Deterministic Control in Knowledge-Based Systems: Application to the Development of a Cybernetic Analytical Instrument

HARI GUNASINGHAM* and M. L. WONG

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

Received October 15, 1987

Cybernetic instruments combine real-time control with decision-making and interpretative capabilities. The implementation of a cybernetic instrument based on a knowledge-based system is described. A novel feature of the design is the integration of deterministic control with nondeterministic inference for the actual control of a process or experiment. Use is made of a frame representation scheme to link procedural and declarative knowledge. An application to pH control is discussed.

INTRODUCTION

Recently there has been considerable interest in the development of intelligent systems for real-time control applications such as in laboratory automation and process control. Of particular interest are systems that have the ability to alter the control of an experiment or process in response to dynamic changes in conditions through judgmental decision making based on uncertain or incomplete data. Such systems have been termed cybernetic in reference to their feedback-oriented, self-adaptive capability and in the way they seek to emulate "human-like performance".[1]

Programming techniques for artificial intelligence applications can be broadly classified as being heuristic or deterministic. The terms refer to both the program structure and the approach to problem solving. The heuristic approach holds "expert" knowledge in the form of rules-of-thumb (heuristics) as the key to problem solving. Writing a heuristic program involves declaring the rules and facts (the knowledge base) describing the problem domain. The control procedure for obtaining a solution to a problem is, however, obtained through an externally derived inference mechanism that is usually nondeterministic.[2]

A deterministic program is, in contrast, rigidly defined and highly procedural. In general, it is written in an imperative language such as Pascal or assembly language that sets out, step by step, exactly how a problem should be solved. Thus, the knowledge and inference are usually not separated.

In a previous paper we discussed the design of a cybernetic instrument, conceived as a knowledge-based (K-B) system in closed-loop interaction with the experimental control functions.[3] The functional parts of the instrument were partitioned into plan, execute, analyze, and refine phases. The plan and refine phases were considered to be heuristic; the execute phase considered to be deterministic and the analyze phase both heuristic and deterministic.

Figure 1 describes the organization of heuristic and deterministic parts of a cybernetic instrument used to control an experiment. The deterministic Pascal program directs the execution of the experiment (the execute phase), whereas the heuristic K-B system directs the plane, analyze, and refine phases.

Although the above structure is adequate from a functional standpoint, after some work with it, we felt it was more natural, consistent, and efficient to integrate the heuristic and deterministic parts into the framework of a single K-B system. Perhaps the most important benefit is the greater facility to implicitly effect changes in the flow of control by altering the rules in the knowledge base. In the earlier approach, where the experimental control program is logically separated from the heuristic K-B system, changes to the flow of control must be explicitly specified in the program. Another advantage of a K-B system is that it affords a higher level of abstraction in describing real-world systems.

This paper describes the design and implementation of a cybernetic instrument that can be configured to work in many application domains and is suitable for both laboratory automation and process control. The K-B system is written in Turbo Prolog, while the actual experimental control procedures are written in assembly language. The two are integrated in the K-B system through a frame representation scheme. An application to pH titrations is described as an illustration of the effectiveness of the system.

TURBO PROLOG

Prolog was first developed by Colmerauer and Kowalski[4] over a decade ago as a symbolic logic programming language based on first-order predicate calculus. The language exclusively deals with objects and relations between objects (predicates). Unlike conventional language such as Pascal, programming involves describing the logical structure of the problem domain by declaring the known objects and predicates regarding the domain. Prolog is its own inference engine in that it automatically deduces a solution from the declared predicates through a nondeterministic search.

Despite early reservations in regard to the general applicability of logic-based languages for developing large K-B systems, they are being increasingly used for this purpose. Recently, interest has been kindled by the Japanese fifth-generation project[4] and the availability of new implementations that get round some of the difficulties of the language.

We have been using the Borland Turbo Prolog,[6] which, while including the essential features of the original Closkin and Mellish implementation of Prolog, has some additional capabilities.[7] It has, for example, a greatly extended capability in I/O handling and numerical processing. It also has a natural interface to languages such as Pascal and assembly language as well as an efficient, window-oriented user interface and advanced editing features.

A Turbo Prolog program consists of four parts: names and structures of objects involved in the problem (domains); names and relations that are known to exist between the objects (predicates); facts and rules describing their relations (knowledge-base clauses); and goals that direct a solution (control clauses). Goals can be input from the keyboard or can be made part of the program.

Turbo Prolog facilitates the modification of the inference mechanism through the use of the control clauses to provide the control knowledge, dictating the order of a Prolog search of the knowledge base. Although they are not part of the knowledge base proper, it is of interest that the control clauses themselves are not committed to any inference mechanism and can be searched to afford the appropriate mechanism to suit

0095-2338/88/1628-0211$01.50/0

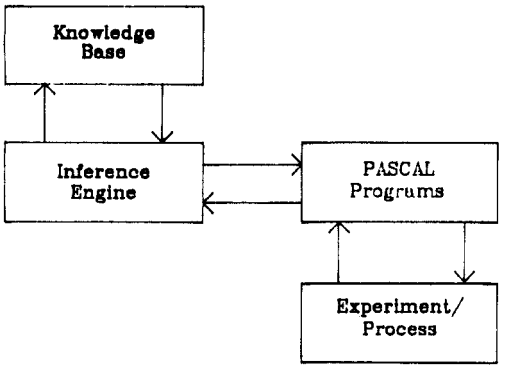

Figure 1.

the problem at hand. The ability to explicitly structure the Turbo Prolog inference mechanism obviates the difficulty of having to reconstruct the correct procedural constraints between scattered fragments of knowledge through a nondeterministic search. The modified inference mechanism thus affords enhanced efficiency, which is translated into better real-time response.

FRAMES

The use of procedural knowledge in rule-based (R-B) systems has been addressed by several workers.[8,9] The approach of Georgeff and Bonollo is to use two knowledge bases: a rule base and an event-graph base.[8] The former is used to represent the declared knowledge regarding the problem domain, whereas the latter represents procedural knowledge through event graphs.

In our approach, we have used the frame structure to represent procedural information within a R-B system. One of the important characteristics of frames is the ability to decide whether it is applicable in a particular situation. This enables considerable gains in efficiency.

A frame is basically a slot and filler representation that can be used to describe an object, concept, or situation.[10] As a slot can also contain procedures, the frame structure allows the integration of procedural and declarative information.

In the present implementation, a frame is used to describe a process controller or analytical instrument. Included in the frame are the operational characteristics of these systems such as the input and output forms, optimized settings for a particular application, and procedural information, namely, the sequence of operations that the instrument must perform in carrying out an experiment or controlling a process.

The general structure of an instrument or controller frame is as follows:

slot	
TYPE	type of operation, e.g., control, transducer input
NAME	name of instrument, e.g., pHmeter
SUBROUTINE	procedural subroutine pointer
ATTRIBUTES	list of attributes, e.g., calibration, voltage, min, max

The procedural subroutine pointer directs a call to a Pascal or assembly language based subroutine that performs the actual control operation.

For example, a frame describing the characteristics of a thermistor can be as follows:

slot	value
TYPE	transducer
NAME	thermistor
SUBR	thermsubr
ATTRIBUTES	channel = 0, min value ; 0, max value = 70, calibration points: (1) 63° = −1775 mV; (2) 51° = −1536 mV; (3) 41° = −1493 mV; (4) 31° = −1227 mV; (5) 26° = −1113 mV; (6) 10° = −427 mV; (7) 0° = 207 mV

The above frame represents a thermistor. It has a control procedure pointed to by the subroutine pointer, thermsubr. The attribute values are the declared characteristic parameters of the instrument used by the subroutine. To execute the frame for the control of an instrument, the attribute values must be bound. Then the subroutine is called, and it directs the actual operation of the thermistor with the bound attribute values. The values could be default or they could be input by the user via the keyboard or updated heuristically. The attributes of the thermistor are the channel number of the analog-to-digital converter, the range of the thermistor, and the calibration points that can be used to determine the actual temperature. The calibration points are determined experimentally and can be input manually via the keyboard. Alternatively, the system can run a calibration procedure that automatically captures the experimental points during the calibration run.

IMPLEMENTATION OF FRAMES IN PROLOG

Frames are naturally implemented in Prolog. The general form of an instrument frame can be represented by a predicate

```
frame(NO,TYPE,NAME,SUBR_NAME,ATTRIBUTES)
```

The objects appearing in the predicate are variables that have to be bound either by default, by the user, or heuristically via the knowledge base. SUBR_NAME is a global object that links the frame to an assembly language routine.

For example, the default frame for the thermistor is given by

```
frame(53, transducer, thermistor, thermsubr,
      [v(adc ch,0), v(min,0), v(max,70), c(7,0,207),
       c(6,10,−427), c(5,26,−1113), c(4,31,−1227),
            c(3,41,−1493), c(2,51,−1536), c(1,63,−1175)])
```

$c(x,y,z)$ represents the calibration points, where x is the number of the calibration point, y is the temperature in degrees centigrade, and z is the equivalent voltage value from the thermistor probe.

IMPLEMENTATION

The general performance objectives for the cybernetic instrument include the following: automatic configuration of hardware and control parameters for the control of the particular experiment or process of interest; integration of deterministic and heuristic control, thus providing a flexible and highly adaptive control system; and that the user be aware only of the heuristic knowledge.

HARDWARE

At present the experiment or process is linked to the K-B system through an interface consisting of an 8-channel, 12-bit analog-to-digital converter, 4 channel, 8-bit digital-to-analog converters, and 32 digital input/output lines for control purposes. The interface is general purpose in that it is suitable for most processes or experimental applications. One of the functions of the K-B system is to configure the appropriate analog and digital input/output lines to suit a particular application. Often, the integrated control of several external devices may be required.

As stated earlier, the actual control of the devices is through assembly language subroutines that are called from the Prolog main program. The configuration information for the interface (e.g., channel number, input voltage range, real-time clock frequency, and the digital input/output line selection) is,

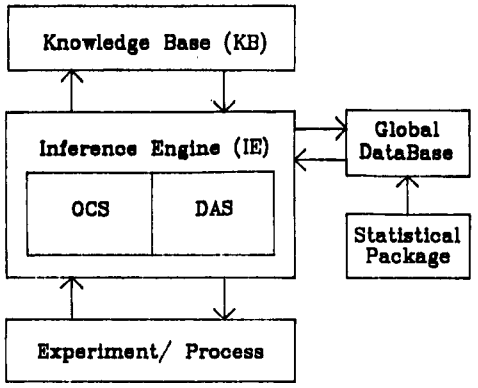

Figure 2.

however, declared in the instrument frames, and this is used by the subroutines.

For the pH control experiment described later in this paper, the ADC channel, input voltage range, and digital output lines (for the control of acid or base addition) have to be configured.

K-B SYSTEM

The architecture of the cybernetic instrument software consists of the inference engine and knowledge base. The inference engine consists of the data acquisition subsystem (DAS), which controls the acquisition of data from the experiment or process, and the operation control system (OCS), which directs the actual control of the experiment or process (both analog and digital control).

Figure 2 gives the organization of the K-B system. A separate data base is employed to store experimental data in symbolic form by using the Prolog "assert" predicate, and a statistical package is used to transform the data before it is added to the knowledge base. The statistical package can also be used to assess the analytical data to evaluate trends that can then aid in the decision-making process for effecting the control operations.

KNOWLEDGE-BASE REPRESENTATION

The knowledge of the expert system consists of three types: (i) Knowledge regarding the configuration of the data acquisition subsystem, which includes settings of the a/d converter and the I/O controller, is declarative and is represented in a low-level Prolog predicate form. For example, the predicate gol(1,instrumentation,pHstat,[2,6]) configures the interface for a single analog input and a graphic display mode, described by frames 2 and 6, respectively. (ii) The frame structure represents the working characteristics of the various instruments and controllers. Here, the characteristics include procedural knowledge. The frame structure has been discussed in earlier sections. (iii) If-then rules are used for deciding the actual control of a process or experiment. These rules are a formal part of the knowledge base (unlike the control rules). The if-then structure used by the rules incorporates the following features: (i) a message that describes the function of the rule; (ii) slots for storing control frames to be executed when the rule is fired; and (iii) a list of conditions that must be satisfied for firing the rule. The rules have the general form

```
rule(NO,NAME,"MESSAGE",FRAMES,
                                CONDITIONS)
```

For example, a specific rule for determining the addition of base in the operation of a pHstate is given by

```
rule(12,pHstat,"0.4 cc base",[77],[1,2,3,4,5])
cond(1,"pH>4")
cond(2,"temperature between 30 and 50")
cond(3,"pH<7")
cond(4,"valve1 is on")
cond(5,"valve2 is off")
```

The above rule states "add 0.4 cc of base using control frame [77] if conditions 1, 2, 3, 4, and 5 are satisfied". The conditions are stated in English-like phrases that have to be interpreted by a specially designed parser. [77] refers to the frame describing the operation of the solenoid-actuated valves.

INFERENCE ENGINE

The inference engine of a R-B system has two functions: (i) inference, directing the matching and unification of rules, and (ii) control, determining the order in which rules are tested and the consequences when a rule succeeds or fails.

As stated earlier, by use of Turbo Prolog, the inference mechanism can be explicitly defined to suit the problem at hand. For the cybernetic instrument, a generalized control strategy is adopted described by the following Prolog rule:

```
inference_engine:-

    repeat,
    gol(NO,TYPE,NAME,DEVICES),
    das (DEVICES),
    ocs (NAME),

    fail.
```

The user supplies the goal condition via the keyboard or a mouse in response to prompts. The goal condition includes the transducers to be used, the signal modification required, and the instrumentation required. From this goal, the system automatically sets up the data acquisition and control procedures.

For example, on the basis of user input, a goal frame is loaded into the knowledge base that states that an instrumentation system is to be configured to serve the function of a pHstat.

The goal frame is passed to a specially built frame processing procedure defined by the predicate das(DEVICES), which recursively processes each frame. DEVICES is a list of frame reference numbers. In the case of the pHstate the numbers refer to the pH meter and thermistor frames. The Prolog implementation of this recursive processing is given by

```
das([]):-!
das([FRAME |R]):-
   frame(FRAME,TYPE,NAME,SUBR_NAME,AT-
TRIBUTES),
   subr(TYPE,NAME,SUBR_NAME,ATTRIBUTES),
   das(R).
```

Next, the DAS interprets the data-handling requirements through the frames stated in the goal frame and initiates the data handling by calling the appropriate assembly language routines through a global predicate defined by SUBR_NAME.

For example, the global predicate thermsubr (ATTRIBUTES,TEMP) defines the call to an assembly language subroutine that controls the operation of a thermistor. The subroutine is called with the ATTRIBUTE values defining the operation of the thermistor and returns with the latest temperature reading in ASCII form. The data are asserted in the knowledge base as data-base clauses.

Once the data acquisition and monitoring requirements have been established, the control passes to the operation control system (OCS). OCS determines what control actions should be performed through a nondeterministic search of the if-then rules. The search is driven by the data acquired from the experiment/process. As such the actuation of control depends on the rule conditions that are satisfied. The search is nar-

Chart I

```
frame(1,control,delay,[v(delay,300)])
frame(2,alarm,red,[v(beep,102)])
frame(3,alarm,pink,[v(beep,102)])
frame(4,alarm,green,[v(beep,102)])
frame(5,alarm,yellow,[v(beep,102)])

frame(20,control,solenoid0,[v(add_rate,1)]) /# solenoid0 assign for
  control #/
frame(21,control,solenoid0,[v(add_rate,3)]) /# of acid #/
frame(22,control,solenoid0,[v(add_rate,5])
frame(23,control,solenoid0,[v(add_rate,10])
frame(24,control,solenoid0,[v(add_rate,30])

frame(30,control,solenoid1,[v(add_rate,1)]) /# solenoid1 assign for
  control #/
frame(31,control,solenoid1,[v(add_rate,3)]) /# of base #/
frame(32,control,solenoid1,[v(add_rate,5])
frame(33,control,solenoid1,[v(add_rate,10])
frame(34,control,solenoid1,[v(add_rate,30])

cond(1,anticipated alkaline)
cond(2,solution is acidic)
cond(3,solution is alkaline)
cond(4,solution is neutral)
cond(5,pH >11)
cond(6,pH between 9 & 11)
cond(7,pH between 8 & 9)
cond(8,pH between 7.5 & 8)
cond(10,anticipated acidic)
cond(11,temperature >50)
cond(12,pH <3.0)
cond(14,pH between 3 and 5)
cond(15,pH between 5 and 6)
cond(16,pH between 6 and 6.5)

rule(1,pHstat,actuate acid solenoid with 45 pulses,[22,23,24],[5])
rule(2,pHstat,actuate acid solenoid with 20 pulses8[23,23],[6])
rule(3,pHstat,actuate acid solenoid with 10 pulses,[23],[7])
rule(4,pHstat,actuate acid solenoid with 3 pulses,[21],[8])

rule(5,pHstat,actuate base solenoid with 45 pulses,[32,33,34],[12])
rule(6,pHstat,actuate base solenoid with 20 pulses,[33,33],[14])
rule(7,pHstat,actuate base solenoid with 10 pulses,[33],[15])
rule(8,pHstat,actuate base solenoid with 3 pulses,[31],[16])

rule(10,pHstat,actuate red alarm—acidic,[2],[2,10])
rule(11,pHstat,actuate pink alarm—neutral,[3],[8,16])
rule(12,pHstat,actuate green alarm—alkaline,[4],[1,3])
rule(13,pHstat,actuate yellow alarm—temperature high,[5],[11])
```

rowed by the fact that only those rules with the NAME specified by the goal are searched.

The OCS employs a natural language interface to check the status of the symbolically stated conditions (e.g., pH greater than 11, 3 less than pH less than 11) using a specially defined parser.

RESULTS: OPERATION OF A PHSTAT

The function of a pHstat is to maintain the pH of a process at a fixed point by the addition of acid or base. In conventional systems, pH control is usually maintained by feedback control based on deviation from a set point. A key characteristic of the control is that it is predetermined and rigidly defined on the basis of a model of the process.

In the case of the cybernetic instrument, pH control is determined by the if-then rules. The goal is to achieve the reference pH value as fast as possible without impairing the quality of the control. Control is effected through a nondeterministic search of the rules. The search is driven by the input experimental data. When actuated, the rules direct the addition of acid or base, the amount that should be added, and at what rate.

The external devices used for a pHstat include a pH meter, a thermistor, and two solenoid-actuated valves for the addition of acid or base.

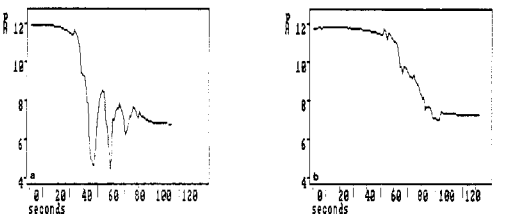

Figure 3.

Chart II

```
frame(1,control,delay,[v(delay,300)])
frame(2,alarm,red,[v(beep,102)])
frame(3,alarm,pink,[v(beep,102)])
frame(4,alarm,green,[v(beep,102)])
frame(5,alarm,yellow,[v(beep,102)])

frame(20,control,solenoid0,[v(add_rate,1)]) /# solenoid0 assign for
  control #/
frame(21,control,solenoid0,[v(add_rate,3)]) /# of acid #/
frame(22,control,solenoid0,[v(add_rate,5])
frame(23,control,solenoid0,[v(add_rate,10])
frame(24,control,solenoid0,[v(add_rate,30])

frame(30,control,solenoid1,[v(add_rate,1)]) /# solenoid1 assign for
  control #/
frame(31,control,solenoid1,[v(add_rate,3)]) /# of base #/
frame(32,control,solenoid1,[v(add_rate,5])
frame(33,control,solenoid1,[v(add_rate,10])
frame(34,control,solenoid1,[v(add_rate,30])

cond(1,anticipated alkaline)
cond(2,solution is acidic)
cond(3,solution is alkaline)
cond(4,solution is neutral)
cond(5,pH >11)
cond(6,pH between 9 & 11)
cond(7,pH between 8 & 9)
cond(8,pH between 7.5 & 8)
cond(10,anticipated acidic)
cond(11,temperature >50)
cond(12,pH <3.0)
cond(14,pH between 3 and 5)
cond(15,pH between 5 and 6)
cond(16,pH between 6 and 6.5)

rule(1,pHstat,actuate acid solenoid with 45 pulses,[22,23,24],[5])
rule(4,pHstat,actuate acid solenoid with 3 pulses,[21],[5])
rule(5,pHstat,actuate acid solenoid with 3 pulses,[21],[6])
rule(6,pHstat,actuate acid solenoid with 3 pulses,[21],[7])
rule(7,pHstat,delay titration,[1,1,1],[7])
rule(8,pHstat,delay titration,[1,1,1],[8])

rule(1,pHstat,actuate base solenoid with 45 pulses,[32,33,34],[12])
rule(4,pHstat,actuate base solenoid with 3 pulses,[31],[14])
rule(5,pHstat,actuate base solenoid with 3 pulses,[31],[15])
rule(6,pHstat,actuate base solenoid with 3 pulses,[31],[16])
rule(7,pHstat,delay titration,[1,1,1],[15])
rule(8,pHstat,delay titration,[1,1,1],[16])

rule(20,pHstat,actuate red alarm—acidic,[2],[2,10])
rule(21,pHstat,actuate pink alarm—neutral,[3],[8,16])
rule(22,pHstat,actuate green alarm—alkaline,[4],[1,3])
rule(23,pHstat,actuate yellow alarm—temperature high,[5],[11])
```

Figure 3a gives the experimental response where the rules (Chart I) for the addition of acid or base are not optimized. Here the heuristic feedback control results in large overshoots in the response because the addition of acid or base is too large. Figure 3b gives the curve where the rules employed (Chart II) have been altered to smooth the response. The rules now allow a finer control of acid or base addition.

A key feature of the rule-based control is the fact that it is not rigidly defined. The control can be modified by changing the rules. As the results show, proper operation of a cybernetic instrument depends on the quality of the rules. Whereas in the above example the rules were manually changed to achieve

this, it is conceivable that it could be done automatically by using the Prolog system. This, in effect, points to the prospects for systems that can learn. Future work will seek to develop this aspect.

ACKNOWLEDGMENT

We gratefully acknowledge a generous grant from the International Development Research Center, Ottawa, Canada, which enabled this work.

REFERENCES

(1) Faulkner, L. R.; Eklund, J. A. *Proceedings of the Electroanalytical Symposium*; BAS: Chicago, 1985; p 225.
(2) Nau, D. S. "Expert Computer Systems". *Computer* **1983**, *15*, 63-84.
(3) Gunasingham, H. "Heuristic Approaches to the Design of a Cybernetic Electroanalytical Instrument". *J. Chem. Inf. Comput. Sci.* **1986**, *26*, 130-134.
(4) Kowalski, R. "Logic Programming". In *Information Processing 83*; Mason, R. E. A., Ed.; Elsevier: New York, 1983.
(5) Fuchi, K. "The Direction the FGCS Project Will Take". *N. Gener. Comput.* **1983**, *1*, 3-9.
(6) *Turbo Prolog Owner's Handbook*; Borland International: CA, 1986.
(7) Clocksin, W. F.; Mellish, C. S. *Programming in Prolog*; Springer-Verlag: Berlin, 1981.
(8) Georgeff, M. P. "Procedural Control in Production Systems". *Artif. Intelligence* **1982**, *18*, 175-201.
(9) Georgeff, M. P.; Bonollou, U. "Procedural Expert Systems"; *Proceedings of the 8th International Conference on Artificial Intelligence*; Karlsruhe, FRG, 1983; pp 151-157.
(10) Barr, A.; Fegenbaum, E. *Handbook of Artificial Intelligence*; William Kaufman: Los Altos, CA, 1982; Vol. 1.

TORTS: An Expert System for Temporal Optimization of Robotic Procedures

T. L. ISENHOUR

Department of Chemistry, Kansas State University, Manhattan, Kansas 66506

P. B. HARRINGTON*

Department of Chemistry and Geochemistry, Colorado School of Mines, Golden, Colorado 80401

Received September 9, 1987

The Temporal Optimizer of Robotic Task Sequences (TORTS) expert system has been devised as a programming aid and a precursor to the merging of laboratory robotics and artificial intelligence. This program predicts the feasibility and run times of various robotic program configurations rapidly, without requiring the robotic system. The TORTS system can evaluate different sequences of robotic tasks seeking to minimize the procedure completion time and efficiently allocate resources.

INTRODUCTION

Most laboratory procedures may be decomposed into a series of tasks. Each task requires certain resources, reagents, equipment, instruments, or personnel. The order in which the tasks are completed is important, because it will effect the overall completion time of the procedure. Scheduling, queuing, and network theories have been devised to maximize the efficiency of one or more procedures by sequencing tasks and allocating resources.[1-3]

Programming robotic systems to run laboratory procedures efficiently is often both tedious and time-consuming. Furthermore, once an operating robot program is obtained, the programmer is often reluctant to modify it. Variable laboratory procedures alter their flow on the basis of tests acquired during run time. For variable or one-time procedures, the time required to program laboratory robotic systems may exceed the time required for the chemist to do the procedure manually. For this reason current robot applications in the laboratory are usually limited to nonvarying procedures repetitiously applied to a large number of samples.

Most laboratory procedures are often variable and are applied to a limited number of samples. Variable robotic procedures would be feasible if efficient robot programs could be rapidly developed. The Temporal Optimizer of Robotic Task Sequences (TORTS) expert system has been developed as a programming aid for versatile and facile robot program development.[4,5] An advantage of the TORTS system is that a programmer can rapidly evaluate a robotic program configuration on a computer without requiring the robotic system. For variable robotic programs each procedure may be optimized separately.

The TORTS system is an expert system devised to maximize the productivity of laboratory procedures, shorten robotic program development times, and acquire knowledge of robotic systems. This system is an integral step in the development of the Analytical Director project.[6]

Chart I

```
REM
REM*** GET A TUBE
REM
  RACK.INDEX=1
  GOSUB GETTUBE.SUB
REM
REM*** FILL TUBE WITH REAGENT
REM
  SYR.N="C"
  SOL.VOL=3.0
  GOSUB ADDSOL.SUB
REM
REM*** MIX SOLUTION
REM
  MIXTIME=120
  GOSUB MIX.SUB
REM
```

ROBOTIC SYSTEM PARADIGM

The optimization of complex robotic procedures is a formidable problem. The TORTS system allows the programmer to simulate the operation of a laboratory robotic system rapidly on a computer. The TORTS system will generate robot task schedules. A schedule contains the predicted starting and completion times for each task in a procedure. Many tasks schedules may be evaluated because computer simulation is much faster than actually running and timing robotic procedures. The best task schedule may be defined as the one producing the shortest project completion time. However, cases may arise where other constraints must be met at the

0095-2338/88/1628-0215$01.50/0