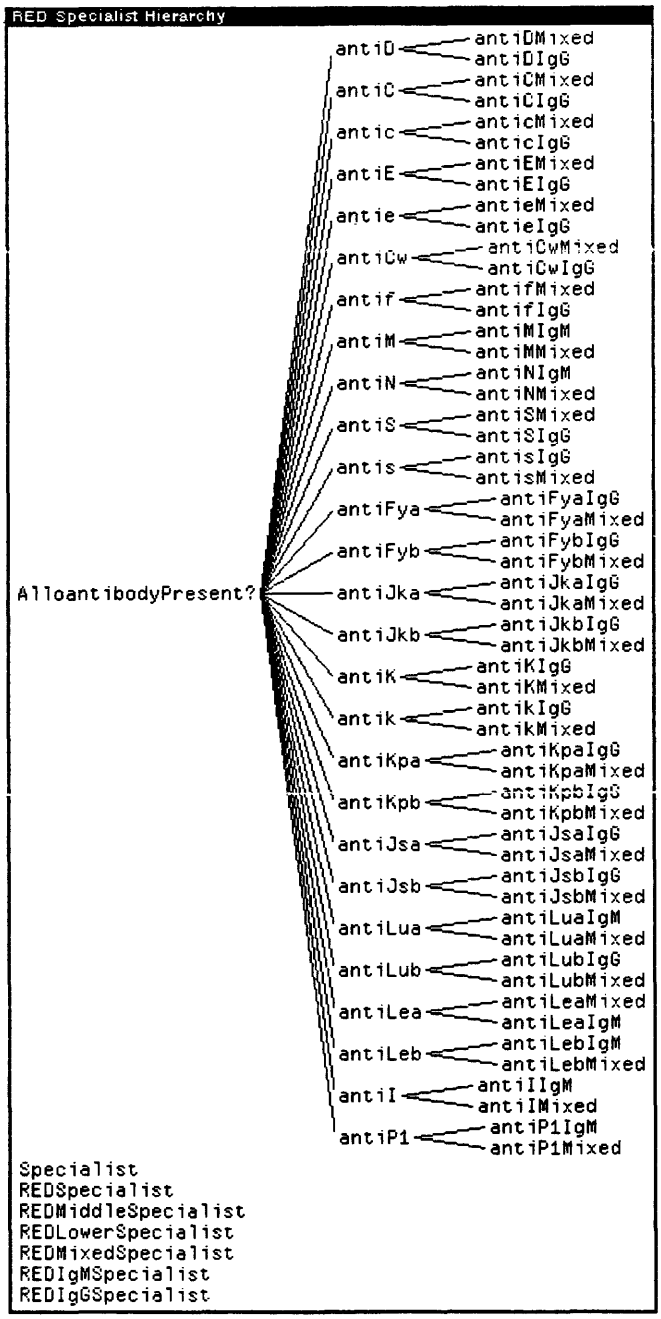
In order to screen for the presence of circulating red-cell antibodies, a small amount of a patient's blood serum is mixed with certain screening cells, chosen especially to have on their surface a full range of antigens, to see if any reaction is provoked. If a reaction is provoked, then more tests are performed to determine precisely what circulating antibodies are present. The first step is to do a "panel" which consists of mixing ten or so "cells" (i.e. specimens of identical cells) with the patient's serum in each of five or so different testing conditions. Thus approximately 50 individual tests are involved in a panel. The test cells in the panel each have certain known antigens on their surfaces. The presence or absence of approximately 30 significant antigens is known about each test cell. These known antigens may be expressed with varying strengths depending on the genetic makeup of the cell. Most of this information about the strength of expression can be inferred from the other antigens present on the cell. Any reactions that occur when the panel is done are graded by the technologist performing the panel as to strength and type of reaction. Thus, the information from a panel consists of 50 or so reactions (counting non-reaction as a kind of reaction). Each reaction is graded into approximately 7 strengths or types, on about 10 red cells, under 5 or so different test conditions, each cell having some subset of perhaps 30 antigens whose strength of expression on the cell might be one of 2 or 3 different grades.

The overall task of RED is to perform an abduction, that is, to arrive at a best explanation for the findings of the case. This explanation is critically assessed to determine exactly which of its parts are to be considered to be "confirmed", which merely "likely", and so on. Further, available explanation parts, not used as part of the best explanation, need to be critically assessed to determine with what confidence they may be rejected.

RED assembles hypotheses parts, or sub-hypotheses, which have possibly overlapping domains of explanation. A hierarchy of antibody specialists determines the plausibilities of the relevant sub-hypotheses and ascertains what test results a sub-hypothesis can account for in the particular case. A module called Overview uses the information from the specialists to build towards a complete explanation. The novel capability is exploited of confirming a sub-hypothesis on the basis of its ability to explain some feature for which there is no other plausible explanation.

Hypothesis interactions are considered to be of two general types, each with its own kind of significance for the problem-solving: (1) explanatory interactions, i.e. due to overlapping in what they can account for, and (2) substantive interactions of mutual support and incompatibility. These two senses in which hypotheses may be said to be "alternatives" are distinguished, and the problem solving organized appropriately.

Hierarchy of antibody specialists in RED



Test Cell Genetic Information

	C	D	E	c	e	Cw	f	V	K	k	Kpa	Kpb	Jsa	Jsb	Fya	Fyb	Jka	Jkb	Lea	Leb	P1	M	N	S	s	Lua	Lub
1	+	0	0	+	+	0	+	0	0	+	0	+	0	+	+	0	0	+	0	+	+	+	+	+	0	0	+
2	+	+	0	0	+	+	0	0	0	+	0	+	0	+	0	+	+	+	0	+	+s	+	+	0	+	0	+
3	+	+	0	0	+	0	0	0	0	+	0	+	0	+	+	+	+	0	0	+	0	0	+	0	+	+	+
4	0	+	+	+	0	0	0	0	0	+	0	+	0	+	+	+	+	+	0	+	+	+	0	+	0	0	+
5	0	0	+	+	+	0	+	0	0	+	0	+	0	+	0	+	0	+	+	0	+w	+	0	+	+	0	+
6	0	0	0	+	+	0	+	+	0	+	0	+	0	+	0	0	+	0	0	0	+	+	+	0	+	0	+
7	0	0	0	+	+	0	+	0	+	+	0	+	0	+	+	+	0	+	+	0	+w	0	+	0	+	0	+
8	0	0	0	+	+	0	+	0	+	+	0	+	0	+	0	0	+	+	0	+	+	+	0	0	+	0	+
9	0	+	0	+	+	0	+	0	0	+	0	+	0	+	0	+	+	0	0	+	+w	0	+	+	+	0	+
10	0	0	0	+	+	0	+	0	+	0	0	+	0	+	+	+	+	+	0	+	0	0	+	0	+	0	+

Matrix showing the antigenic makeup of the test cells

Test Reactions

	10	9	8	7	6	5	4	3	2	1
AlbuminIS	0	1+	0	0	0	0	2+	2+	2+	0
Albumin37	1+	2+	1+	1+	0	0	2+	2+	2+	0
Coombs	3+	3+	3+	3+	0	0	3+	3+	3+	3+
EnzymeIS	1+	2+	1+	1+	0	0	1+	1+	1+	0
Enzyme37	1+	2+	1+	1+	0	0	2+	2+	2+	0

Matrix showing the results of tests. The numbers indicate the strength with which the patient's blood reacted to the test cells.

MDX/MYCIN

by

Jon Sticklen

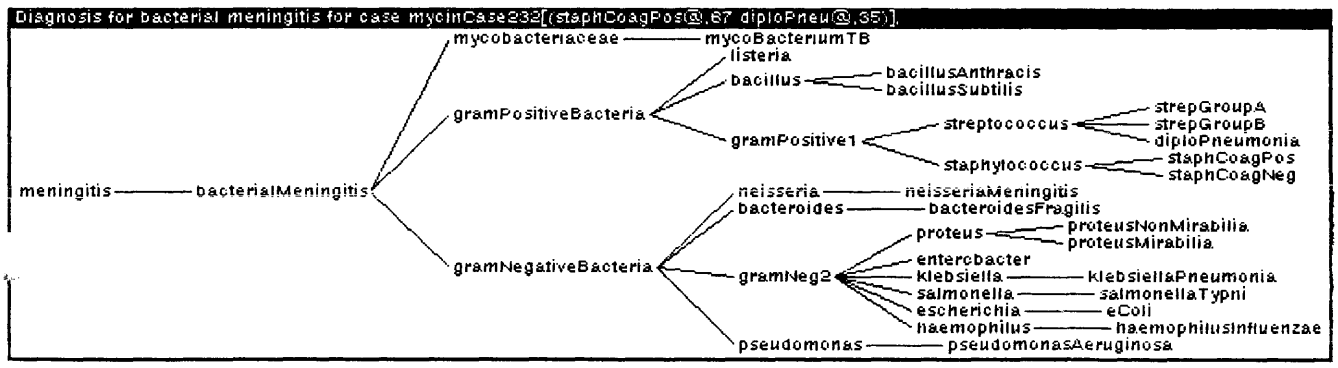
Ohio State University

One of the earliest medical diagnostic systems constructed in LOOPS within the OSU AI Group was a system attacking a sub-domain of the MYCIN system. In addition to certain theoretical points that we hoped to demonstrate with this system, MDX/MYCIN, we also quickly came to appreciate the knowledge engineering advantages offered by a marriage of the OSU diagnostic paradigm with LOOPS/INTERLISP environment.

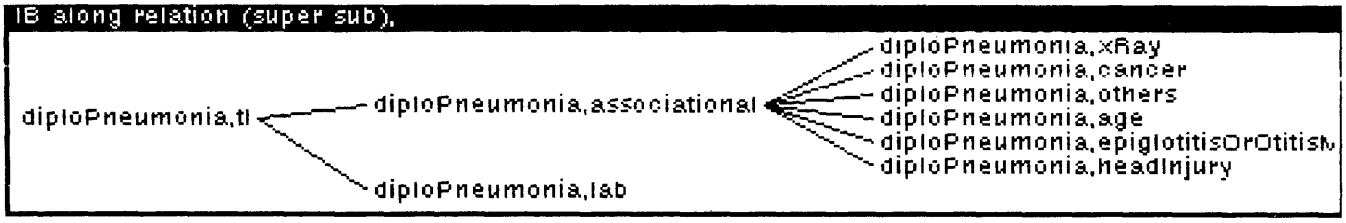
Comparison of different approaches to expert system design for a given task, such as diagnosis, is difficult since they are often embodied in systems for domains with very different characteristics. It is *a priori* difficult to decide if a given difference in the approaches is necessitated by the differences in the domain. For example, it might be suggested that MYCIN's global and numeric uncertainty calculus is needed in domains such as MYCIN's, apparently characterized by a great deal of uncertainty in knowledge and data, while the approach of MDX, another medical system, which only uses local combinations of qualitative probabilities, may be too weak in such domains. In order to study the relationship between the domain characteristics and problem solving approaches of the two systems, we constructed an MDX-like system, MDX/MYCIN, for a subdomain of MYCIN, and conducted a number of experiments on the resulting system. The results demonstrate that the MDX paradigm is effective in this domain, and, additionally, offers knowledge engineering advantages along the dimensions of debugging ease and system extensibility.

The use of the LOOPS environment coupled with the diagnostic paradigm of MDX allowed the MDX/MYCIN system to be prototyped rapidly; the phase of actual system building required but two weeks. In addition, we found that the LOOPS environment with its many graphical facilities allowed ease of system presentation to both the knowledge engineer and for demonstration purposes.

Shown below are two window images during the operation of MDX/MYCIN. The first is the complete specialist hierarchy of the system.



The second screen image is of the "knowledge groups" that make up one of the diagnostic specialists. Within the MDX paradigm, there is a two tier factoring of domain knowledge, first into specialists, then within each specialist into semantically meaningful, named knowledge groups.



Both of these images show active LOOPS browsers; by performing mouse operations on nodes in the browsers, we may perform such operations as "running" a knowledge group in isolation, and editing a knowledge group.

Auto-Mech

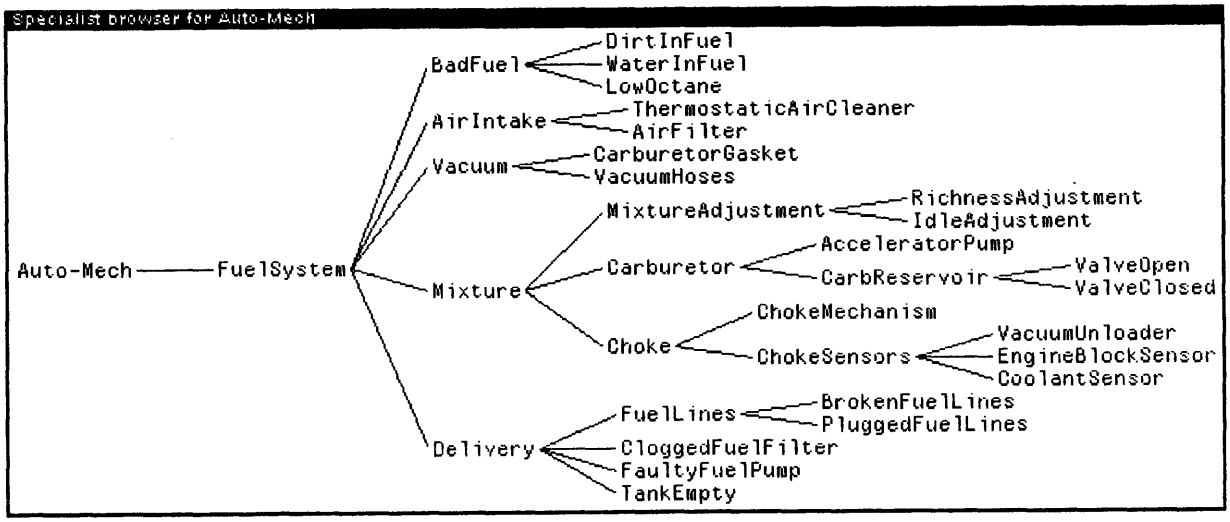
by

Mike Tanner, Tom Bylander

Ohio State University

To explore the utility of the CSRL language (mentioned above) in a non-medical domain, we undertook the construction of demonstration system working in the domain of automobile diagnosis.

Auto-Mech is an expert system which diagnoses automobile fuel systems. Its organization and strategies are patterned after MDX, an expert diagnosis system developed in our AI laboratory. The problems that these systems are able to diagnose are represented as nodes within a hierarchy. Each node has knowledge about how to confirm or reject the problem hypothesis, as well as knowledge about what nodes to consider next. This approach is intended to be a domain-independent methodology for providing focused problem solving and for localizing knowledge in a conceptually relevant manner. Auto-Mech is implemented in a recently developed language called CSRL, which is specifically intended for building diagnostic expert systems.



Specialist hierarchy for Auto-Mech

Announcements

Intermezzo

The next release, Intermezzo, will include major enhancements to the capacity and functionality of Interlisp-D.

1. The virtual address space has been enlarged from an 8MB maximum to 32MB. Interlisp-D is extremely efficient in memory utilization, and very few users have ever run out of virtual memory space even when working on large systems within the previous 8MB limit. However, anticipating the needs of increasingly ambitious development projects, this maximum has been quadrupled.
2. Aside from increasing the total addressable memory size, major improvements to memory allocation programs help minimize fragmentation.
3. The number of atoms permitted in an Interlisp-D system has been doubled, from 32K to 64K . This enhancement will handle special cases where huge numbers of atoms are generated automatically.
4. The TCP/IP protocol is available upon request as of Intermezzo. This protocol allows XEROX workstations to communicate with non-XEROX hosts, primarily UNIX systems, over the Ethernet.
5. Other features of the release include improvements to printing and filing routines, the TEdit package, and device-independent graphics functions.

Submissions to Newsletter

We need submissions for future issues. Submit articles and other items for publication, as well as requests to be added to the mailing list, as follows:

INTERLISP ARTICLES:
AINewsletter↑.pasa@Xerox
OR: US Mail
AINEWSLETTER
ms 1232
Xerox Special Information Systems
250 North Halstead Street
Pasadena, California 91109

LOOPS ARTICLES:
Hausladen.PA@Xerox
OR: US Mail
Mary A. Hausladen
Xerox AI Systems
3333 Coyote Hill Road
Palo Alto, California 94304

Also, we are now preparing a listing of software available from third party vendors. If you have anything you would like to offer in this catalogue, please contact AINEWSLETTER at the address and/or 800 phone numbers.

For Xerox software support, messages can be sent to 1100Support.pasa@Xerox. Our toll free numbers are:

(808) 228-5325 -- US, including Hawaii and Alaska
(800) 824-6449 -- within California

Interpress Documents

Because of recent interest in the Interpress Printing Standard, Xerox has made available a set of three Interpress-related documents priced at $50. This set includes the Interpress standard itself, as well as an introductory tutorial and a reader's guide.

Formerly, this package was available only in a set of 12 documents for $250 that also includes all of the published standards and protocols that make up Xerox Network Systems architecture. That package of 12 documents continues to be abavilable.

Xerox also offers classes, implementation aids, and consulting services in support of Interpress. To obtain any of the document packages mentioned, or to obtain more information about support services for Interpress and other available documents, please contact:

Dennis Frahmann,
Manager, Protocols Marketing, Xerox Corporation
2100 Geng Road, Palo Alto, CA 94303
(415)-496-6088

Interlisp and Loops Training Classes

Xerox Artificial Intelligence Systems offer Interlisp and Loops classes. Each class is one week in duration, and includes both lecture and lab. Below is a short description of our four most asked for classes.

"Introduction to Interlisp-D" is designed for programmers with no previous Lisp experience, and gives a basic working knowledge of the Interlisp-D Language and its programming environment.

"Intermediate Interlisp-D Programming" is geared toward people with programming experience and a good working knowledge of Lisp, or our "Introduction". After a brief review of the introductory material, this course gives a thorough grounding in Interlisp-D. We examine arithmetic processing, windows, menus, bit-maps, fonts, more on Lisp Data types, use of the mouse, I/O processes, error protection, error correction user packages, networking and utilizing the print and file servers.

"Advanced Interlisp-D Programming" moves on to the more sophisticated expressions of the language. Some topics covered are, error handling, monitor locks, user-defined masterscope templates, user-defined file package comands, macro processing, Ethernet communications and customizing the entire system to your needs. At least several months programming in Interlisp-D is required.

"Knowledge Programming in Loops" Loops builds on Interlisp-D to integrate procedure-oriented programming with object-oriented, access-oriented and rule-oriented programming. The course will help you determine if Loops is the right development tool for your application, and will equip you to build your own knowledge-based system. Interlisp-D programming experience is required.

We are in the process of developing a full range of expert systems development courses and will announce their availability soon. Call the Training Coordinator on extension 2676 at the above toll free numbers for more information.