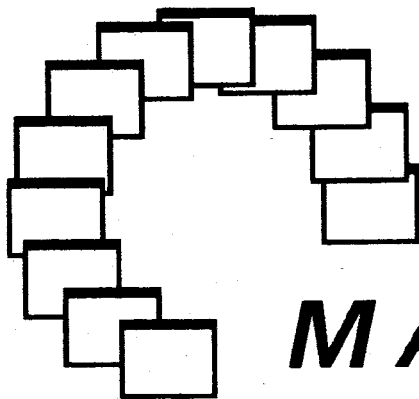




Volume 2, Number 1

August 1986

MASTERSCOPE

DABKON: Heuristic Programming for Decision Support in CAD

Martin Brooks
Center for Industrial Research
Box 350 Blindern
0314 OSLO 3
NORWAY

The department for Knowledge Based Systems at the Center for Industrial Research in Oslo has developed a demonstration expert system that helps a mechanical designer choose the appropriate type of bearing for his application. DABKON is a decision support system; the designer inputs the requirements that the bearings must fulfill, DABKON then initiates a dialogue with the designer, and finally DABKON presents a list of appropriate bearing types, ranked in order of appropriateness.

DABKON is intended to help the designer during the early phases of design, in particular before the requirements have been numerically determined. This is in accordance with design practice, where one iteratively converges on a final design through a series of decision/analysis cycles. In order to choose bearings, one first chooses the type of bearing based on general requirements, and then uses more detailed (i.e., numerical) requirements in order to determine the specific bearing and its dimensions. There exists various computer-based and manual procedures for the latter problem, however to our knowledge DABKON is the only system aiding in the choice of bearing type. Bearings were chosen as the first application area because it is a relatively simple, albeit non-trivial, decision process.

DABKON is the first of a series of expert systems intended to aid designers during the early design phases. Other applications currently being investigated include choice of gear types and choice of shaft boss

design. The work is supported by the Royal Norwegian Council for Industrial Research. Product development based on this work will be carried out together with industrial partners; readers interested in more information may contact us at the above address or telephone number.

The bottom left of Illustration No. 1 on Page 4 shows how a designer inputs the requirements to the bearings. Each column is a menu giving the possible values to a single attribute. The user must specify each attribute, at which point he continues by selecting the "finished" button. Note that with the exception to the maximum and minimum operating temperatures, the values are non-numerical. These values correspond roughly to orders of magnitude of numerical specifications.

DABKON contains a table of 11 bearing types described in terms of the attributes used in the requirements specification. This table was taken, with minor modifications, from the German handbook of systematic design ("Konstruieren mit Konstruktionskatalogen," K. Roth, Springer-Verlag 1982). There are three logically distinct results of this table lookup: (1) exactly one bearing type meets the requirements, (2) several bearing types meet the requirements, or (3) no bearing types meet the requirements. In the first case DABKON simply presents the bearing type; no expert system behavior is needed. In case (2) DABKON asks the designer a series of questions in order to rank the bearing types according to appropriateness for the specific application. In case (3) DABKON analyzes the requirements relative to the bearing types in the table in order to determine why the requirements cannot be met and then presents the designer with a series of tradeoffs between the specified attribute values. The designer then indicates which values he is willing to trade off against others. On this basis DABKON constructs several new, satisfiable requirements specifications, and presents them, together with the bearing types that satisfy them, to the designer.

Illustration No. 1 on Page 4 is a look at case (2), where several bearing types fulfill the requirements.

The top left shows a series of questions asked of the user. These questions reflect the process that a human designer would go through in order to choose between the bearing types. There are three factors determining the selection of bearing types in such a situation: the choice roller vs. journal bearings, price, and bearing lifetime. These factors are determined by a series of questions to the designer, shown in the top left.

The ranked list of bearing types is presented to the user in menu form, shown in the bottom right. Selection from this menu results in a complete description of the bearing type, shown above the menu of bearing types. The designer may ask why the bearing types were listed in that particular order by selecting the "explain" button, resulting in an explanation shown in the top right.

This completes our description of DABKON's behavior in the case that several bearing types meet the requirements. The case where no bearing type meets the requirements is more complicated. Basically the goal is to identify the least significant attributes for the application such that reducing the requirements

to these attributes results in a satisfiable specification. This is accomplished via tradeoff analysis (See Illustration No. 2 on Page 4).

The reason that a requirements specification is not satisfiable is that certain subsets of the attributes are specified such that they cannot be satisfied simultaneously. When there are two attributes in such a subset then they can be traded off against each other. For example, no bearing type can both tolerate very high shock loading and operate with very low friction. By examining the table of bearing types DABKON finds that either the requirement to shock load must be reduced to "high" or the requirement to friction must be reduced to "medium" in order that there are any bearing types at all which satisfy the requirements to these two attributes. In other cases there may also be the possibility of simultaneously reducing the requirements to both attributes in the tradeoff. Choosing one of these alternatives does not guarantee that the resulting requirements specification is satisfiable because there may be conflicts between other attributes; but it does eliminate that particular conflict.

DABKON systematically presents every conflict to the designer in graph form. The designer then indicates which alternatives are preferred, acceptable, and/or unacceptable by selecting the point on the graph and then choosing from a pop-up menu, as seen in the top of Illustration No. 2.

When the designer has indicated his preferred and acceptable alternatives to all the conflicts DABKON combinatorially generates all possible specifications incorporating these alternatives. DABKON then ranks the satisfiable specifications among them according to the number of preferred alternatives in each and presents the bearing types satisfying them in menu form. Selecting from this menu results in presentation of the new specification and complete information about the bearing type, shown in the bottom right of Illustration No. 2. In the new specification, the starred attributes are those which were changed from the original specification as a result of tradeoffs. The designer can select the "explain" button to ask how this new specification was made, resulting in the tradeoff choices used in that specification being drawn in black, as shown.

Dabkon was implemented in six man-months over a period of two months. Two man-months were used for design, four for implementation. The system consists of 3000 lines of Interlisp-D code comprising 140 functions. DABKON is an example of a "procedural" expert system, i.e., it is based on Lisp programs, not on rules. These programs embody a variety of heuristics at the specification, design, and implementation levels.

32-108601
(MIL-STD-883C)
33

Does the environment require quiet or vibration-free running? NO

Is the application exposed to vibration under standard conditions? NO

Does the application require partitioned bearings? NO

Is the shaft movement oscillatory? NO

Is the application to be produced in volume? YES

What is the diameter of the shaft? (millimeters) 25

EXPLANATION OF UNDERSPEC ORDERING
There are 5 bearing types meeting the given spec.
Roller bearings are preferable to journal bearings because of general cost and availability considerations, and because there are no special requirements concerning vibration, split casing, or oscillatory movement.
The roller and journal bearing types were sorted with respect to cost because they are to be produced in volume.

5 bearing types meet the given spec

LOAD	LOAD DIRECTION	SHOCK LOAD	ECCENTRIC LOAD	ROTATIONAL SPEED	ACCEPTABLE FRICTION	MAX TEMP.	MIN TEMP.
VERY HIGH	RADIAL	VERY LARGE	NO	VERY HIGH	HIGH	181	-42
HIGH	AXIAL	LARGE	NO	HIGH	MEDIUM	+450	0
MEDIUM	COMB	MEDIUM	NO	MEDIUM	LOW	+400	-100
LOW	COMB	SMALL	NO	LOW	VERY LOW	0	-250
VERY LOW	COMB	NONE	NO	VERY LOW	NONE	0	-200

RollerBall
RollerSpherical
JournalFluid
JournalGas
JournalMagnetic

EXIT DASHON

Illustration No. 1

MinTemp

-248

-200

-60

PREFERRED

SPECIFIED

ACCEPTABLE

None Small Medium Large VeryLarge

ShockLoad

ShockLoad reduced to Medium

MaxTemp

199

100

PREFERRED

SPECIFIED

UNACCEPTABLE

None Small Medium Large VeryLarge

ShockLoad

FINAL ATTRIBUTE PRIORITY

IMPORTANT: LoadDirection MinTemp ShockLoad

NORMAL: RotationalSpeed EccentricLoad

UNIMPORTANT: RotationalSpeed EccentricLoad

1 alternate

DEFAULT TRADE OFF CHOICES
SPEC: (RotationalSpeed Low) (MinTemp -248)
AttPri: (RotationalSpeed Unimportant) (MinTemp Important)
CHOICE: (RotationalSpeed VeryLow) (MinTemp -248)
SPEC: (MinTemp -248) (Friction Medium)
AttPri: (MinTemp Important) (Friction Unimportant)
CHOICE: (MinTemp -248) (Friction High)

NEW SPECIFICATION

LoadDirection Combined
MinTemp -248
ShockLoad Medium
MaxTemp 199
Load Medium
RotationalSpeed VeryLow
EccentricLoad No
Friction High

LOAD	LOAD DIRECTION	SHOCK LOAD	ECCENTRIC LOAD	ROTATIONAL SPEED	ACCEPTABLE FRICTION	MAX TEMP.	MIN TEMP.
VERY HIGH	RADIAL	VERY LARGE	YES	VERY HIGH	HIGH	199	-248
HIGH	AXIAL	LARGE	NO	HIGH	MEDIUM	+450	0
MEDIUM	COMB	MEDIUM	NO	MEDIUM	LOW	+400	-100
LOW	COMB	SMALL	NO	LOW	VERY LOW	0	-200
VERY LOW	COMB	NONE	NO	VERY LOW	NONE	0	-200

JournalDry

LoadDirection Radial Axial Combined
MinTemp -250
ShockLoad Medium
MaxTemp 289
Load Medium
RotationalSpeed VeryLow
EccentricLoad Yes
Friction High

Illustration No. 2

Theorist

Dr. John Kruse
3M Center
St. Paul, Minnesota

3M derives much business from a large number of polymer-based products. Polymers are long-chain compounds with carbon backbones, used in making adhesives, films, and plastics. The Theorist is an expert system developed on a Xerox 1108 workstation. Its purpose is to help 3M's polymer chemists explore the vast combinatorial space of possible polymers, by predicting the physical properties of polymers from a description of the chemical structure. Much initial effort went into definition and implementation of a user interface which allows one to graphically describe a chemical structure. The system then analyzes the structure, decomposing it into various subgroup constituents. These subgroups are represented in a database, and have associated with them a number of properties relevant to predicting useful polymer properties. Several properties of the polymer are predicted and displayed. Work can be saved, along with the user's comments, and the results of a given session can be compared to previous work.

The user interface was built on a Xerox 1108 workstation, using a great deal of Interlisp-D code and the KEETM (Knowledge Engineering Environment) product from Intellicorp. This chemical editor has the flavor of MacPaintTM, in that there are a number of activities available to the user at any given point in time, and to a large extent, a command or icon for any given activity is visible on the screen of the 1108 at all times. The editor allows the user to build up a complex polymer repeat unit (fragment) atom by atom, or by combining entire subgroups from a library. The structure can be modified, by deleting atoms or bonds, and made to resemble a publication-quality chemical structure drawing. During the editing process, an up-to-date object-based representation of the edited material is maintained.

Once a structure has been suitably described, the user can initiate the prediction routines. Some of the knowledge embodied in the system comes into play here, for example, in determining the presence of particular subgroups even when they have other subgroups contained within them. The structure search was initially carried out using a ruleset, but it has since been rewritten entirely in Interlisp-D. A number of heuristics resolve conflicts in cases where the user's structure could be broken down in more than one way.

Several algorithmic prediction routines are used to determine the bulk properties of a polymer once its composition has been determined. These can be displayed in numeric or in some cases graphical form. Further expertise allows the interpretation of these numbers, that is, the system helps the user determine whether the polymer, with its predicted properties, will be suitable for a given class of applications. Of course, given the numerical aspect of much of the information provided to the user, the system also relies heavily on graphical depiction of numerical relationships.

The system accepts feedback from individual users on the accuracy of its predictions, because knowledge about the properties and behavior of particular chemicals tends to be distributed throughout the 3M polymer community. An individual chemist may know a great deal about a particular family of

compounds, but less about other classes of compounds. Thus a user-contributed database and a collection of flexible search routines can enable some communication between specialists.

The Theorist was developed by John Collins and Dr. John Kruse of 3M's Software and Electronics Resource Center in collaboration with Intellicorp's Dr. Ray Carhart. Most of the polymer science expertise and consultation was provided by Dr. Michael Martin of 3M. In addition, it was Martin who initially proposed this particular project. The Theorist was the first expert system application undertaken at 3M, and as such it served as a learning vehicle for 3M's AI group. It was a good project to start with, because the problem was fairly well-defined, and yet the predictions are complex and tedious enough to dissuade most chemists from engaging in them. Much of the knowledge is available in books, which aided the knowledge acquisition process. The Interlisp-D environment provided the means to rapidly build a prototype system with a user-friendly interface that immediately overcomes computer-naïve users' reluctance to use a new software system. The Theorist can be used as the core of a general "lab assistant" for 3M's polymer science community.

1108/1186 Communication: TCP/IP

Ron Clarke
Customer Support
Xerox AI Systems

Since TCP/IP was introduced as a supported communication protocol in Interlisp-D over a year ago there has been a growing interest in its use. The following is a step by step set of directions for adding 1108/1186s to a network supporting TCP/IP and communications with others on the net.

OBTAIN NETWORK ADDRESS

The first thing you need to do is to get a TCP/IP address assigned to each 1108/1186 from your network administrator. You will also need to know the network addresses and operating system of the hosts you want to communicate with and the addresses of any network gateways you have. The printed representation of an Internet Protocol (IP) address takes the form of W.X.Y.Z, where W through Z are the decimal values of eight-bit bytes that constitute the address. Depending on the class of network that you have, these numbers are broken down as follows, where N = Netnumber and H = Hostnumber on that net:

- Class - A: N.H.H.H The first number is between 0-127 (e.g. 122.0.2.1)
- Class - B: N.N.H.H The first number is between 128-191 (e.g. 153.4.23.5)
- Class - C: N.N.N.H The first number is between 192-255 (e.g. 194.5.67.3)

Be sure to find out whether your net is a true Class A, B or C network and is not broken up into subnets. If it is broken up into subnets, be sure to read the discussion on SUBNETMASKs in the TCP/IP Lisp Library documentation section on "A Primer on IP Networks."

CREATE HOSTS.TXT FILE

You will then need to create a file named HOSTS.TXT which contains these addresses. Place the file on a directory contained in the DIRECTORIES search path of each workstation. For example:

```
HOST:192.20.10.1:BACH:XEROX-1108:INTERLISP:TCP/TELNET,TCP/FTP:
HOST:192.20.10.3:PARC-VAXC:VAX-11/780:UNIX:TCP/TELNET,TCP/FTP:
HOST:192.20.10.15:OBERON:VAX-11/780:VMS:TCP/TELNET,TCP/FTP:
HOST:192.20.10.30:EXPLORER:TI-EXPLORER:TOPS-20:TCP/TELNET,TCP/FTP:
HOST:174.23.77.22:SUNRISE:SUN:UNIX:TCP/TELNET,TCP/FTP:
HOST:174.23.30.21:RUTGERS:VAX-11/780:TOPS-20:TCP/TELNET,TCP/FTP:
HOST:174.23.76.21:SIMBA:SYMBOLICS:SYMBOLICS-3600:TCP/TELNET,TCP/FTP:
GATEWAY:192.20.10.240,174.23.77.250:HELLSGATE:VMS:IP/GW:
```

This example shows a host table with four hosts (BACH, PARC-VAXC, OBERON 192.20.10.0, three hosts (SUNRISE, RUTGERS and SIMBA) on net 174.23.0.0 and that connects the two. **Note:** All entries must be in uppercase, there should be a carriage return and a carriage return should terminate each line. T.I. Explorers™ users should be using the operating system.

Also, if you ever modify the HOST.TXT table after TCP.DCOM has been loaded (HTE.READ.FILE 'hosttable') to reread the file. For example (HTE.READ.FILE '{dsk

CREATE LOCAL IP.INIT FILE

Load the file TCPCONFIG.DCOM and perform (TCP.CONFIGURE). Follow the procedure on the local disk of each 1108/1186 machine on the net. For the 1108 in the a

```
HOSTNAME:    BACH
HTE.FILE:    {dsk}<lispfiles>hosts.txt (or where ever the host:
DEFAULT.GATEWAY: 192.20.10.240
LOCAL.NETWORKS: 192.20.10.0 (since it is a Class - C network)
SUBNETMASK:    192.20.10.0 (Same as local net if no subnet routing
LOCAL.ADDRESSES: 192.20.10.1
```

This will create a file {dsk}<lispfiles>ip.init on the local disk. **Note:** This file should be created on an Interlisp-D machine **before** TCP.DCOM is loaded.

ADD HOST AND OPERATING SYSTEM NAME TO NETWORKOSTYPES

The variable NETWORKOSTYPES is used during Chat to determine the sequence when performing auto-login. There should be an entry in NETWORKOSTYPES for each host you want to communicate with in the form (TCPHOSTNAME . OSTYPE). For example ((SUNRISE . UNIX)(RUTGERS . TOPS-20) etc).

LOADING TCP

Make sure the variables DIRECTORIES and LISPUSERSDIRECTORIES point to the location of the following DCOM files: TCP, TCPHTE, TCPLLAR, TCPLLICMP, TCPBLIP, TCPCHAT, TCPDEBUG, TCPFTP, TCPNAMES, TCPTFTP and TCPUDP.

You can then load TCP.DCOM which in turn loads the DCOM versions of TCPLLIP, TCPHTE, TCPLLICMP and TCPLLAR.

If you plan to do TCP file transfers, load TCPFT which also loads TCPNAMES.

If you plan to Chat to a TCP host read the Lisp Library package Chat documentation first. Then load the library package CHAT, CHATTERMINAL, DMCHAT and then TCPCHAT.DCOM.

If you plan to use the TCP Trivial File Transfer Protocol load TCPTFTP.

VERIFYING TCP CONNECTIONS

Load TCPDEBUG. Execute (TCPTRACE T) and you will be prompted to open a window to show TCP packets. Select Incoming, Outgoing and Contents from the window's menu. If the host that you are communicating with has a TCP echoserver process you can then try (TCP.ECHOTEST 'HOSTNAME 3) . For example, using the above HOST.TXT file this would be (TCP.ECHOTEST 'SIMBA 3)). You will be prompted to open a window for the echo test and should see printed in it something like:

This is byte number 21

This is byte number 45

This is byte number 69

You should also see packets being sent and received in the TCPTRACE window.

Note: If a remote host is not running a TCP echo server process you will not get this response.

CONNECTING, FILE TRANSFER AND CHATTING TO A HOST

First login to the host by typing (LOGIN 'host). For example, (LOGIN 'SUNRISE). You will then be prompted for a user name and password. This will be sent to the host when you attempt to CONNECT or Chat. Use the command CONN {host}<directory>subdir> to connect to a particular host. All operating systems, except for UNIX, can use this directory/subdirectory specification. For UNIX you need to specify it as {host}</dir/subdir/>. Using the above HOST.TXT file as an example you can do the following:

UNIX:	CONN {SUNRISE}</dir/subdir/sudir/>
VMS:	CONN {OBERON}<dir>subdir>subdir>
TOPS-20:	CONN {RUTGERS}<dir>subdir>
SYMBOLICS-3600:	CONN {SIMBA}<dir>subdir>

You can then do a DIR of the remote host, COPYFILE files to and from the host, assuming TCPFTP is loaded, MAKEFILE, etc.

Since the TCPFTP specification does not specify filetype conventions, the variable TCP.DEFAULT.FILETYPES is used to associate a file's extension with the type of file it is. It is a list in the form (extension . type). For example, ((DCOM . BINARY) (TXT . TEXT) etc). Since UNIX systems are case sensitive you should also have the lower case version of the file extensions on this list. If a file extension is not found on this list, the variable TCP.DEFAULTFILETYPE is used as the default file type during file transfers

Note: From a remote host you will not be able to CONN or do a DIR to an 1108/1186 machine because there is no TCPFTP server available at this time. However, if the remote host supports the trivial file transfer protocol (TFTP), Interlisp-D provides a TCPTFTP server. Load TCPTFTP.DCOM and perform (TFTP.SERVER). You can then use the appropriate TFTP commands to copy files to and from the Interlisp-D machine. For example TFTP.PUT and TFTP.GET

To Chat to a remote host select Chat from the background menu and enter the host name when prompted. You will be prompted for a chat window and should then be able to chat to the host. If you have problems opening the chat connection try evaluating (CHAT 'host 'NONE). This will suppress the automatic login.

MAKING A SYSOUT THAT CONTAINS TCP/IP

1. Load Koto Sysout.
2. Create TCP host table.
3. Load TCPCONFIG.DCOM and run TCP.CONFIGURE if there is no IP.INIT file on local disk.
4. Load TCP.DCOM, TCPFTP.DCOM, TCPCHAT.DCOM.
5. Load TCPDEBUG.DCOM if you always want to have trace and echo facilities available.
6. Evaluate (TCP.STOP)
7. Evaluate (STOPIP)
8. Evaluate (SETQ RESTARTETHERFNS (LIST '(LAMBDA NIL (AND \IPFLG (\IPINIT)))))
9. Loading any other files that you want in this syout.
10. Evaluate SYSOUT to the device of your choice. Evaluate (\TCP.INIT) to reenale TCP.
11. Load TCP sysout on other machine
12. Create TCP host table.
13. Evaluate (TCP.CONFIGURE) and identify the new machine
14. Evaluate (\TCP.INIT) to reenale TCP
15. Evaluate (/IPRINT) to restart the IPLISTENER

TCP DEBUGGING AIDS

With TCPDEBUG loaded use (TCPTRACE T) to open up a trace window of TCP traffic. The appropriate items need to be selected from the windows menu in order for data to be seen.

(SETQ TCPCHAT.TRACEFLG T) will print TELNET negotiations to a file, which is what the variable TCPCHAT.TRACEFILE points to.

(FTPDEBUG T) opens a scrolling trace window that displays FTP commands as they are issued. Note: You will see unencrypted passwords if they are issued.

KNOWN PROBLEMS

You must use the new 1186 microcode in order for TCP/IP to work on an 1186. This was released as a new Systems Files # 1 floppy and is part of the Koto 2.0 Patch Kit.

TCP/IP will not work with UNIX systems that have Trailer Encapsulation enabled. Connections will hang and then eventually break.

Directory enumeration on a VMS system results in NIL.

If you have to call the function (TCP.CONFIGURE) again to modify any information, you MUST put in the value of the SUBNETMASK. It does not default properly if you simply hit a carriage return.

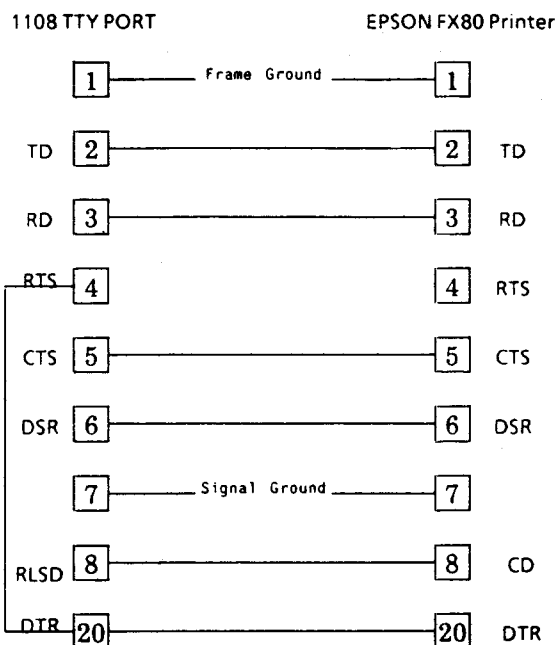
NOTES, CAUTIONS and HELPFUL HINTS

- In Koto there is a problem printing from an 1108/1109/1186 to an Epson FX80 printer which uses Epson's Intelligent Serial Interface Card. This problem manifests itself in loss of characters, blank pages or hanging of the RS232/TTY port. This problem has not shown up on FX80s that use other serial interface cards (e.g. Practical Peripherals Interface Card).

The following workaround is for printing from the 1108's TTY port using FASTFX80STREAM Lisp Library package.

Create a cable with the following configuration:

Cable Configuration for 1108 TTY Port to EPSON FX 80 Printer



Epson Intellieght Serial Interface card switch configuration:

SW-1	SW-2
1-OFF	1-ON
2-OFF	2-ON
3-OFF	3-ON
4-OFF	4-ON
5-OFF	5-OFF
6-ON	
7-OFF	
8-OFF	

Software Installation:

1. Load a fresh sysout.
2. (SETQ DIRECTORIES '({dsk}<lispfiles> {floppy}))
3. (SETQ LISPUSERSDIRECTORIES '({dsk}<lispfiles> {floppy}))
4. (SETQ DISPLAYFONTDIRECTORIES '({DSK}<LISPPFILES>FONTS>)) or to wherever your fonts are located. Make sure MODERN10-C0.DISPLAYFONT is located on that directory before loading TEdit.
5. Load TEDIT.DCOM which should load all of its related files.
6. Load DLTTY.DCOM
7. Load FX80STREAM.DCOM
8. Load FASTFX80STREAM.DCOM
9. (TTY.INIT 9600 8 'NONE 1 'DTR)
10. To print a TEdit document: Right button in the TEdit window's title bar, move to hardcopy and slide off to the right to select "To a File". Type in {TTY}.FASTFX80 and a carriage return. After formatting the file, printing will be initiated.

See the FASTFX80STREAM documentation for complete details for printing.

- ❑ In Interlisp-D there are several spelling lists maintained which are used for spelling correction and completion. If a Litatom name is on the USERWORDS list then after typing only the first several characters the name can be expanded by typing the TAB key in Intermezzo and the EXPAND key in KOTO. The four variables are SPELLINGS1, SPELLINGS2, SPELLINGS3 and USERWORDS. These can be added to directly with ADDSPELL. For a more extensive description see Section 20.7 in the three volume Interlisp Reference Manual.
- ❑ **COPYFILES:** there is a bug in COPYFILES when using the RENAME option. All files that have more than one version are renamed with the opposite version numbers, for example:

```
(COPYFILES 'foo* 'fie* 'RENAME)
```

causes foo;3 to become fie;1.

- Suppose you want to enter A B C and have it return (A B C). This functionality is provided by a documented function READLINE which is used for DEDIT type-in and also provides the functionality which permits one to enter FN(arg) at the top level. Some users have found READLINE difficult to use in that it does not behave the same as READ and most of the other input functions. To use READLINE one must enter it in combination with one of the other input functions for example :

```
(CONS (LISPXREAD T T) (READLINE)).
```

FLOPPY.SCAVENGE

If (FLOPPY.SCAVENGE) does not work try:

```
(ADVISE '\FLOPPY.ASSUREFILENAME 'AROUND
        '(PROGN (SETQ NOERROR T)
                (OR * (GENSYM 'BADFILENAME))))
```

Then perform (FLOPPY.SCAVENGE) again. After scavenging the floppy, there will probably be some filenames like {FLOPPY}BADFILENAME0009. You should then RENAMEFILE these files with new names. If FLOPPY.SCAVENGE still does not work, there is also FLOPPY.TO.FILE.

FLOPPY.TO.FILE will put the content of the floppy into a file. Note that if something is wrong with the directory structure on your original floppy, copying the contents of that floppy to a second floppy via FLOPPY.TO.FILE and then FLOPPY.FROM.FILE won't clear the problem up. It will give you a second bad floppy that looks exactly like your first bad floppy. FLOPPY.TO.FILE can be used to put the contents of your floppy into one big file on a file server that you can then edit to get at the text of your files. Since source files also begin with FILECREATED, one good way to find the beginning of files in your floppy image would be to search for "FILECREATED".

- When hardcopying graphs there is a problem which causes the nodes not to be printed on the page if the nodes have a negative X or Y position. The following ADVICE will translate the coordinate of the graph appropriately. To compile the patch, load UTILSPORS. Note: loading UTILSPORS will effect the way THEN and ELSE pretty print in DEdit.

```
(DEFINEQ (DSPGRAPH.TRANSPATCH (LAMBDA (GRAPH)
  (LET* ((GRAPHNODES (fetch GRAPHNODES of GRAPH))
        (MINX (for NODE
                  in GRAPHNODES minimize
                  (DIFFERENCE (fetch XCOORD
                                of (fetch NODEPOSITION
                                      of NODE))
                              (QUOTIENT (fetch NODEWIDTH of NODE)
                                           2))))
        (MINY (for NODE in GRAPHNODES minimize
                  (DIFFERENCE (fetch YCOORD
                                of (fetch NODEPOSITION
                                      of NODE))
                              (QUOTIENT (fetch NODEHEIGHT
                                                of NODE)
                                           2)))))
    (CONS (LISPXREAD T T) (READLINE)).
```

```
(create POSITION
```

```
  XCOORD ←(DIFFERENCE 2 MINX)
```

```
  YCOORD ←(DIFFERENCE 2 MINY)]
```

```
(ADVISE 'DISPLAYGRAPH-IN-HARDCOPYGRAPH 'BEFORE
```

```
  '(OR TRANS (SETQ TRANS (DSPGRAPH.TRANSPATCH GRAPH))))
```

- ❑ The Koto release has solved several questions which have come up over the past few months. The following are described in the Koto Release Notes.

The function DIR now has a DELVER option which will delete all but the highest version of a file.

There is a new function VMEM.PURE.STATE which can be used to modify the page swapping algorithm so that a partition is always rebootable. This has been requested by those sites which have their 1108/1186 workstations in a pooled setting. Please note that when running in "pure state" the vmem use is increased substantially so that it is most appropriate for situations where regular rebooting from a clean sysout is desired.

- ❑ For those with 3.5 Mb memories, it is usually possible to backup your current working state by executing SAVEVM every 20-30 minutes. On average it appears to take about that long before swapping begins.
- ❑ **TCP-IP:** In Intermezzo, if a sysout is created with TCP loaded, and that sysout is installed and booted, a "Hard Disk Error" will occur under \IPINIT.

Corrective Action:

Before making the sysout that contains TCP/IP do:

```
(CHANGENAME '\ETHEREVENTFN '\IPINIT 'NILL)
```

This will disable the system from trying to restart IP on reboot. In the new sysout, you must call (\IPINIT 'AFTERSYSOUT) explicitly after each reboot and after any calls to (RESTART.ETHER). Before continuing, you will also need to call (\IPINIT 'AFTERSYSOUT) on the machine that the sysout was originally made on.

- ❑ **PRINTOUT:** There are several documented formate options; .SUB, .SUP, .BASE which do not work in either Intermezzo or Koto and cannot be used.

Also, several users have asked why PRINTOUT does not reset the DISPLAYFONT when completed. Conceptually if you are doing print formatting then the programmer has taken control of the format. It is just as reasonable to assume that one wants the last format to remain as to be reset to the original default. So it is left up to the programmer. A recommended procedure for a lot of I/O is to wrap these functions in RESETFORM's and related functions which can be used to control system state when a form has completed either as programmed or via an error. eg.

```
(RESETFORM (DSPFONT NIL) (PRINTOUT T .FONT ..... ))
```

QUESTIONS AND ANSWERS

- Q.** How do I create a function without having it noticed by the file package?
- A.** Use the function (PUTD fn def). PUTD will put the definition, def into the function cell of fn without being noticed by the the file package. Similarly with PUTDQ.
- Q.** Is there any easy way to keep the TTY window from constantly bringing itself to the top? I have a program that prints information in many windows and it's rather annoying to have to move them to be able to see what's in them. It's especially annoying to bury or shrink the TTY window and have it immediately pop back up.
- A.** You can keep the TTY window from constantly bringing itself to the top by giving the keyboard to another process and (optionally) suspending the EXEC process. You can do this either programatically (using TTY.PROCESS and SUSPEND.PROCESS) or by invoking the process status window (PSW) from the background menu (use the KBD← and SUSPEND commands). The MOUSE process is an alternate process you can give the keyboard.
- Q.** How can I retrieve the image that is stored under a particular window?
- A.** (WINDOWPROP window 'IMAGECOVERED) will return the bitmap image from under the window.
- Q.** How can I change the font of a window's title bar?
- A.** (DEFINEQ (GIVE.BIG.TITLEW (WINDOW TITLE FONT) (** will give WINDOW the title TITLE in the font FONT and return the old title as WINDOWPROP would) (PROG ((OLDFONT (WINDOWTITLEFONT FONT)) (OLDTITLE(WINDOWPROP WINDOW (QUOTE TITLE) TITLE)))) (WINDOWTITLEFONT OLDFONT) (RETURN OLDTITLE]
- Q.** How can I print a TEdit document in Landscape format?
- A.** The following procedure can be used to format a Landscape TEdit document. It does not send the document immediately to the printer, but creates an Interpress file that can be hardcopied upon demand.
- (1) You have to reset the TEdit page margins to make them appropriate for landscape printing. Essentially, you want to change the margins so the document is formatted as 11 x 8.5. The easiest way to do this is to use the Page Layout Menu. Unit of measurement is picas, with 6 picas to the inch. Changing the top margin to 20.0 and the right margin to -20.0 would give you a 1 in. margin on the top of the landscaped page and .5 inch on the right.
 - (2) Open an interpress imagestream specifying a destination file.
- (SETQ IMSTRM (OPENIMAGESTREAM 'MYFILE.IP 'INTERPRESS '(LANDSCAPE T)))

- (3) Hardcopy the TEdit file to the stream, but don't send it to the printer.
(TEDIT.HARDCOPY (OPENTEXTSTREAM 'TEditFile) IMSTRM T)
- (4) Close the imagestream and the TEdit file.
(CLOSEF IMSTRM) this will return the full file nome for MYFILE.IP.
(CLOSEF 'TEditFile)
- (5) Print the file.
(NSPRINT 'Jupiter:aissouth 'MYFILE.IP)

Q. How do I verify that my Ethernet is working between two 11XXs (e.g. 1108 -> 1186)

A. On Machine A type \MY.NSADDRESS. This will return something like 3500#0.125000.67113#0. On Machine B type (NS.ECHOUSER '3500#0.125000.67113#). Note that the number typed is the same as the number given on Machine A without the number after the second "#". If the test is successful, you will see a pattern such as !+!+!+!+. The ! indicates a successful packet transmission; the + indicates the packet was successfully echoed back. Anything else indicates that the net is down.

Q. If I store a sysout on my local file system by doing (SYSOUT '{dsk}<lispfiles>lisp.sysout) or (MAKESYS '{dsk}<lispfiles>lisp.sysout "SYSOUTNAME") how can I let other machines load it into a partition over the ethernet (i.e. as if it were on a file server).

- A. (1) On each machine evaluate the variable \MY.NSHOSTNUMBER. This will return something like (NSHOSTNUMBER 0 43585 5079). Also evaluate (PORTSTRING(ETHERHOSTNUMBER)) on each machine. This will return something like 2#4# which is the net#pup#.
- (2) On the machine that has the sysout stored on it load the Koto Lisp Library packages FTPSERVER.DCOM and MINISERVE.DCOM.
- (3) On this same machine build the association list NS.TO.PUP.ALIST of the XNS Host number to PUP ID number in the form ((NSHOSTNUMBER a b c) . PUPNumber) for each machine that will be accessing the sysout. For example, (SETQ NS.TO.PUP.ALIST '(((NSHOSTNUMBER 0 43585 5079) . 3) ((NSHOSTNUMBER 0 2343 25239) . 7) etc)
- (4) Start the FTPSERVER process by calling (FTPSERVER). Start the PUP ID SERVICE process by calling (STARTMINISERVER).
- (5) To load the sysout on a particular machine:
 - a. Get into the system tool and select the PUP-FS item under "Device:"
 - b. For FILE type in [net#pup#]<directory>name.sysout. "net#pup#" is what is returned by (PORTSTRING(ETHERHOSTNUMBER)) on the machine where the sysout is stored and has the FTPSERVER and PUPIDSERVICE process running "directory" is the directory on which the sysout file is stored. If it is on the default directory, the field can be omitted. "name.sysout" is the sysout name. (e.g. [2#4#]lisp.sysout))

- c. Select SYSIN! or SYSIN & BOOT! to start the SYSIN process.

Q: How do you change the default font used for all new TEdit windows?

A: There is no longer a global variable referencing the TEdit default font itself. Instead, the FONT property is one of a number of properties which can be set using the global variable TEDIT.DEFAULT.PROPS. For example, if you want to change the font to Modern 12 Regular, do the following:

```
(SETQ TEDIT.DEFAULT.PROPS (LIST (QUOTE FONT)(FONTCREATE (QUOTE MODERN) 12
  (QUOTE MRR))))
```



LOOPS

PAGE-1: An Application of LOOPS to Troubleshooting

Mohammed Nasiruddin, John C. Sederberg and Charles H. Strandberg
Honeywell, Inc.

INTRODUCTION

PAGE-1 is a knowledge-based system developed for aiding the troubleshooting of Honeywell non-impact Page Printing Systems (PPS). The PPS produces high-quality computer printout at up to 18,000 lines per minute. The printout may consist of text, forms, or a combination of both; it may be sorted, collated or stacked.

PAGE-1 is a tool to improve the field engineer's troubleshooting efficiency. The field engineer interacts with the system in a way that will make the troubleshooting process more cost effective. The problem for the design team was to develop an efficient, cost-effective system. PAGE-1 was implemented using LOOPS on a Xerox 1108. PAGE-1 also runs on the Xerox 1186.

THE APPLICATION DOMAIN

The difficulty associated with troubleshooting the PPS resides in its complexity there are about 25 sub-systems involving approximately 2,000 components. The sub systems are of different types: computer hardware and software for control, mechanical for paper manipulation, hydraulic for mixing and distributing printing fluid, HVAC for drying paper and reclaiming fluids, and electrical for motors, high-voltage printing surfaces and computer circuits.

PAGE-1 OVERVIEW

The PAGE-1 expert system has four major component: knowledge base, inference engine, user interface and the knowledge-base editor.

KNOWLEDGE BASE

The PAGE-1 knowledge base is divided into two layers of networks: a hypothesis layer and the troubleshooting assertion language (TAL) layer.

The hypothesis layer is basically a graph in which a node (a LOOPS object) corresponds to a hypothesis of failure and a link between two nodes corresponds to a cause/effect relationship between the two hypotheses. Each hypothesis object contains information about its causes (i.e. children), its effects (i.e.

parents), a pointer to its semantic representation in the TAL layer, a pointer to the test method in the TAL layer, the cost of performing the test, and a method for fixing the problem if it is an ultimate cause of failure.

Each hypothesis in the hypothesis network has its meaning and its test meaning represented as TAL assertions in the TAL layer. Each TAL assertion is itself represented as a semantic network of LOOPS objects. Formally, TAL is an empirically determined subset of first-order predicate calculus without quantification and is governed by a graph grammar. TAL assertions are constructed from four types of primitives: components, attributes, relations and events. The component object type represents the physical components of the PPS that are significant in troubleshooting. All components are arranged in a system/sub system hierarchy which roughly corresponds to the real system hierarchy. Each component can have an arbitrary number of attribute objects linked to it. Attributes have an actual value which is by default, unknown, a normal value and a set of constraints on the actual values. The event object represents events of interest that occur within the system. The relation object type represents relationships between components, relationships between attributes and constants, relationships between relations and relationships between events.

Relation and Attribute object types are the only entities in TAL that can have a value. Values for relation and attribute objects are determined by local derivation of the truth value of the TAL assertion they are in. This is realized computationally as propagation of values using the active value feature of LOOPS. Because of the way TAL assertion objects are shared, propagation of values may cause a change in value of another TAL assertion in addition to the one being derived. This mechanism is independent of the inference engine.

INFERENCE ENGINE

The basic goal of the inference engine is to traverse the hypothesis network in search of a terminal hypothesis (a hypothesis without causes) that can be confirmed. If such a hypothesis is found the search is successful. A data-driven strategy is used which is basically a heuristic generate-and-test strategy. The state-space for this problem is the hypothesis network. The search is represented using a notion of a context (a set of hypotheses). On each cycle, the most probable hypothesis is selected and tested. If the result of the test is true, a new context is created with the children of the current hypothesis as its contents. Failure on the test results in the current hypothesis being discarded and the next most likely hypothesis being selected. If the test results are unknown, all the children of the current hypothesis are inserted in the current context. There are a couple of exceptions to this basic strategy. If the user makes an assertion to the system, the inference engine performs a jump to the area indicated by the user. The same thing happens when the system itself deduces a fault which is not on the search path.

USER INTERFACE

One of the goals of the project was to be able to interact with PAGE-1 either by sitting in front of the workstation or via remote communications using a modem. The user interface and the inference engine are implemented as two separate processes which communicate with each other through a data structure called the 'purple board', which is a LOOPS object. The inference engine posts all its queries on the purple-board. The user interface is constantly scanning the purple-board. When it encounters a question, it displays it to the user and conveys the user response back to the inference engine via the purple board.

In the local mode of operation, the user interface provides support for tracking the progress of the search. This could take the form of displaying the relevant portion of the hypothesis network, a listing of the hypotheses being considered and the hypothesis currently selected. In the remote communications mode only textual information is presented.

KNOWLEDGE BASE EDITOR

The knowledge-base editor is built around the Lattice Browser. This graphic editor can be used to enter, modify, verify and browse through the knowledge base. A search facility is provided for searching through the network for nodes that match the given description. Multiple views of the knowledge base can be obtained. The user need not have any knowledge of LOOPS or Lisp to use the editor, making it possible for the domain expert to enter and modify the knowledge base.

STATUS

PAGE-1 was started in November 1984 and a beta-version was available in December 1985. Currently, PAGE-1 is being field tested. Results from the testing have lead to additions to the knowledge base and improvement of the user interface. Current development work on PAGE-1 focuses around better anglicizing of the questions, consistency checks on the knowledge base and improved remote communications facilities.

Alarm Filtering System

Dan Corsberg

Idaho National Engineering Laboratory (INEL)

For several years, the INEL has been involved in the research, development, and evaluation of signal processing systems that identify a nuclear power plant's operating state during plant disturbances. Recently, that work has focused on the use of intelligent, knowledge-based systems such as the Alarm Filtering System (AFS) that was developed for a reactor at INEL. AFS analyzes alarm data from process instrumentation and responds to that data according to knowledge encapsulated in objects and rules. AFS specifically addresses the information overload problem that operators can experience during major process transients. Using AI techniques and a knowledge-based approach to this problem, INEL is utilizing the power of the computer to filter and analyze the plant sensor data presented to human operators and provide them with a better description of the process state. The on-rush of process data is analyzed by AFS and only the data that is of importance to the operators is emphasized. Using functional relationships in hierarchical rulesets, AFS performs the following:

- Generates a description of a situation implied by combinations or sequences of alarms.
- Suppresses display of information that confirms or is a direct consequence of a previously described situation.
- Emphasizes alarms that do not fit previous conclusions or alarms that are expected (due to previous alarms or conditions) but are not received within specified time periods.

AFS is implemented in LOOPS which has allowed several programming paradigms to be integrated into a functional system. Objects are used to represent all the information in AFS that is specific to the process being monitored. The message-sending capabilities of the objects are used to invoke further processing as required and to notify other objects of status changes. Rules are used to encapsulate the general knowledge about how to handle and respond to types of alarm and process state relationships. These rules contain the decision-making knowledge that is generic in the sense that it doesn't address specific alarms or process states. Access-oriented programming is used to make the system truly data driven and allow the system to respond to process states even when alarms are not activated in expected sequences.

AFS has simplified the interface between plant and operator while simultaneously attacking the alarm filtering problem with new techniques. As a system, AFS integrates rule-oriented programming into an object-oriented environment to avoid exhaustive searches of extensive databases or structures. Since the only rules checked and only objects referenced are those directly related to the event being processed, AFS's computational requirements per event are not dependent upon the alarm space size, but rather upon the relational complexity of those alarms.

The use of an object-oriented alarm model has also resulted in a very flexible system where, once the functional relationships of the alarms and states have been defined, a working system can easily be built and maintained. The modularity of the knowledge base provides AFS with the ability to handle modifications in the instrumentation since those changes do not affect the structure of the decision-making process, only the knowledge that process uses. This same separation of knowledge makes the technology and approach used in AFS applicable to a diverse set of systems and processes.

Possible future work on AFS includes developing some 'deep' knowledge systems that would have an understanding of the functions of the systems being monitored. The 'shallow' knowledge of AFS would handle a high percentage of situations, but when a conflict is recognized or verification is needed, AFS would hand the problem up to the 'deep' knowledge system for further analysis. In addition, because of the data driven aspect of AFS and the relative independence of the signal processing, AFS seems to be a natural application of the use of multiple processors.

AFS is described in detail in 'An Object-Oriented Alarm-Filtering System', presented at the Power Plant Dynamics, Control, and Testing Symposium, Knoxville, Tennessee, April 14-16, 1986.

ANNOUNCEMENTS

FROM XEROX AIS

TERMINAL FONTS (6, 8, 10, 12, REGULAR AND BOLD) AND LISP PATCH SERVICE 5.0 AVAILABLE FOR 8044 PRINTER SERVER

The Koto Release of 1186 and 1108 Interlisp-D and 1108 Xerox Quintus Prolog makes a new font, Terminal (6, 8, 10, 12, Regular and Bold), available for display and printing.

If you purchased your Xerox 8044 Print Server from A Xerox Business Systems Group (BSG) representative rather than from the Artificial Intelligence Systems organization, you may not have received these fonts for the 8044 Print Server.

Additionally, the Intermezzo Release included the distribution of a floppy disk labeled Lisp Patch Service 5.0. This patch corrected a timeout problem (signalled by a "Connection lost" error in Lisp). It was installed on the 8044 Print Server. This patch is still required in the Koto Release. Again, if you purchased your Xerox 8044 Print Server from a BSG representative during the time of the Koto Release, you may not have received this patch.

Please contact Product Operations for your Terminal Fonts Kit.

USERS' GROUP ACTIVITIES

Contact the following people for information on Users' Group Meetings in your area:

NORTHEASTERN AREA (NEW YORK, PENNSYLVANIA and NEW JERSEY)

- New York and area code (201) in New Jersey:
George Hunt -- (212) 916-2240
Xerox Artificial Intelligence Systems
2nd Floor
205 E. 42nd Street
New York, NY 10017
- Pennsylvania and (609) area code (609) in New Jersey:
Damon Connor -- (201) 663-1411
Xerox Artificial Intelligence Systems
21 Ripplewood Drive
Lake Hopatcong, NJ 07849

SWXAISLUG (ARIZONA, NEVADA, UTAH, SOUTHERN CALIFORNIA & HAWAII)

Josh Stern -- (818) 351-2351
 Xerox Artificial Intelligence Systems
 250 Noth Halstead St.
 Pasadena, CA

NECIAI (NEW ENGLAND)

A.J. Matsis -- (603) 881-7028
 Xerox Artificial Intelligence Systems
 400 Amherst Street
 Suite 300
 Nashua, NH 03063

MAX-AI (DELAWARE, MARYLAND, D.C., VIRGINIA and WEST VIRGINIA)

Wellington Pitts (703) 247-6952
 Xerox Artificial Intelligence Systems
 1901 North Moore St.
 Lower Lobby Suite 5
 Arlington, VA 22209

XEROX AIS EDUCATIONAL SERVICES

Xerox Artificial Intelligence Systems offers Interlisp-D, Loops and Prolog classes. Each class is one week long and includes both lecture and lab. Below is the schedule for the second half of 1986.

DATE (week of)	CLASS	LOCATION
7 July	LISP I: Introduction to Interlisp-D	Pasadena, CA
14 July	LISP II: Intermediate Interlisp-D Programming	Pasadena, CA
21 July	LOOPS I: Programing in LOOPS	Pasadena, CA
8 Sept	LISP I: Introduction to Interlisp-D	Pasadena, CA
15 Sept	LISP II: Intermediate Interlisp-D Programming	Pasadena, CA
6 Oct	LISP I: Introduction to Interlisp-D	Leesburg, VA
13 Oct	LISP II: Intermediate Interlisp-D Programming	Leesburg, VA
20 Oct	LOOPS I: Programing in LOOPS	Leesburg, VA
17 Nov	Prolog I: Introduction to Xerox Quintus Prolog	Pasadena, CA
1 Dec	LISP II: Intermediate Interlisp-D Programming	Pasadena, CA
8 Dec	LOOPS I: Programing in LOOPS	Pasadena, CA

For more information contact your local Sales Representative or the Xerox Educational Services Coordinator at extension 2676 at the toll free numbers.

SUBMISSION TO NEWSLETTER

Masterscope is published by Xerox Artificial Intelligence Systems to serve as a communications link between Xerox and its customers. Without your participation and contributions, Masterscope will not serve as that link. Please send submissions for future Masterscope issues and requests to be added to the mailing list to:

Via network:

AINewsletter ↑ .pasa@Xerox

Via US Mail:

AI Newsletter

ms 432

Xerox Artificial Intelligence Systems

250 North Halstead Street

P.O. Box 7018

Pasadena, CA 91109-7018

Xerox®, Interlisp-D and LOOPS are trademarks of Xerox Corporation
August 1986

Printed in USA

Masterscope
Xerox Artificial Intelligence Systems
250 North Halstead Street
P.O. Box 7018
Pasadena, CA 91109-7018