```
    consistent with a ground state quadrupole deformation of epsilon
    2=0.28+or-0.01 assuming axial symmetry and a uniform charge
    distribution. In the ground state rotational sequence a backbend is
    observed at h(cross)wc approximately=0.32 MeV. This appears to be due
    to the alignment of h11/2 protons. These properties are compared with
    the results of potential energy surface and cranked shell model
    calculations and are assessed in the context of the systematic
    behaviour of even-even nuclei in this mass region.
CC  5A2110F; 5A2760; 5A2320L; 5A2320C; 5A2320E; 5A2110R
CT  GAMMA-RAY ANGULAR DISTRIBUTION; NUCLEAR COLLECTIVE STATES AND GIANT
    RESONANCES; NUCLEAR ENERGY LEVEL LIFETIMES; NUCLEAR ENERGY LEVEL
    TRANSITIONS; NUCLEAR SHAPE; NUCLEI WITH MASS NUMBER 90 TO 149
ST  gamma decays; excited states; 126Ce; 92Mo(40Ca, alpha 2p)126Ce;
    multiple particle-gamma coincidence; angular distributions;
    production cross sections; lifetimes; ground state quadrupole
    deformation; axial symmetry; uniform charge distribution; ground
    state rotational sequence; backbend; potential energy surface;
    cranked shell model; even-even nuclei
ET  Ce; 126Ce; is; Ce is; Mo(Ca,alpha 2p)Ce; 92Mo(40Ca,alpha 2p)126Ce;
    92Mo t; 40Ca r; 40Ca alpha 2p; 126Ce f; Mo is; 92Mo; Ca is; 40Ca

L5  ANSWER 2 OF 2
AN  87:2966449  INSPEC     DN  A87109119
TI  Measurement of lifetimes in the light rare-earth region.
AU  Moscrop, R.; Campbell, M.; Gelletly, W.; Goettig, L.; Lister, C.J.;
    Varley, B.J. (Dept. of Phys., Manchester Univ., England); Price, H.G.
SO  Proceedings of the International Nuclear Physics Conference
    Editor(s): Durrell, J.L.; Irvine, J.M.; Morrison, G.C.
    Bristol, England: IOP 1987. p. 160 vol.1 of 2 vol. (vii+556+xvi+606)
    p.; 1 refs.
    Conference: (IUPAP), Harrogate, Yorks., England, 25-30 Aug. 1986
DT  Conference
TC  Experimental
LA  English
AB  Lifetime measurements for the lowest states of 13262Sm70 and 136Gd72
    have been performed using the recoil distance method. Measurements
    were made using a 180 MeV 40Ca beam on a 0.7+or-0.1 mg/cm2 92Mo
    target set at nineteen positions relative to a fixed gold stopper
    (corresponding to mean flight times of 20 ps to 3.2 ns) and on a
    0.9+or-0.1 mg/cm2 96Ru target set at eleven positions (corresponding
    to mean flight times of 78 ps to 3.2 ns). Lifetime results are
    presented and the deduced B(E2)'s discussed with regard to the
    variation of shape and deformation of the region.
CC  4A2320C; 4A2110F; 4A2760; 4A2320L; 4A2570G
CT  HEAVY ION-NUCLEUS REACTIONS; NUCLEAR ENERGY LEVEL LIFETIMES; NUCLEAR
    ENERGY LEVEL TRANSITIONS; NUCLEAR SHAPE; NUCLEI WITH MASS NUMBER 90
    TO 149
ST  nuclear level lifetimes; 132Sm; 136Gd; B(E2); 132Sm decay into
    126Ce+4He+2p; 132Sm decay into 128Ce+4p; 136Gd decay into 132Nd+4p;
    136Gd decay into 133Pm +3p; 92Mo(40Ca,2p alpha )126Ce;
    92Mo(40Ca,4p)128Ce; 96Ru(40Ca,4p)132Nd; 96Ru(40Ca,3p)133Pm; lowest
    states
ET  Gd72; 136Gd72; is; Gd is; 136Gd; Ca; 40Ca; Ca is; Mo; 92Mo; Mo is;
    Ru; 96Ru; Ru is; Sm; 132Sm; Sm is; Gd; Ce*He; Ce sy 2; sy 2; He sy 2;
    Ce is; 126Ce; He is; 4He; 126Ce+4He; Ce; 128Ce; Nd; 132Nd; Nd is; Pm;
    133Pm; Pm is; Mo(Ca,2p alpha )Ce; 92Mo(40Ca,2p alpha )126Ce; 92Mo t;
    40Ca r; 40Ca 2p alpha ; 126Ce f; Mo(Ca,4p)Ce; 92Mo(40Ca,4p)128Ce;
    40Ca 4p; 128Ce f; Ru(Ca,4p)Nd; 96Ru(40Ca,4p)132Nd; 96Ru t; 132Nd f;
    Ru(Ca,3p)Pm; 96Ru(40Ca,3p)133Pm; 40Ca 3p; 133Pm f
```

This reaction would be written $^{92}Mo(^{x}Ca,...)^{126}Ce$. In this formula x is the isotope number for the calcium nucleus, and the ellipse contains other specifications about the reaction such as leaving particles. To conduct this search on a version of INSPEC that contains the ET field, a successful search strategy would involve searching that field for the specific isotopes ^{92}Mo and ^{126}Ce, any isotopes of calcium, and as much of the reaction formula as possible.

To demonstrate the precision of retrieval the ET field provides, this search was executed in the INSPEC file covering the years 1969 to present as it appears on STN International.

INSPEC now adds a chemical information field (CI or CHI) to their records, which contains the element symbols mentioned in a citation. Each online host has chosen to implement this field differently. For clarity, this field has not been considered for this search example.

CONCLUSION

The element terms field is an access point to chemical information contained in technical databases that do not index their contents from a chemical point of view. The ET field enables users to retrieve more precise answers by enhancing access to chemical information contained in nonchemical databases by making elements, compounds, and reaction notations searchable. This precision is especially useful if the symbol for the element or compound being sought is synonymous with another abbreviation or a real word. Special symbols, especially grouping symbols such as parentheses, are retained in the ET field; in the basic index, these symbols would be replaced with blank characters as if they were a form of punctuation. Reaction or compound expressions containing these symbols can easily be searched by masking the symbols with quotes or single-character truncation. The ET field is available exclusively on STN International and appears in these files: COMPENDEX, ENERGIE, ENERGY, INIS, INSPEC, LPHYS, MATBUS, MEET, METADEX, and PHYS.

Graphics and Natural Language Interface for a Cybernetic Analytical Instrument

HARI GUNASINGHAM* and MUN-LEONG WONG

Department of Chemistry, National University of Singapore, Kent Ridge, Singapore 0511

Received January 12, 1989

The design and implementation of a graphics and natural language interface for a cybernetic analytical instrument are described. A novel feature of the design is the way the symbolic representation of active elements of the experiment is integrated with their functional and procedural characteristics by using a Prolog-based frame formalism. The graphics and natural language interface obviates the need for any programming knowledge on the part of the user. Further, it enables the user to dynamically interact with the experiment in real time.

INTRODUCTION

Recently there has been increasing interest in the concept of a cybernetic analytical instrument that enables the intelligent control of experiments. One of the key characteristics of such instruments that distinguishes them from conventional computer-based instruments is the use of heuristic knowledge to make decisions based on uncertain experimental data.[1]

In an earlier paper[2] we described a design for a cybernetic analytical instrument conceived as a knowledge-based system in closed-loop, real-time interaction with the instrument hardware. The knowledge base is nondeterministic and contains only the logic of the process. In our design, a frame-based formalism was employed. The actual control is directed by the inference engine, which selects and fires the relevant frames at the appropriate time. A key feature of the design is that the inference mechanism, while being deterministic at run time, can be altered by the user to suit the application at hand.

The knowledge-based system was implemented in Prolog. This language has several advantages, arising from its origins in logic, that have no direct analogues in other languages.[3,4] These include declarative programming style, autonomous control of search, independence of logic and control, built-in pattern matching, and modularization.

In our work the version of Prolog used was Turbo Prolog (Borland International). A Turbo Prolog program consists of four essential sections: domains, predicates, database, and

0095-2338/89/1629-0283$01.50/0

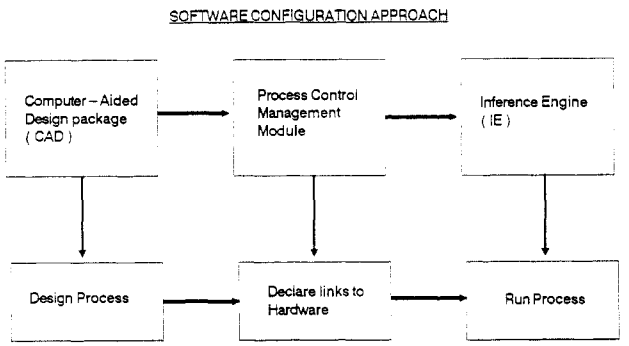

Figure 1. Block diagram showing the way the software system is configured for a particular application.

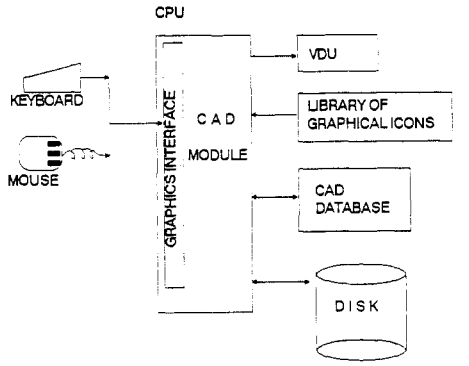

Figure 2. Computer-aided design (CAD) module.

goals. The domain section specifies the types of values that can be used for the objects that are part of each predicate. The predicates section defines the names and relations that are known to exist between objects. The database section contains special predicates that can be added to or subtracted from the knowledge base by using assert(fact) and retract(fact), respectively. The goals section contains control clauses that can direct a solution. Goals can be input from the keyboard or can be made part of the program. Details about Turbo Prolog can be found in the Borland operating manual and several textbooks on the subject.

The emphasis of the work reported in our previous papers[1,2] was on developing the knowledge-based system capabilities such as the structure of the knowledge base and inference engine. The user interface was limited to an interactive menu-driven system with which the user directed the operation of the instrument by means of the keyboard in response to prompts that appeared on the VDU.

Whereas a menu-driven system is often adequate for applications that are simple and well-defined, it is unsatisfactory when they are complex or poorly defined. Also, because the user generally has to configure the instrument to suit a particular application, a menu-driven system can be very restrictive, especially when it is necessary to have an idea of how the experimental setup will perform during the design stage. Finally, instruments function in a dynamic environment that can be conceptually better understood by way of symbols and diagrams.

In this paper we describe the design and implementation of a graphics and natural language interface that enables the user to design and configure external instruments and controllers for a particular experiment without the need to have any programming experience. An important feature of the interface is that the experiment is represented in the form of icons (representing various active elements of the experiment) that can be directly linked to the external instruments and controllers. The interface also provides an interactive environment for the user to interrogate the system to ascertain the status of the experiment in real time.

A further novel aspect is the integration of the graphics and natural language interface with the frame representation scheme of the knowledge base using Prolog.

OVERVIEW OF SOFTWARE DESIGN

The core software is separated into three distinct modules: the computer-aided design (CAD) module, the process control management module (PCM), and the inference engine (IE). Figure 1 is a block diagram showing the functional arrangement of the software modules.

The first step is to design the schematics of the experiment. The CAD module is used for this purpose. Either standard icons can be used (e.g., representing valves, sensors, and devices) or they can be specially drawn.

The second step is to assign the links between the various icons and the actual hardware. This is done by the process control management (PCM) module. The key to the linking process is the use of a frame representation scheme to store the operational characteristics of the specific device or system associated with the icon. These characteristics include procedures (deterministic routines that direct the protocol for the operation of the devices), hardware drivers that actually control the devices, heuristics (rules of thumb that are brought to bear when decisions have to be made in regard to when and how to operate the device), and facts (working parameters). An icon can have more than one frame associated with it. These are all stored in the knowledge base.

The IE module executes the experiment by using the heuristics that are declared in the knowledge base. The IE combines deterministic control with nondeterministic inference, thus affording considerable versatility.

The following sections describe the design and implementation of each module in more detail.

COMPUTER-AIDED DESIGN MODULE

Figure 2 describes the CAD module used for drawing the schematic representation of an experiment. The CAD database is used to store a library of icons. The CAD module also supports the use of the mouse for drawing and designing new icons. To draw an icon on the VDU screen, the mouse is used to point to the icon stored in the library of icons and then to select the position on the VDU where the icon should be placed. The CAD module also provides editing features for the user to alter the position, pattern, color, and size of the icon.

A unique feature of the CAD module is the paging coordinate system, which allows a complex experiment to be separated into smaller modular windows. Each window is referenced by its workstation number and page number. The present design of the graphics interface allows 24 workstations, and each workstation can have up to 24 graphics screen pages. This means that up to 24 × 24 graphics screens can be used to configure several experiments at the same time. Using the paging coordinate system, the user can shift from one window to another by changing either the workstation or page number with the mouse.

When completed, the schematics of the experiment can be saved onto disk storage and referenced by a unique filename. The file can then be retrieved and the diagram updated if any amendments are required.

The icons that make up the experiment schematics are declared in Prolog by using the following general structure:

Domains

```
filename, name =          string
ws, page, r, c, color, fill,   integer
  seq, layer, ind =
drawing =                 p(r,c);
```

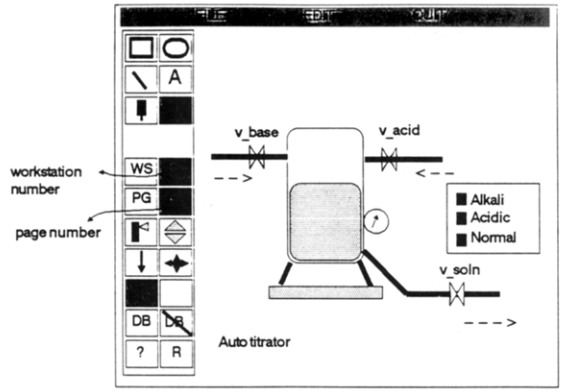

Figure 3. Experiment schematics of window(1,1) relating to the autotitrator application.

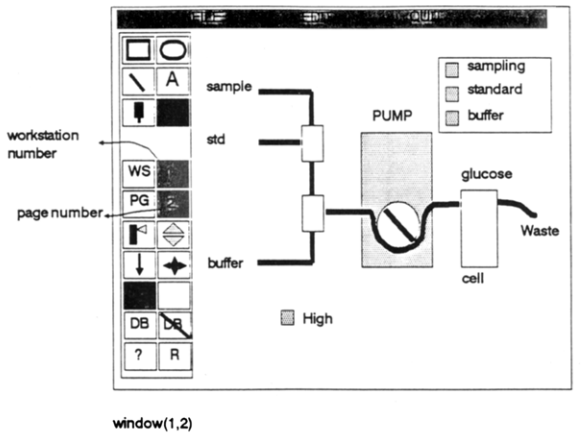

Figure 4. Experiment schematics of window(1,2) relating to the glucose monitoring application.

```
drawings =      drawing*
type =          bx(r,c,r,c,fill,color);
                defn(r,c,r,c,fill,color);
                ruler(r,c,r,c,fill,color);
                text(r,c,r,c,fill,color);
                pen(r,c,r,c,fill,color,drawings);
                polygon(r,c,r,c,fill,color,draw-
                  ings);
```

Database

```
background(ws,page,filename)
obj(ws,page,seq,layer,name,type,ind)
where
ws =            logical workstation number
page =          logical page of workstation
filename =      bit-mapped background image for win-
                  dow store in filename
seq =           unique sequence number
layer =         layering of icon either top or bottom
name =          unique name of object icon
type =          graphics structure of icon
ind =           indicator to represent whether the icon
                  is active being deleted
```

By way of example, Table I is a listing of Turbo Prolog database predicates that implement the process schematics shown in Figures 3 and 4. The process schematics of Figure 3 relate to an autotitrator displayed on window(1,1), whereas Figure 4 relates to a glucose monitor displayed on window(1,2).

DECLARING THE HARDWARE LINKS

The next step in the design of the experiment is to assign the links between the various icons to the hardware interface.

Table I

```
obj(1,1,32,1,normal,text(106,470,8,84,1,0),1)
obj(1,1,30,1,acidic,text(90,470,8,76,1,0),1)
obj(1,1,28,1,alkali,text(75,470,8,70,1,0),1)
obj(1,2,26,1,FIA,system for glucose monitoring,
text(178,108,8,458,3,0),1)
obj(1,2,25,1,high,text(156,296,8,54,3,0),1)
obj(1,2,22,1,buffer,text(39,402,8,86,3,0),1)
obj(1,2,20,1,standard,text(29,404,8,122,3,0),1)
obj(1,2,18,1,sampling,text(19,406,8,106,3,0),1)
obj(1,2,16,1,waste,text(113,542,8,74,3,0),1)
obj(1,2,15,1,glucose,text(76,470,8,94,2,0),1)
obj(1,2,14,1,buffer,text(134,110,8,88,2,0),1)
obj(1,2,13,1,std,text(56,108,8,44,2,0),1)
obj(1,2,12,1,sample,text(23,110,8,82,2,0),1)
obj(1,2,9,1,cell,text(147,458,8,44,1,0),1)
obj(1,1,7,1,→,text(153,432,8,62,1,0),1)
obj(1,1,6,1,←,text(61,408,8,34,1,0),1)
obj(1,1,5,1,→,text(65,124,8,38,1,0),1)
obj(1,1,4,1,v_soln,text(127,428,8,82,1,0),1)
obj(1,1,3,1,v_base,text(41,160,8,90,1,0),1)        ← 1000[a]
obj(1,1,2,1,v_acid,text(38,364,8,84,1,0),1)
obj(1,1,1,1,pH,defn(75,336,22,48,1,0),1)           ← 1001[b]
obj(1,2,11,1,pinch_2,defn(92,234,25,38,1,0),1)
obj(1,2,10,1,pinch_1,defn(49,232,25,38,1,0),1)
obj(1,2,8,1,pump,text(50,324,8,60,1,0),1)
obj(1,2,23,0,box23,bx(15,370,36,172,3,0),1)
obj(1,1,33,0,box33,bx(66,428,55,136,1,0),1)
obj(1,1,27,1,a_alkali,bx(72,442,11,20,3,1),1)      ← 1002[c]
obj(1,1,29,1,a_acidic,bx(87,442,11,20,3,1),1)
obj(1,1,31,1,a_normal,bx(102,442,11,20,3,1),1)
obj(1,2,24,1,high_glucose,bx(155,276,6,12,1,1),1)
obj(1,2,17,1,s_sample,bx(20,386,6,12,1,1),1)
obj(1,2,19,1,s_standard,bx(29,386,6,12,1,1),1)
obj(1,2,21,1,s_buffer,bx(38,386,6,12,1,1),1)
obj(1,1,34,1,temp,text(74,176,8,58,1,0),1)
background(1,1,patent1.pic)                        ← 1003[d]
background(1,2,patent2.pic)
```

[a] Data 1000 expresses the structure of the icon representing the solenoid valve for acid addition. [b] Data 1001 expresses the structure of the icon representing the pH meter. [c] Data 1002 expresses the structure of the icon used to represent a screen-based alarm. [d] Data 1003 expresses the structure of the icon representing the bit-mapped background image used in window(1,1).

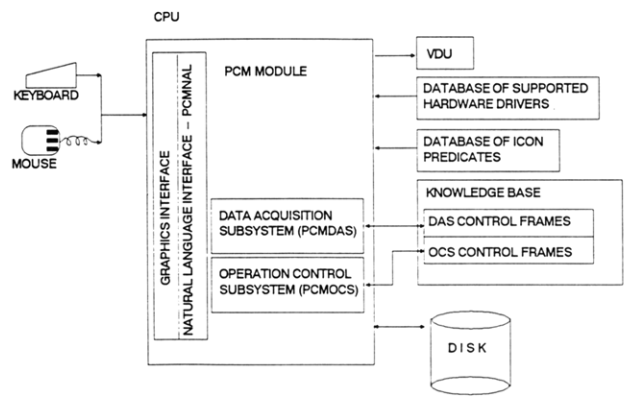

Figure 5. Process control management module (PCM).

This is done by the PCM module.

A more detailed structure of the PCM module is shown in Figure 5. The front end of the module is a graphics interface that allows the selection of icons for configuration and the natural language interface (NAL). The NAL is an English-language parsing subroutine.

Once the icons have been linked to the interface, the user can input heuristics (stated in English-like form) relating to the control of the icons using the NAL. The NAL also checks the validity of the heuristics.

Hardware drivers written in Prolog or Assembly language direct the actual control of external instruments that are

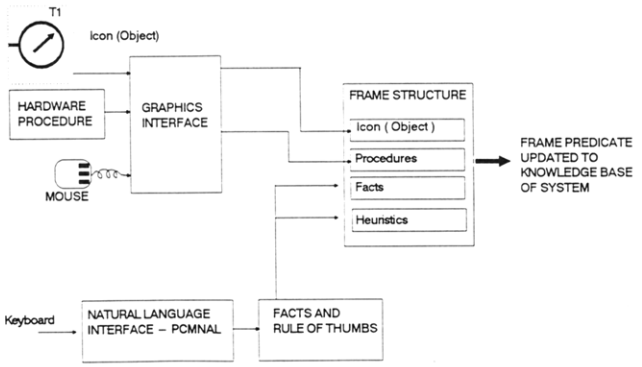

Figure 6. Linking the experiment icons to the hardware by using the PCM module and updating the frame-based knowledge base.

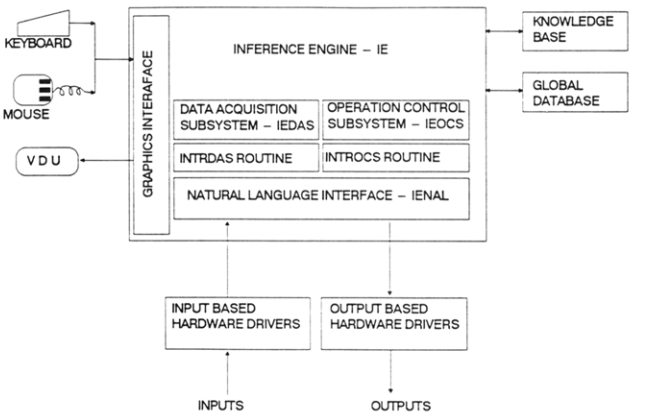

Figure 7. Inference engine.

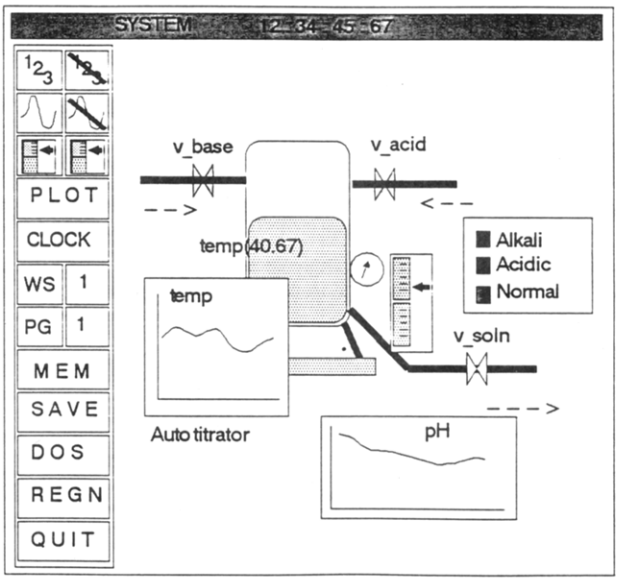

Figure 8. Typical display showing real-time trends and data values.

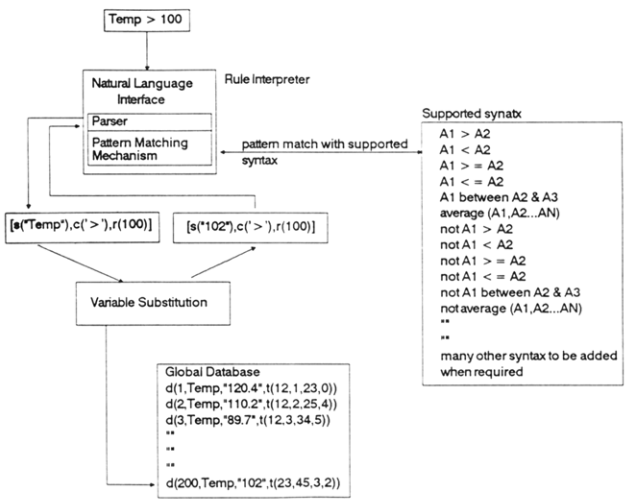

Figure 9. Working of the inference engine natural language interpreter.

connected to the PC. Instruments can be connected through an electronic analog-to-digital interface, serial port, or other connecting devices using the bus of the microcomputer.

Hardware drivers can be either input drivers or output drivers. Input drivers are employed for data acquisition, and output drivers are for control purposes. Thus, in the configuration of the hardware linkages both the input control rules and the output control rules have to be declared.

The PCMDAS is one of two subsystems of the PCM module. It is responsible for configuring data acquisition requirements of the process as well as linking the particular icon representing the input device. The second subsystem, PCMOCS, is similarly responsible for configuring the output control requirements of the process and linking the icon representing the output device. Both the PCMDAS and the PCMOCS establish the input and output configuration and linking functions by creating frames that are stored in the knowledge base.

The procedure for creating a data acquisition frame by use of PCMDAS is shown in Figure 6. The empty frame structure is first created. Then by use of the graphics and NAL interfaces the various slots are filled with the following details: (i) particulars of the input device icon, e.g., analog–digital channel; (ii) procedures relating to input device, the name of one or more subroutines written in Prolog or Assembly language that actually direct the operation of the external device(s); (iii) facts and heuristics relating to the operation of the input device that determine when and how the external device(s) should be used. Once the slots have been filled, the frame is asserted into the knowledge base.

The PCMOCS functions in the same way to configure frames for control or output devices.

The PCMNAL interface is an English-like parsing routine that checks the validity of the English conditions statements such as "pH >7.5" so as to see that it conforms to the permitted language syntax.

After the control rules of the process have been declared, the knowledge base of the PCM module can be saved on disk and can be retrieved and updated when required.

RUNNING THE EXPERIMENT

Once the experiment is configured, the next step is to execute it by use of the inference engine. Figure 7 shows the structure of the inference engine (IE). The inference engine comprises a graphics interface for manipulation of icons, IEDAS for data acquisition and IEOCS for operation control purposes.

The IE graphics interface provides access to the IEOCS and IEDAS. The interface is a subset of the CAD graphics interface. The user can query the IEOCS through the interface and determine the reasoning behind particular control actions. The IEDAS can also be accessed, and plots, trends, bar charts, and data profiles from external sensors can be displayed dynamically on the VDU. A typical display is shown in Figure 8.

Interrupts can be generated through the interface to enable manual overide of the IEDAS and IEOCS operations. INTRDAS and INTROCS are the respective interrupt routines for IEDAS and IEOCS.

Figure 9 describes the IENAL interface in more detail. The IENAL interface includes an English-language parser and a

pattern matcher that allows the IEDAS and IEOCS to analyze and understand the English-like control heuristics of the dasframe and ocsframe predicates. The IENAL parses each sentence and converts its meaning into tokens. It then substitutes variable tokens with real values from a global database. The validity of the control heuristics is then determined by pattern matching.

Apart from rule interpretation, IEDAS and IEOCS also control data acquisition and experimental control, respectively. The slots of the control frames contain structural, attributive, procedural, and heuristic information. Once the IENAL interface determines that the heuristics or firing conditions of the frames are satisfied, it calls a procedural subroutine that executes the necessary control sequences as specified in the procedural slot of the frame.

CONCLUSION

One of the important functions of the graphics and natural language interface is to enable the user who has no knowledge of computer programming to use the knowledge-based software effectively to configure and run experiments. The system also provides the capability for the user to query the experiment or process in real time to ascertain dynamic values of the instruments and controllers, determine trends in the form of graphical plots, and learn why a particular control action was performed.

ACKNOWLEDGMENT

The provision of a grant from the International Development and Research Center (IDRC), Ottawa, Canada, is gratefully acknowledged.

REFERENCES AND NOTES

(1) Gunasingham, H. *J. Chem. Inf. Comput. Sci.* **1986**, *26*, 130.
(2) Gunasingham, H.; Wong, M. L. *J. Chem. Inf. Comput. Sci.* (in press).
(3) Kowalski, R. A. *Logic for Problem Solving*; North-Holland Elsevier: New York, 1979.
(4) Bratko, Ivan. *Prolog Programming for Artificial Intelligence*; Addison-Wesley: Reading, MA, 1986.

COMPUTER SOFTWARE REVIEWS

Asystant GPIB

MATEJ PENCA
USDA, ARS, BARC-W, Beltsville, Maryland 20705-2350

Received September 6, 1989

The Asystant GPIB from Asyst Software Technologies, Inc., is mainly intended for users in PC environments who would like to collect, manipulate, analyze, and graphically display their scientific and engineering data. Asystant GPIB enables users to acquire data from instruments controlled through the GPIB. The GPIB and Analysis features together provide numerical and statistical analysis, input and output to files, and more sophisticated graphics capabilities.

The maximum use of Asystant GPIB can be gained from a knowledge and understanding of IEEE-488 manuals and both Asystant Tutorial and Reference manuals. This software package allows users to communicate with instruments through GPIB commands. There are two ways that the commands can be executed, either immediately (Interactive Mode) or in batch for later programmable routines (Program Mode).

Asystant GPIB comes with excellent documentation, which includes Reference and Tutorial manuals. In the Tutorial manual there are clear explanations of Asystant's features with step by step instructions, enriched with self-explanatory examples.

The user of this software package will probably get familiar with the use of Asystant very quickly as the Analysis Tutorial provides seven different demonstration sessions that show how to use various features of Asystant. These sessions look so easy that one can get the impression that working with Asystant is easier than it actually is.

When users start working on their own with Asystant GPIB, they may face problems that have not been considered. In that case there is an online help as an option in the main menu, which gives a lot of information about options and features available on this software. In the cases where the problems are complicated and cannot be resolved simply by reading the manual or online help, Asyst Software Technologies, Inc., provides a Technical Support Hotline free of charge to new owners of Asystant GPIB for 60 days from date of purchase. After the 60-day period, customers may enroll in the Extended Support Plan that includes ongoing Hotline support and many other benefits.

Asystant GPIB also has nicely organized menus. In the main opening menu, besides the Main Menu options there are also Calculator Functions, Stack Contents, and Parameters and Variables presently used. If one is learning how to work with Asystant GPIB, I suggest selecting the Help option and reading about different Asystant options because all of the secondary menus (menus for each of the main options) and a short description for each of them will be given. Each secondary menu has a Return option that takes the user back to the Main Menu from which one can either select another option, look for online help with the Help option, or exit Asystant GPIB by selecting the Save/Exit option. If the user wants to display on the screen or plot anything, the user needs to go to the Main Menu and select the Graphics option. This option changes the screen to high resolution, which has more points per square inch and thus makes the graphs on the screen more accurate than they would be on a normal resolution screen. Before plotting, the user must set variables and parameters into Y and X values which are needed to graph the particular function. After the user has finished experimenting with the graph and the way it is presented, the graph can either by printed on the printer, plotted on the plotter, and/or dis-

0095-2338/89/1629-0287$01.50/0