BEYOND OMNIPOTENT ROBOTS

Gary G. Hendrix

Technical Report NL 14

March 1973

Natural Language Research for CAI

Supported by

The National Science Foundation

Grant GJ 509X

The Department of Computer Sciences

and CAI Laboratory

The University of Texas

Austin, Texas 78712

ABSTRACT

A new methodology for the construction of world models is presented. The central feature of this methodology is a mechanism which makes possible the modeling of 1) simultaneous, interactive processes, 2) processes characterized by a continuum of gradual change, 3) involuntarily activated processes (such as the growing of grass) and 4) time as a continuous phenomenon.

BEYOND OMNIPOTENT ROBOTS

1.0 Introduction: The omnipotent robots

Several artificially intelligent entities have been constructed in the last few years which employ what might be called "robot minds." A robot mind is an artificial system capable of some (perhaps very limited) understanding of the world. Examples of such entities include the STRIPS robot (2), the Siklossy-Dreussi robot (6), the Winograd system (9) and the Hendrix question answerer (3).

In some form, all of these systems contain a "robot" which uses a model of the world to do reasoning. In each robot's model of the world there are a collection of objects, a collection of relationships between objects and a collection of operators. The operators describe how objects and relationships between objects may be changed. In these systems, the robot is a necessary participant in each operation. Since the robot may pick and choose the operations in which it wishes to participate, it has complete control over all changes in objects and relationships between objects.

Such omnipotence on the part of the robot has allowed the systems under discussion to conveniently model certain micro-worlds. However, in a world containing many intelligent entities, each entity may exercise its own will and no one entity need be omnipotent. Further, there exists a multitude of processes (e.g., the melting of a cube of ice, the growing of the grass, the falling of a rain drop) which transpire without the aid of robots or humanoids.

If an intelligent system is ever to understand a dynamic world containing numerous intelligent entities and a variety of simultaneous processes, the omnipotent approach to modeling must be abandoned. The robot must understand that its own actions are only a part of a larger whole. To do this, it seems reasonable for the robot to adopt the philosophy that the world is a collection of on-going processes. Some of the processes are controlled by the robot, some are controlled by other intelligent entities and others are simply natural phenomena. With the passage of time, old processes are brought to completion and new processes are initiated. At all times, objects and relationships between objects are defined by (and have their meaning through) the processes which sustain or transform them from instant to instant.

2.0 The post omnipotent robot

In harmony with the philosophy just presented, it seems necessary to provide a robot (or other intelligent entity) with a mechanism for the basic understanding of processes. Such a mechanism must surmount such fundamental problems as the modeling of time, the modeling of continuous, gradual change and the modeling of multiple, interactive processes. Further, the mechanism must not view the robot as the omnipotent master of the world, but must consider all processes on an equal basis.

In the opinion of the author, a central component of such a mechanism is a process monitor which maintains a set of process models and some concept of time. This monitor is the part of the robot's brain which allows the robot to think about simultaneous processes.

At any moment in actual (real world) time, the monitor has a set of process models describing the simultaneous processes occurring in the model world at some particular model time. With model time frozen, the monitor may (sequentially) ponder each process modeled by members of the set of process models. Through this examination, the monitor may determine the earliest moment (in model time) at which one of the processes will "do something interesting." The "something interesting" is usually the causing of some effect which will in turn cause new processes to be initiated, old processes to be terminated, or goals to be met. To model the propagation of the various processes, the monitor increments the model time to the "interesting time" and creates a new model world (or modifies the old world) to reflect changes which would have occurred in the time interval. Then the monitor looks for the next interesting time and the cycle repeats.

In order to add some substance to the discussion above, a more complete description of a particular multi-process model will be given shortly. It is important to emphasize that this model is only a rough first attempt to design the more general concept of a robot which views the world as a collection of processes. The model borrows freely from its omnipotent robot predecessors, especially from STRIPS. To put the new mechanism into perspective, it will be helpful to first review the systems supporting omnipotent robots and to reconsider the very nature of processes.

3.0 Omnipotent robot systems

(Readers familiar with STRIPS may skip this section.)

The omnipotent robot systems attack the modeling of the world by dividing knowledge of the world into two categories: process knowledge and state knowledge. State knowledge relates to knowledge about the world at certain instants in time. Knowledge relating to a particular instant is represented by a state of the world model (SWM). An SWM is like a still photograph of a dynamic situation, representing objects and relationships among objects as they exist at the moment the photograph is taken. Process knowledge is a body of information describing how one state may be transformed into another. By combining these two types of knowledge, a space of states of the world is defined. Connections between points in the state space correspond to processes which take one state into another.

The SWM used to represent an omnipotent robot's state knowledge is typically a set of well-formed formulas in the first order predicate calculus which are true for the world state being modeled. However, objects and relationships between objects may just as easily be depicted by other means. (For example, by semantic nets. Simmons (7).) Process knowledge is represented by a collection of operators which model processes in the real world. These operators may either be programs (Winograd (9)) or data structures which may be interpreted as programs (STRIPS (2), Hendrix (3)), or both (Siklossy and Dreussi (6)). (The line between program and data structure is often thin and arbitrary.)

Each operator is characterized by two principal components. The first component is a set of (sometimes explicit) preconditions which

must hold in the state of the world before the operator may be applied. Typically, the preconditions are a collection of well-formed formulas containing variables which partially specify a state of the world. This partial state specification defines a family of world states comprising the domain (or the context) of the operator. The second component characterizing an operator is a set of effects produced by the operation. Given an SWM s_1 depicting a real world state S_1 in the domain of some operator, the operator's effects define a new (perhaps partially specified) SWM s_2 which depicts a world state S_2 that would result if the process modeled by the operator were performed on the world state S_1. Some decision process (e.g., means-end analysis in the case of robots, or the input sentence in question answering) dictates a particular order in which operations are applied. The first operator selected is applied to the initial state of the world, the next operator selected is applied to the resulting state and so forth.

The ideas of the preceding paragraph are best clarified by an example (Simmons(8))presented in the explicit representation of the STRIPS system. Imagine a very simple world consisting of a robot (named ROBOT), two boxes (called B1 and B2) and a room. Suppose further that the robot and boxes may either be in the room or outside the room. An SWM for one possible state of such a world is

```
((BOX B1)
 (BOX B2)
 (INROOM ROBOT)
 (INROOM B1)
 (OUTSIDEROOM B2))
```

Operations in this simple world may include GOINTOROOM, GOOUTSIDEROOM, PUSHBOXIN, PUSHBOXOUT and perhaps some others. The operator GOINTOROOM

may be defined as follows:

Operation name: GOINTOROOM

Parameters: ∅ — Robot moves from outside to inside.

Preconditions: ((OUTSIDEROOM ROBOT))

Effects:

delete list: ((OUTSIDEROOM ROBOT))

add list: ((INSIDEROOM ROBOT))

The operator may be applied in any state satisfying the precondition. (The precondition defines a family of states, the set of all states in which (OUTSIDEROOM ROBOT) is true. The operator may be applied to any member of the family.) When the operator is applied, the relationship (OUTSIDEROOM ROBOT) is removed from the SWM and (INSIDEROOM ROBOT) is added.

Variables may be used in the definition of operators. For example, consider the following:

Operation name: PUSHBOXIN

Parameters: (b) — Robot pushes box b into the room

Preconditions: ((OUTSIDEROOM ROBOT) (BOX b) (OUTSIDEROOM b))

Effects:

delete list: ((OUTSIDEROOM ROBOT) (OUTSIDEROOM b))

add list: ((INSIDEROOM ROBOT) (INSIDEROOM b))

The operator PUSHBOX may be applied if some b can be found which satisfies the preconditions. The preconditions state that the value of b must be a box and that that box must be outside the room. Note that the same b is used to describe the effects of the operation.

The reader who is not familiar with SWMs and operators such as those just presented is encouraged to read the STRIPS paper and to construct various SWMs and operators for himself. In the remainder of this paper, a complete familiarity with such SWMs and operators is assumed.

It is hopefully clear that operations such as those just presented are completely under the control of the robot. Since the robot must actively participate in each operation, simultaneous operations are not possible. The necessity of robot participation and the inability to model simultaneous processes lead to certain difficulties in modeling even ordinary situations. Considering the simple process of filling a bucket with water (suggested by L. Siklossy(6)). Through various sequential operations the robot should place the bucket beneath a water tap and turn the tap on. Once the water has been turned on, the process which actually fills the bucket begins. Since, once the tap is turned on, the water flows into the bucket without the robot's help, the robot should be free to perform other tasks while the bucket fills. Unfortunately, it is impossible to model this situation by using the kinds of operators found in omnipotent robot systems. The main difficulty, of course, is that there is no provision for processes (such as the filling of the bucket) which take place without the active participation of the robot.

Other difficulties, caused by an inadequate representation of time, are also apparent. Notice that operators have no notion of elapsed time associated with them. Further, there is no provision for specifying the infinite number of intermediate states of the world which exist, changing

from instant to instant, over the duration of a process. These deficiencies make it appear as if the process modeled by an operator were initiated and brought to completion in a single instant. Thus, it becomes impossible to adequately model the gradual changes, such as the slow filling of the bucket, which so often characterize a process.

4.0 A closer look at the nature of processes.

Since the representation of process knowledge has led to difficulties in previous modeling systems, it is no doubt worthwhile to reconsider the very nature of processes. For purposes of this paper, a process is a bringing about of a set of changes through a continuum of alterations. Thus, a process brings about a continuous, uninterrupted, time ordered sequence of changes over some time interval.

Although it is tempting to believe that some changes are not brought about through a continuum of alterations, a more microscopic view of a situation seems always to reveal an ongoing of graduate modifications. Even such flip-flop changes as the throwing of a switch, the changing of a computer bit or the changing of one's own mind are brought about by continuums of alterations. Seemingly sudden gross changes in the state of the world as seen from the macro point of view (apparently) are always explainable from the micro vantage as the reaching of certain thresholds through gradual alterations.

Clearly, from the standpoint of a robot or a human being, many processes occur so quickly that for all practical purposes the corresponding changes are effectively instantaneous. For such processes some construct similar to the operation prototype is useful and merits

inclusion in a new system. However, since there are processes which do not occur quickly, a new construct must be found for modeling them. One possibility for such a new construct is to depict certain aspects of the world by real variables whose values are defined by numerical equations involving time. Since such equations would be valid only during the duration of a process, before pursuing this new construct further, it will be necessary to take a closer look at the mechanisms involved in initiating and terminating processes.

4.1 Necessary and sufficient conditions for process initiation

In the STRIPS robot and similar systems, an operation may be applied when its preconditions are met. A robot usually must choose among a variety of possible alternatives. Only the operation selected by the robot is actually carried out. Hence the preconditions are necessary conditions for operator application, but they are not sufficient. Operators are applied only when the preconditions are met and the robot dictates that the operation indeed be performed.

Now consider a world in which an empty bucket is positioned below a water tap and the tap is on. Given these preconditions (without tricks) it is obvious that the bucket will begin filling with water no matter what the robot dictates.

Thus there appear to be two types of processes. One type is initiated involuntarily as soon as its preconditions are met. The other type is subject to the choice of some intellignet entity which performs a part of the work required to carry out the process. These two types become confounded in worlds containing more than one intelligent entity, since

each entity may choose work only for itself but must somehow take into account not only involuntarily initiated processes, but also those processes which may be voluntarily engaged in by the other entities.

To simplify this situation it may be argued that the selection of a single choice from among many alternatives is itself a process. Further, an entity's selection at any moment may be modeled in the SWM. For example, if the robot has currently selected to (PUSHBOXIN B2), then a relation such as (SELECTED ROBOT PUSHBOXIN B2) could appear in the SWM. If (SELECTED ROBOT PUSHBOXIN b) is made a precondition of PUSHBOXIN, then PUSHBOXIN should be initiated as soon as its preconditions are met. Under this scheme, the preconditions of all processes define the necessary and sufficient conditions for process initiation.

(Since the selection of a choice involves a process, there is no reason why the selection process itself may not be modeled. Indeed, if a robot is to be able to predict the actions of other robots, then the predicting robot must contain some model of the selection process used by other robots.)

If simultaneous processes are allowed, then in mid course some process A may cause the preconditions of a process B to be met. Immediately B is initiated while A continues.

4.2 The termination of a process

Since a process is not instantaneous, once a process has begun it is necessary to consider how long the process will remain operative. A process, once initiated, remains in progress as long as a certain set of conditions are met. For example, a man may continue to walk toward

a point X so long as the man exists, is able to walk, wishes to walk to X and is not yet at X. Further, the point X must continue to exist and there must always be some step which the man can take which will bring him closer to X. This process has a natural completion. When the man reaches X, the condition that he be not yet at X is broken and the process stops. The breaking of any of the other conditions will also interrupt (and thus terminate at least temporarily) the process. For example, if the man becomes tired and decides that he no longer wishes to walk (just now) but wishes to rest, then the process is interrupted.

In building a construct for the modeling of processes, it will be necessary to take these conditions for the continuation of a process into account. Thus, in addition to process initiation conditions there must also be process continuation conditions.

5.0 A process model for non-omnipotent robots: an overview

Although not yet implemented on a large scale, a schemata which corrects many of the deficiencies suffered by omnipotent robot systems has been devised. The schemata calls for a new system composed of three basic parts: a process monitor (outlined earlier), a set of process scenarios and an SWM. The process scenario is analogous to the operator in previous systems. The control portion of previous systems which keeps track of the sequential application of operators is somewhat analogous to the monitor.

Each type of process allowable in the world being modeled is characterized by one of the process scenarios. The scenario indicates

the process's initiation conditions (the necessary and sufficient conditions for process initiation) and the process's effects. For those processes which occur so rapidly as to be effectively instantaneous, the effects are specified simply by add and delete lists. (Such processes were well modeled by the operators of previous systems.) However, if a process is sustained for more than a brief instant, equations are included to describe how the process alters the world with the passage of time. These equations constitute one of two methods employed in representing facts in the SWM.

The process monitor is a control program which keeps track of the SWM and the various processes which are in operation at any given time. A major feature of the process monitor is an elastic set of process control blocks which grows or diminishes with the number of active processes. Each control block is characterized by a reference to some process scenario and a set of process parameter bindings. If several similar processes occur simultaneously, then several control blocks are set up, each referencing a common scenario. Of course such similar processes will be distinguished one from the other by differences in their sets of bindings. (These control blocks are very similar to the control blocks used in multi-processing environments to support multiple users, any number of whom may actually be using the same reenterable program.)

Although control of the system is in fact centered in a monitor executive, the system performs in such a way that each process modeled by the various control blocks seems to alter the SWM with the passage of time in accordance with the rules of the associated scenario. From

a vantage point external to the system, it appears as if all the processes modeled by the control blocks are modifying the world simultaneously. In a sense, this is actually what does happen. Unlike simulation models which update themselves at small regular intervals, the process monitor solves sets of simultaneous equations to determine critical times in the set of ongoing processes. These simultaneous equations, of course, come from the definitions of the various processes. But more of this later.

Intuitively, the process monitor behaves as if it were a demon in charge of carrying out all the processes in the modeled world. Given an initial state of the world and a set of process scenarios, the demon determines all processes which would be initiated. For each process to be initiated, the demon selects an imp (a control block) and charges the imp with the realization of the process. As the various imps make changes in the state of the world, the demon watches for new opportunities to start other processes, assigning more imps as needed. When an imp's process is completed or interrupted, the imp notifies the demon who in turn releases the imp from its charge.

In performing its task, the monitor is constantly referring to and altering the system's representation of the state of the world. The system has two types of data structures for storing state knowledge. One data structure is a set of explicit relations such as (TYPE BOX1 BOX). This data type is very much like that used by STRIPS. To record relations which are undergoing gradual change, a different construct is needed. All gradual changes are modeled in the system by employing real variables. Thus, to model the altitude of a slowly rising balloon, BALLOON, the altitude, Yaltitude, is a real number. An entity such as

(ALTITUDE BALLOON Yaltitude) is called a relation skeleton since the relation term Yaltitude is really a variable. If at some moment Yaltitude = 1000, then the relation skeleton and numeric equation serve to define the explicit relationship (ALTITUDE BALLOON 1000). A numeric variable such as Yaltitude is most often defined by systems of equations associated with some process which is changing the relationship indicated by the skeleton in which the variable appears.

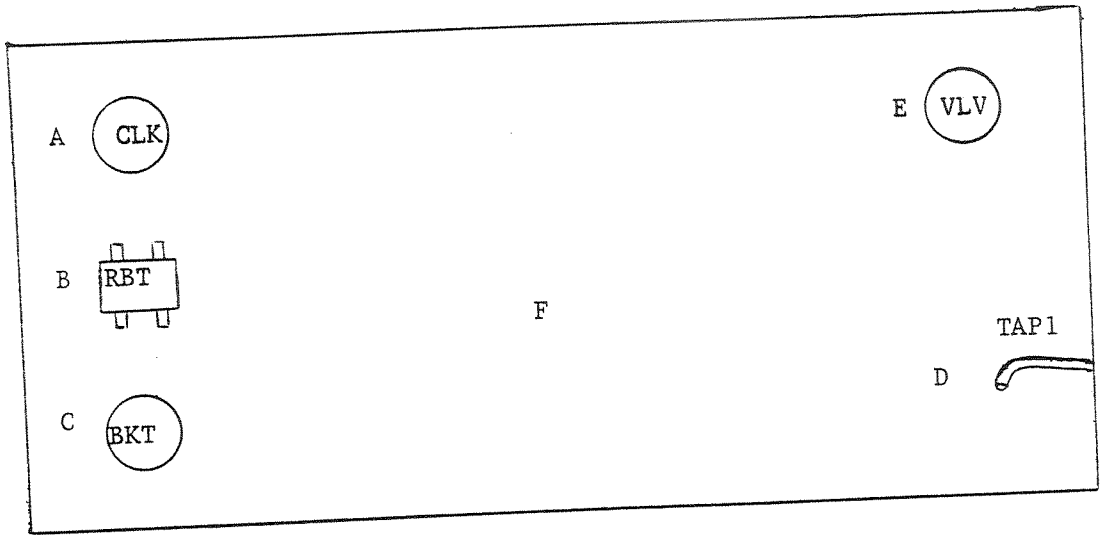
Symbols for real variables are formed by conjoining the capital letters C and Y with a string of lower case letters and numbers. (e.g., Clength2, Yaltitude) The definitions of process scenarios which follow make extensive use of such variable names. Variables whose names begin with C, C-variables, are bound when a process's initiation conditions are met and hence remain constant for the duration of the process. Y-variables are variables which a process itself defines. The definition is in terms of a system of equations involving C-variables and time variables.

The notion of time is modeled in the system by real variables. The symbol @ is used to denote the real variable expressing current model time. The value of @ is not a time of day (such as 3:00 PM), but is a quantity expressing the year, month, day, hour and fraction of the hour. Process scenarios make extensive use of the locally defined time variables ç and $. The local variable ç is always bound to the time at which the local process was initiated. The local variable $ is defined to be @ - ç, the age of the local process.

6.0 A sample world

The sections which follow will further describe the process monitor, scenario formalisms and the representation of the SWM. Since it appears impossible to describe any one of these three components of the system individually, a series of examples of scenarios will be presented. Each new scenario will introduce new complications in at least one of the three system components.

In order to illustrate various facets of the system, it will be helpful to have some simple world in mind. Consider the world shown in Figure 1. The initial state of the world is expressed entirely by use of the set of explicit relationships. The world consists of a single doorless room containing a robot, a clock, a bucket, a water tap and a valve. The valve and water tap are mounted on the ceiling. The robot may turn the valve if it stands beneath it. Several areas, A through F, are indicated. These areas are large enough to contain the clock, the robot and the bucket simultaneously. There is a drain, not shown, in the floor beneath the tap. Thus, any water not caught by the bucket simply disappears into the drain.



Initial SWM given by explicit relationships

```
(TYPE   A    LOCATION)
(TYPE   B    LOCATION)
(TYPE   C    LOCATION)
(TYPE   D    LOCATION)
(TYPE   E    LOCATION)
(TYPE   F    LOCATION)
(TYPE   CLK  CLOCK)
(TYPE   RBT  ROBOT)
(TYPE   BKT  BUCKET)
(TYPE   VLV  VALVE)
(TYPE   TAP1 TAP)
(ALARM  OFF  CLK)
(PUSHABLE CLK)
(PUSHABLE BKT)
```

```
(AT  CLK  A)
(AT  RBT  B)
(AT  BKT  C)
(AT  VLV  E)
(AT  TAP1 D)
(ONFLOOR CLK)
(ONFLOOR RBT)
(ONFLOOR BKT)
(ONCEILING VLV)
(ONCEILING TAP1)
(ORIENTATION BKT UP)
(CAPACITY BKT 100)
(CONTENT BKT 0)
(CONTROL VLV TAP1)
(MAXRATE VLV 10)
(RATE VLV 0)
```

FIGURE 1

An Initial State of a Sample Model World

7.0 Elementary examples of scenarios. (Process scenarios for early risers)

Focusing attention now on the process scenario, each scenario consists of two basic parts: process initiation conditions (ICs) and process effects. Scenarios for instantaneous processes (processes that transpire so quickly that their duration effectively consumes zero units of time) are almost identical to STRIPS operators. Consider the scenario for setting an alarm clock.

Scenario name:	SETALARM	Robot r sets clock k (at place n) to sound at time Cstime.
Parameters:	(r k Cstime / n)	
Initiation conditions:		
symbolic:	((SELECTED r SETALARM r k Cstime) (ALARM OFF k) (AT r n) (AT k n))	
numeric:	ç < Cstime	
	12 > Cstime - ç	
Effects - I:		
delete list:	((ALARM OFF k))	
add list:	((ALARM SET k Cstime))	

A process control block using this scenario will reference the scenario by its name, SETALARM, and by a set of bindings for the parameters (r k Cstime n). The symbol Cstime is an example of the C-variables mentioned earlier. The value of Cstime (written =Cstime) may be any real number. However, such variables as r, k and n must take on only discreet values. (e.g., =n = A or =n = B ... or =n = F.) The slash in the parameter list is used to divide the list into two parts. Parameters in the first portion of the list are called "primary" parameters. Those in the latter portion of the list are called "secondary" parameters. Given the bindings of the primary parameters and the fact that the

initiation conditions are met, the values of the secondary parameters may be uniquely determined for that state of the world meeting the initiation conditions. The usefulness of primary parameters will be seen later when initiation inhibitors are discussed.

The ICs specify the necessary and sufficient conditions for the process to be initiated. The conditions are divided into two categories: symbolic and numeric. Since the robot must choose to set the alarm before the alarm may be set, the first symbolic condition guarantees that the robot =r has indeed elected to perform the task.

The SELECT relation is intended to indicate what the robot plans to do in the next instant. It does not reflect long term goals. For example, if the robot and the clock are not at the same location and the robot's immediate goal is to set the alarm, then the robot must SELECT to GOTO the area containing the clock. The robot uses the processes of goal oriented, means-end analysis to plan its sequence of selections.

The second symbolic IC states that the alarm of clock k must be off (not SOUNDING and not SET) before the alarm may be set. The last two ICs guarantee that the robot is close enough to the clock to set the alarm. Other conditions involving the TYPE relation could be included in the ICs, but are not necessary. For example, (TYPE k CLOCK) is not needed since =k must be a clock if (ALARM OFF =k) is true.

The symbolic initiation conditions are augmented by numeric conditions. In the numeric equations ç represents the time at which the process begins (in model time). Since the SETALARM process takes place so quickly, the entire process may be thought of as transpiring in the single instant

indicated by ç. The first numeric condition is written in the scenario as

ç < Cstime.

This constraint indicates that the alarm may only be set to sound in the future. That is, the set time, Cstime, must be greater than the current time, ç. In an actual computer implementation, the condition would be presented in some more convenient form such as (GREATERP Cstime ç). Since all clocks in the robot world have a twelve hour cycle, the robot cannot set the alarm to go off after more than twelve hours. This restriction is specified by the second equation.

The effects portion of the scenario is headed by "Effects - I" which indicates that the effects of the process are instantaneous. The add and delete lists indicate that (ALARM OFF =k) is to be deleted from the SWM and (ALARM SET =k =Cstime) is to be added. Although (SELECTED =r SETALARM =r =k =Cstime) will not be deleted by SETALARM, it will be deleted by the process which determines the robot's next selection.

Setting the alarm is worthless unless there is a process for sounding the alarm at the proper time. Hence

Scenario name:	SOUNDALARM	An alarm clock k sounds at time Cstime and continues to sound.
Parameters:	(k Cstime /)	
Initiation conditions:		
symbolic:	((ALARM SET k Cstime))	
numeric:	ç = Cstime	
Effects - I:		
delete list:	((ALARM SET k Cstime))	
add list:	((ALARM SOUNDING k))	

The important thing to notice in this scenario is the absence of a SELECTED condition in the ICs. Once set, the clock does not need the

instantaneous processes, but are destroyed almost immediately.) Associated with each of the duration process control blocks is an interrupt time, a number (or infinity) which indicates the earliest model time at which the process is certain to be interrupted. The monitor executive determines the earliest model time at which any existing process is certain to be interrupted. Call this time T_I. To determine what to do next, the monitor executive searches through the process scenarios looking for a scenario which, for some binding of its variables, may be initiated before model time T_I. If no such scenario is found, the monitor executive sets the current model time to T_I and performs the indicated interruption. If more than one scenario is found which can be initiated before model time T_I, then the monitor determines the process which can be initiated at the earliest model time, sets the current model time to that time and creates a new control block to model the new process.

Thus, the monitor executive skips the current model time from one critical point to another, initiating or interrupting the model processes associated with the critical points. (At any critical point in model time, an initiation or interruption of one process may cause a cascading effect. That is, an initiation or interruption may precipitate a series of other initiations and interruptions.)

Consider now how the procedures just discussed may be used to model the initiation of the SOUNDALARM process. Suppose the current model time is @ = 1572.3 and (ALARM SET CLK 1580.0) is true in the SWM. Suppose further that the robot is asleep, that some duration processes are being modeled and that the earliest interruption time for any duration process is T_I = 1583.7. Using the SOUNDALARM scenario and SWM, the process monitor

determines that the next critical model time is model time 1580.0. The monitor sets @ to 1580.0 and creates a control block for SOUNDALARM. Using the control block, the monitor builds up a new SWM by making the appropriate deletions and additions. Once the update is made, the control block is abandoned.

With the SOUNDALARM process out of the way, the monitor recycles and again determines the next critical model time. The next interrupt is still pending at T_I = 1583.7, but, because of the SOUNDALARM process, the initiation conditions for AWAKENROBOT are now met. Hence, the next critical time is the current model time. Keeping @ frozen, the monitor models the awakening of the robot.

Cycling again, the monitor may find that no other processes may be initiated before T_I = 1583.7. Hence, the monitor resets @ to 1583.7 and performs the interruption of the duration process.

8.0 More complex scenarios: The filling of a bucket

The power of the process scenarios does not begin to become apparent until processes involving a continuum of gradual change are considered. Consider the filling of a bucket with water. A push process may easily be defined which places the bucket BKT under the tap TAP1. The following scenario describes the process of turning the controlling valve.

Scenario name:	TURNVALVE	Robot r turns valve v to rate Crate.
Parameters:	(r v Crate / Cmaxrate n)	
Initiation conditions:		
symbolic:	((SELECTED r TURNVALVE r v Crate) (MAXRATE v Cmaxrate) (AT r n) (AT v n))	
numeric:	$0 \leq$ Cmaxrate - Crate $0 \leq$ Crate	

Effects - I:

delete list: ((RATE v *))

add list: ((RATE v Crate))

For purposes of the model, TURNVALVE is an instantaneous process. Note the use of * in the delete list. All relationships of the form (RATE =v --) are deleted from the SWM. Thus, the flow rate before the valve is turned is unimportant and is left unspecified. The flow rate after the valve is turned (Crate) is constrained to be between zero and the maximum rate. (Turning the valve to a zero flow rate turns the valve off.)

Once the control valve has been set to some non-zero flow rate, the process of filling the bucket begins immediately. The scenario for FILLBUCKET is as follows:

Scenario name: FILLBUCKET — Bucket b is filled by water from tap t.

Parameters: (b t / v n Crate Ccapacity Cinitialcontent)

Initiation conditions:

symbolic: ((CONTROL v t) (RATE v Crate) (AT t n) (AT b n) (ORIENTATION b UP) (CAPACITY b Ccapacity) (CONTENT b Cinitialcontent))

numeric: $0 < Crate$

$0 < Ccapacity - Cinitialcontent$

Effects - G:

symbolic: ((CONTENT b Ycontent))

numeric: $Ycontent = Cinitialcontent + Crate \cdot \$$

Continuance conditions:

symbolic: ((RATE v Crate) (AT b n) (ORIENTATION b UP))

numeric: $0 < Ccapacity - Ycontent$

This scenario differs markedly from those presented earlier because its effects are continuous and gradual rather than instantaneous. (Instantaneous effects are indicated in the scenario by "Effects - I." Gradually changing effects are indicated by "Effects - G".) The initiation conditions are stated in exactly the same way as the ICs of previous scenarios. Notice that the flow rate must be positive and there must be room for more water in the bucket. Further, there is no SELECTED relationship needed for the initiation of the process since, given the preconditions, the initiation of the filling of the bucket is independent of the robot's will. Note also that (ONFLOOR b) and (ONCEILING t) are omitted from the ICs. This may be done if it is assumed that there is no process for moving a tap or removing a bucket from the floor.

8.1 Gradual effects

Unlike previous scenarios, the effects of this process are not given by a delete list and an add list. Rather, a formalism is presented which states how conditions will gradually change with the passage of time. The symbolic portion of an "Effects - G" formalism uses relation skeletons with Y-variables to indicate which relationships will be gradually altered by the process. In the present example, the only relationship skeleton given is (CONTENT b Ycontent). This skeleton indicates that the relationship (CONTENT =b --) will be altered. The skeleton further indicates that gradual alteration will be accomplished by variations in Ycontent with respect to time. The precise relationship between Ycontent and the various process parameters is indicated by the numeric portion of the formalism which states that Ycontent is equal to the content of the bucket when the process began, Cinitialcontent, plus the product of the flow rate,

and elapsed process time, $.

8.2 The changing of the SWM via the process monitor

Consider now how the process monitor uses the FILLBUCKET scenario to change the SWM. As was discussed earlier, state knowledge is represented by two distinct data types. One of these types is a set of explicit relationships. The effects of all processes heretofore considered have simply been to add or delete relationships from this set. The other data type, which uses real variables and relation skeletons, is needed for the modeling of gradual change. At the time when the ICs of FILLBUCKET are met, (CONTENT =b =Cinitialcontent) is an explicit relationship in the set of explicit relationships. When the process monitor executive determines that the ICs of FILLBUCKET have been satisfied, a process control block is created to model the process. In the process control block a pointer is set up which points to the FILLBUCKET scenario. Further, the bindings of all the scenario's parameters are recorded in the block. The time at which the process is initiated is also recorded (as the binding for ¢). By using the parameter bindings and the scenario, the monitor determines that the relationship (CONTENT =b --) will be altered by the process. Since the relationship (CONTENT =b --) will be altered by the process, the explicit relation (CONTENT =b =Cinitial-content) is removed from the set of explicit relationships. At the same time, the relation skeleton (CONTENT =b Ycontent) is added to a new set, the set of relation skeletons. Associated with each skeleton in the set is a pointer to the control block which models the process defining the variables in the skeleton. (All variables of a skeleton must be defined by a unique process. Further, no two skeletons on the

skeleton list may differ only in the names of variables. For example (CONTENT =b Y1) and (CONTENT =b Y2) are not both allowed since such a situation would indicate a conflict in the definition of the content of bucket =b.)

If at some instant it is important for the system to know the content of some bucket, say BKT, then a search is made in the SWM for an element of data matching the pattern (CONTENT BKT --). A datum matching this pattern might appear either in the set of explicit relations or ("exclusive or") in the set of relation skeletons just introduced. If the matching datum is found in the set of explicit relations, then the third component of the datum is an explicit value for the content of the bucket. If the matching datum is found in the set of relation skeletons, then the third component of the datum is a variable. Further, there is associated with the datum (which is a relation skeleton) a pointer to a process control block whose modeled process defines the value of the variable. To evaluate the variable, and thus to determine the content of the bucket, the system has only to use the equations in that scenario which is pointed to by the control block. All pertinent equation parameters have been dutifully recorded in the process control block for use at this time.

8.3 On continuance conditions

In addition to a description of the continuing effects, the scenario also indicates conditions which must hold if the process is to continue. The symbolic continuance conditions indicate that the rate of water flow must remain constant at Crate, the bucket must remain beneath the tap and its orientation must remain UP (i.e., it must not be tipped over). Further,

the numeric portion of the continuance conditions states that the content of the bucket must not surpass the bucket's capacity. Additional conditions such as (CONTROL v t) and (CAPACITY b Ccapacity) might also be included in the continuance conditions. However, since neither the controlling of tap =t by valve =v nor the capacity of bucket =b is subject to change, devices guaranteeing the continuance of these relationships are unnecessary.

8.4 How the process monitor handles continuance conditions

When a process control block is set up, provisions are made for the eventual interruption of the process being modeled. Associated with each relation in the set of explicit relations is a list of pointers known at the "sustains list." Each pointer on the list points to a process control block which is modeling some process whose continuation depends on the associated relationship. If the relationship is ever deleted from the set of explicit relationships, all processes whose control blocks are pointed to by members of the relationship's associated sustains list will be interrupted. Thus, when a process control block is set up for a FILLBUCKET process, pointers to the block are added to the sustains lists of (RATE =v =Crate), (AT =b =n) and (ORIENTATION =b UP).

(The symbolic continuance condition (RATE v Crate) is technically a relation skeleton since it contains a real variable, Crate. However, all C-variables are bound when the ICs are met. Thus, within the effects portion of the scenario, C-variables are correctly interpreted as constants. This means that a relation skeleton such as (RATE v Crate) becomes equivalent to the explicit relation (RATE =v =Crate).)

The interrupt provisions just discussed are used to monitor the

symbolic continuance conditions. The numeric conditions are monitored by another mechanism. When the control block is set up, a system of simultaneous equations is constructed from the numeric portion of the effects formalism and the numeric portion of the continuance constraints. Assuming that the symbolic continuance constraints are not broken, this system defines a feasibility space for the process, the dimensions of which are model time and the quantities represented by Y-variables. For the FILLBUCKET process, the system of equations is

$$Ycontent = Cinitialcontent + Crate \cdot \$$$

$$Ycontent < Ccapacity$$

Using variations of a process's system of equations, the monitor finds certain points in time which are critical to the process. For example, by solving the system

$$Ycontent = Cinitialcontent + Crate \cdot \$$$

$$Ycontent = Ccapacity$$

for $, a critical point is determined for FILLBUCKET which indicates that the numeric continuance condition will be broken when $ = (Ccapacity - Cinitialcontent)/Crate. Now the variable $ is local to the process. Using @ = $ + ¢, the time (in terms of the global, monitor system model time) at which the constraint will be broken is computed. This interruption time is recorded in the control block where it may be used by the monitor executive. As described earlier (Section 7.1), this time may eventually be used to interrupt the process.

8.5 Examples of interruption

Consider now the interruption of the filling process by the breaking

of some continuance condition. The first continuance condition (RATE v Crate) may be broken by the process TURNVALVE. Suppose the flow rate through the valve is changed from =Crate to 0. Since the rate changes, FILLBUCKET is interrupted. The content of the bucket remains as defined by the value of Ycontent computed at the time of interruption. Further, this content remains (at least temporarily) constant since the initiation conditions needed to restart FILLBUCKET are not present. Specifically, the flow rate is not positive as demanded by the first numeric pre-condition.

On a more technical level, the process TURNVALVE causes (RATE VLV =Crate) to be deleted from the set of explicit relationships. When the process control monitor makes this deletion, it checks the associated sustains list. Because a pointer to the FILLBUCKET process's control block is found on the list, the process is interrupted. Since the process will (sometimes) no longer be operative, state information which was being represented by skeletons and equations must be moved to the set of explicit relationships. Specifically, the numeric equation is solved for Ycontent using the model time of the interrupt to compute $. With Ycontent computed, (CONTENT =b =Ycontent) becomes explicit and is entered into the set of explicit relationships. Once this critical information has been salvaged, the process control block is abandoned for garbage collection.

Suppose the rate were changed from =Crate to some new positive value. The change in Crate would again interrupt the FILLBUCKET process, but in this instance, FILLBUCKET would restart immediately with the new flow rate (and a new initial bucket content reflecting the increase in volume accomplished at the old rate).

Suppose the bucket is moved. This too interrupts the FILLBUCKET process leaving the content of the bucket as defined at the time of the interruption. Of course, moving the bucket does not change the flow rate. The valve stays open, but the water from the tap simply disappears into the drain in the floor.

If the bucket is turned over during filling, FILLBUCKET is interrupted. Some new process (EMPTYBUCKET) must exist in the process set to define how the water spills from the overturned bucket.

If the FILLBUCKET process is left undisturbed, the numeric continuance condition causes the process to "self-interrupt" or "come to normal completion" when the bucket is full. The content of the bucket remains at =Ycontent = =Ccapacity and water continues to flow throught the tap, spill over the bucket and go down the drain. (This interruption is realized when the FILLBUCKET process's critical time is the earliest interrupt time of all the currently active processes.)

Of course, a combination of interrupts could occur simultaneously. In such cases, (in a consistent model) the resulting state of the world will cause various processes or combinations of processes to be initiated which describe the situation.

8.6 Inhibiting extraneous processes

Before moving on to a more complex scenario, an important aspect of the monitor must be discussed. It has been pointed out that the process monitor is continually attempting to start up new processes by finding scenarios whose initiation conditions are met. Clearly, the FILLBUCKET scenario's initiation conditions will be satisfied at all times during

the continuance of a FILLBUCKET process. Unless some special inhibiting mechanism is provided, the monitor will attempt to set up a multiplicity of process blocks, all attempting to model the filling of the same bucket from the same tap. Such process blocks would differ only with respect to process initiation time and the value for the initial content of the bucket. To avoid this situation, no two process blocks are allowed to use the same scenario with identical bindings for primary variables. Thus, no two FILLBUCKET processes may simultaneously be filling the same bucket. Of course, multiple FILLBUCKET processes may still occur simultaneously so long as each process fills a different bucket and uses a different tap.

9.0 Scenarios with effects sandwiches

Many processes are best characterized by sandwiching a set of continuing effects between two sets of immediate effects. For example, suppose TURNVALVE is redefined to mean "robot r turns valve v at turn rate Cturnrate until a flow rate of Cflowrate is achieved." (That is, Cturnrate is the rate at which the flow rate is to be changed. It is helpful to think of Cturnrate as an "acceleration" applied to a "velocity" Cflowrate.) The immediate effect of this process is to redefine the turnrate to be Cturnrate. With the turn rate established, the flow rate itself undergoes gradual change until the process is interrupted. Upon interruption, the valve is no longer being turned and hence the turn rate is immediately set to zero. The scenario for this process is as follows.

Scenario name: TURNVALVE — Robot r turns valve v at turnrate Cturnrate until flow rate Cflowrate is achieved.

Parameters: (r v Cturnrate Cflowrate / Cinitialflowrate Cmaxflowrate Cmaxturnrate n)

Initiation conditions:

symbolic:
((SELECTED r TURNVALVE r v Cturnrate Cflowrate)
(RATE v Cinitialflowrate) (MAXRATE v Cmaxflowrate)
(MAXTURNRATEABS v Cmaxturnrate) (AT r n) (AT v n))

numeric:
Cflowrate $<$ Cmaxflowrate
Cflowrate $\geq$ 0
Cturnrate $\leq$ Cmaxturnrate
Cturnrate $\geq$ -Cmaxturnrate
0 $<$ (Cflowrate - Cinitialflowrate) $\cdot$ Cturnrate

Effects - I:

delete list: ((TURNRATE v *))

add list: ((TURNRATE v Cturnrate))

Effects - G:

symbolic: ((RATE v Yflowrate))

numeric: Yflowrate = Cinitialflowrate + Cturnrate $\cdot$ $

Continuation conditions:

symbolic:
((SELECTED r TURNVALVE r v Cflowrate Cturnrate)
(AT r n))

numeric: Yflowrate $\neq$ Cflowrate

Effects - P:

delete list: ((TURNRATE v Cturnrate))

add list: ((TURNRATE v 0))

The initiation conditions of this scenario should be self-explanatory with the possible exception of the fourth and fifth numeric constraints. Since the flow rate may be either increased or decreased, the turn rate may be either positive or negative. Taken together, the third and fourth equations guarantee that $|Cturnrate| \leq$ Cmaxturnrate. If the desired rate

is greater than the current rate, Cinitialflowrate, then Cturnrate must be positive. Hence, (Cflowrate - Cinitialflowrate) · Cturnrate must be positive. If Cflowrate is less than Cinitialflowrate, then Cturnrate must be negative. Hence, (Cflowrate - Cinitialflowrate) · Cturnrate is the product of two negative numbers and is again positive. Thus, $0 <$ (Cflowrate - Cinitialflowrate) · Cturnrate guarantees that the valve will be turned in the proper direction for achieving Cflowrate. Further, if Cflowrate = Cinitialflowrate, then the process is meaningless. But if Cflowrate = Cinitialflowrate, then (Cflowrate - Cinitialflowrate) · Cturnrate = 0 and the fifth numeric constraint is not met.

The effects portion of the scenario indicates that the process proceeds in three stages. As soon as the process is initiated, all relations of the form (TURNRATE =v --) are deleted from the SWM and the explicit relation (TURNRATE =v =Cturnrate) is added. The mechanisms used to achieve such immediate effects have already been discussed.

The continuing effects are also set in motion when the process is initiated. (The process monitor executive handles the instantaneous effects first, but model time is frozen until constructs for the continuing effects are set up in the control block.) Provisions for the continuance conditions are set up exactly as before. Note that the condition (AT r n) probably could be omitted since the robot r will not wish to move from n so long as it continues to SELECT to turn the valve. However, if a second robot is introduced, the second might forcedly push the first robot away from the valve and stop the turning process.

When the TURNVALVE process is eventually interrupted (by exactly the same procedures described earlier), the valve flow rate is saved by

adding (RATE =v $=Y_I$) to the set of explicit relations, where $=Y_I$ is the value of Yflowrate at interruption time. As soon as the bookkeeping accompanying the interrupt has been completed by the process monitor, the monitor deletes (TURNRATE =v =Cturnrate) from the set of explicit relations and adds (TURNRATE =v 0). This immediate post-process effect is handled by the usual immediate effect mechanisms.

It is of some interest to note what happens if, while turning the valve, the robot decides to change the turn rate and/or the desired final flow rate. When the robot makes its decision, (SELECTED =r TURNVALVE =r =v =Cflowrate =Cturnrate) is deleted from the SWM, causing the interruption of the TURNVALVE process. When this happens, the turn rate is set to zero even if the interrupt was precipitated by the robot's desire to increase the turn rate. As soon as the robot's new selection of values for Cflowrate and/or Cturnrate are entered in the SWM (by adding (SELECTED =r TURNVALVE =r =v =Cflowrate′ =Cturnrate′)) the TURNVALVE process is reinitiated. Since all of these procedures are accomplished while the process monitor executive keeps model time frozen, it appears as if the turn rate never had been set to zero.

9.1 A new definition for FILLBUCKET

The new scenario for TURNVALVE makes necessary a new formulation of the FILLBUCKET scenario which will allow the rate to change. One possible new scenario is the following.

Scenario name:	FILLBUCKET	Bucket b is filled by water from tap t.
Parameters:	(b t / v n Cinitialflowrate Ccapacity Cinitialcontent Cturnrate)	

Initiation conditions:

symbolic: ((CONTROL v t) (RATE v Cinitialflowrate)
(TURNRATE v Cturnrate) (CAPACITY b Ccapacity)
(CONTENT b Cinitialcontent) (AT t n) (AT b n)
(ORIENTATION b UP))

numeric: $0 < \text{Cinitialflowrate}$
$\text{Cinitialcontent} < \text{Ccapacity}$

Effects - G:

symbolic: ((CONTENT b Ycontent))

numeric: $\text{Ycontent} = \text{Cinitialcontent} + \text{Cinitialflowrate} \cdot \$ + \frac{1}{2} \cdot \text{Cturnrate} \cdot \$^2$

Continuance conditions:

symbolic: ((TURNRATE v Cturnrate)
(AT b n)
(ORIENTATION b UP))

numeric: $\text{Ycontent} < \text{Ccapacity}$

The equation given for the computation of the content of the bucket (Ycontent) may be unfamiliar. Compare the equation to the well known formula

$$d_t = d_0 + v_0 t + \tfrac{1}{2} a t^2$$

where d_t is displacement at time t, d_0 is displacement at time 0, v_0 is initial velocity and a is acceleration.

Notice that there is no reference to flow rate in the continuance conditions. This omission is due to the fact that Ycontent is defined in terms of the initial flow rate and the valve turn rate rather than in terms of the varying flow rate. This formulation of Ycontent does, of course, reflect the varying flow rate.

It is clear that the FILLBUCKET process must be interrupted if the flow rate ever changes to zero. Although no explicit mention of this is made in the continuance conditions, provisions for such an occurrence

are implicitly included. If the flow rate changes to zero while FILLBUCKET is operative, then a TURNVALVE process must be at work. But careful examination of the TURNVALVE scenario shows that if the flow rate is ever changed to zero, then TURNVALVE will be interrupted. (To change the rate to zero, Cflowrate of TURNVALVE must be zero.) The interruption of TURNVALVE will cause the turn rate to be changed from some negative quantity to zero. This in turn will cause the desired interruption in FILLBUCKET.

10.0 Summary

The preceding sections have outlined a new methodology for the construction of world models. The underlying philosophy of this methodology, the philosophy which views the world as a collection of processes, has been advanced. To support the new methodology, previous modeling schemes have been reviewed, the nature of processes has been investigated and a rough sketch of a system using the multiple process modeling scheme has been given.

The application of multiple process modeling is clearly not restricted to robots. The ability to understand a dynamic world is an essential skill in the repertoire of any intelligent entity. In particular, multiple process modeling should prove very useful to question answerers and to CAI systems.

The breadth of applicability of the new methodology has been left largely to the imagination of the reader. Such complex tasks as playing baseball (in which the robot must coordinate the movement of this bat with the movement of the pitched ball) and introspection (the robot

thinks about the process he uses to solve problems) seem well within the scope of the multiple process modeling scheme (if not within the scope of the simple system outlined in sections 5 through 9).

REFERENCES

1. Fikes, Richard E. and Nilsson, Nils J., "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, II, 1971, 189-208.

2. Fikes, R. E., Hart, P. E. and Nilsson, N. J., "Learning and Executing Generalized Robot Plans," Artificial Intelligence, III, 1972, 251-288.

3. Hendrix, G. G., "Question Answering via Canonical Verbs and Semantic Models: A Model of Textual Meaning," The University of Texas, Austin, Technical Report #NL-12, January 1973.

4. Raphael, B., "The Frame Problem in Problem-Solving Systems," Proc. Adv. Study Inst. On Artificial Intelligence and Heuristic Programming, Menaggio, Italy, August 1970.

5. Raphael, B., "The Relevance of Robot Research to AI," Formal System and Non-Numeric Problem Solving by Computer, Springer-Verlag, Berlin, 1970.

6. Siklossy, L. and Dreussi, J., "A Hierarchy-Driven Robot Planner Which Generates Its Own Procedures," The University of Texas, Austin, Technical Report TR-10, February 1973.

7. Simmons, R. F., "Semantic Networks: Their Computation and Use for Understanding English Sentences," in Schank, R. and Colby, K. (Eds.), Computer Simulation of Cognitive Processes, Prentice Hall, IN PRESS.

8. Simmons, R. F., "Mapping English Strings into Meanings," The University of Texas, Austin, Technical Report #NL-10, January 1973.

9. Winograd, T., Understanding Natural Language, Academic Press, New York, 1972.