# MODELING SIMULTANEOUS ACTIONS AND CONTINUOUS PROCESSES

Gary G. Hendrix

Technical Report NL 19

September 1973

A Revised and Extended Version of

BEYOND OMNIPOTENT ROBOTS

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. Considerable attention is given to the application of this methodology in robotics.

## ACKNOWLEDGMENT

The author gratefully acknowledges the advice and guidance of Professor Robert F. Simmons and the programming and debugging help provided by Dr. Daniel Friedman.

The work reported herein was supported in part by National Science Foundation grant GJ 509 X.

# MODELING SIMULTANEOUS ACTIONS AND CONTINUOUS PROCESSES*

## 1.0 Introduction: World modeling

Several artificially intelligent entities have been constructed in the last few years which employ what might be called "world modeling" (or "robot modeling") to gain some (perhaps very limited) understanding of the world. To date, the primary application of world modeling has been in the construction of such robot systems as STRIPS (1), the Siklossy-Dreussi-Roach robot (8,9) and the Winograd system (13). (Non-robotic applications of world modeling have also been pursued by Hendrix (5).)

A typical world model contains 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 general, systems using world modeling contain some construct for describing the state of the world at any given point in (model) time and a mechanism for describing (or simulating) changes in the world brought about by the application of operators.

The success of world modeling techniques in enabling robot systems to partially understand and manipulate certain micro-worlds has clearly been demonstrated, the Winograd system being the paramount example. However, the world modeling schemes used by the robot systems cited above suffer from a number of limitations. In each of these systems the robot is a necessary participant in every operation. Since the robot may pick and choose the operations in which it wishes to participate, it appears to have complete control over all changes in objects and relationships between objects. Clearly, in a larger world there may be many intelligent entities, each capable of

*Supported in part by National Science Foundation Grant GJ 509X.

causing changes in the state of the world. Further, there exist 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 other intelligent entities.

Conventional world modeling systems are also greatly limited by their notion of time. The systems allow only one operation to be performed in any one instant. Moreover, there is no notion of time duration associated with their operations.

If an intelligent system is ever to understand a dynamic world containing numerous intelligent entities and a variety of simultaneous processes, the techniques of world modeling must be expanded. To do this, it seems reasonable to build artificially intelligent entities around the philosophy that the world is a collection of ongoing processes. Some of the processes are controlled by the intelligent entity itself, some are controlled by other entities and still other processes are simply natural phenomena. With the passage of time, old processes are brought to completion and new processes are initiated. At all times, objects are defined by (and have their meaning through) the processes which sustain or transform them from instant to instant.

The purpose of this paper is to show how this process philosophy may be captured in a world modeling system. While the discussion will be given in terms of robotics, hopefully, it is clear that a world modeler has other applications. In particular, the system to be presented may be used as a process simulator or, in conjunction with other devices, as a dynamic model of a narrative text.

## 2.0 Realizing the philosophy of processes

In harmony with the philosophy presented above 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 any one entity as the 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 a 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 system which views the world as a collection of processes. The model borrows freely from its predecessors, especially from STRIPS. To put the new mechanism into perspective, it will be helpful to first review the systems supporting conventional robots and to reconsider the very nature of processes.

## 3.0 World modeling in conventional robot systems

(Readers familiar with STRIPS may skip this section.)

Conventional 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 a conventional 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 (10).) Process knowledge is represented by a collection of operators which model processes in the real world. These operators may either be programs (Winograd (13)) or data structures which may be interpreted as programs (STRIPS (1)), or both (Siklossy and Dreussi (8)). (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., a robot's planner) 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 (11)) presented in an explicit representation similar to that used by the STRIPS system. Imagine a very simple world consisting of a robot (named ROBOT), two boxes (called B1 and B2) and a room. Imagine 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)
 (INSIDEROOM ROBOT)
 (INSIDEROOM 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: | ∅ | |
| Preconditions: | ((OUTSIDEROOM ROBOT)) | Robot moves from outside to inside. |
| 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. Consider the simple process of filling a bucket with water (suggested by L. Siklossy (8)). 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 conventional world modeling 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 gradual 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 (by using a robot planner) 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 intelligent 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.

When necessary and sufficient preconditions are employed in modeling worlds in which simultaneous processes are allowed, in mid course some process A may cause the preconditions of a process B to be met. Immediately B is initiated while A continues.

Consider now the selection of a single choice from among alternatives. 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 (i.e., the robot planner) used by other robots.

A robot with multiple environment affecting devices (e.g., two legs, three arms, a mouth and a stinger) might select to participate in multiple (compatible) activities simultaneously. For example, a robot musician might simultaneously select to play the guitar with its arms and hands, sing with its mouth, tap one of its feet to the music and wink at spectators. To more accurately convey the flavor of the simultaneous allocation of various environment affecting resources to sundry tasks, SELECTED relationships may be replaced by relations of the form (ALLOCATED-ACTIVATED r p t), meaning that robot r has selected to allocate its environment affecting subpart p to the accomplishment of task t. Further, because of its selection, robot r has activated p to perform task t. For example, (ALLOCATED-ACTIVATED ROBERT-ROBOT ROBERTS-RIGHT-WRIST ROTATE +60) indicates that Robert the robot has selected to allocate the resource of its right wrist to the task of rotating 60$^{o}$ (as opposed, say, to the task of bending) and has activated the task by, say, setting a certain bit combination in the wrist operation register.

## 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 The process model: An overview

A world modeling system based on the process philosophy outlined in section 1.0 has recently been devised to correct several deficiencies suffered by conventional world modeling systems. The new system is 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. (The new modeling system, like the modeling portions of conventional robots, does not address itself to robot planning.)

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 time sharing 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 ¢ and $. The local variable ¢ is always bound to the time at which the local process was initiated. The local variable $ is defined to be @ - ¢, 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 scenario 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. The robot itself has two environment affecting devices: a mobility unit (a set of legs or wheels) and an arm.

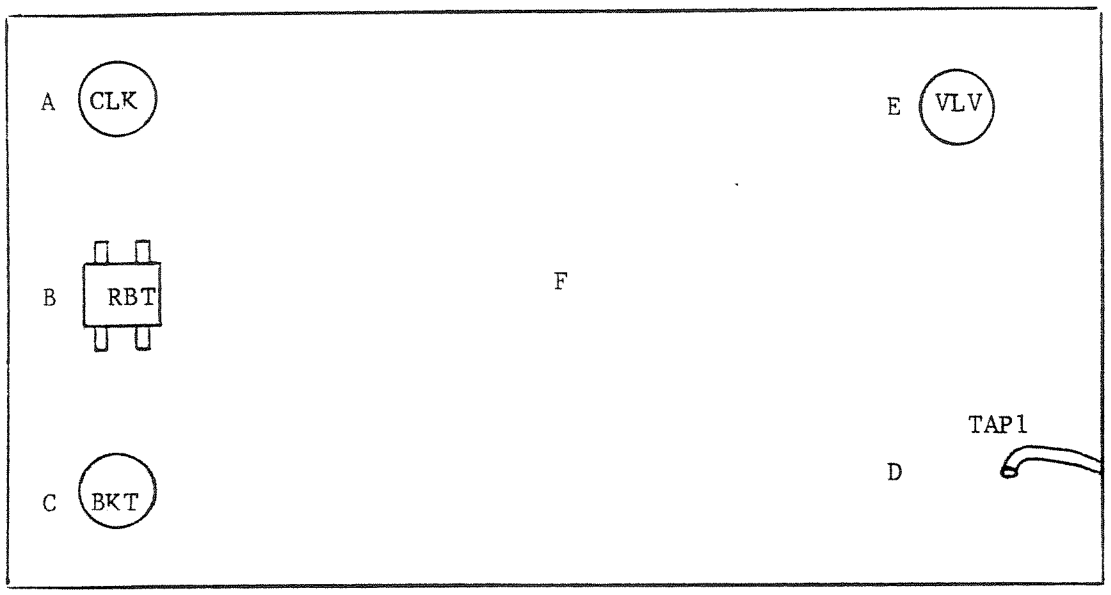
## 7.0 Elementary examples of scenarios

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 uses its arm
                                                       a to set clock k
Parameters:             (a k Cstime / r n)             (at place n) to
                                                       sound at time Cstime.
Initiation conditions:

     symbolic:          ((ALLOCATED-ACTIVATED r a SETALARM a k Cstime)
                         (TYPE a ARM)   (ALARM OFF k)
                         (AT r n)  (AT k n ))

     numeric:           ¢ < Cstime
                        12 > Cstime - ¢
```



Initial SWM given by **explicit relationships**

| | |
|---|---|
| (TYPE A LOCATION) | (AT CLK A) |
| (TYPE B LOCATION) | (AT RBT B) |
| (TYPE C LOCATION) | (AT BKT C) |
| (TYPE D LOCATION) | (AT VLV E) |
| (TYPE E LOCATION) | (AT TAP1 D) |
| (TYPE F LOCATION) | (ONFLOOR CLK) |
| (TYPE CLK CLOCK) | (ONFLOOR RBT) |
| (TYPE RBT ROBOT) | (ONFLOOR BKT) |
| (TYPE BKT BUCKET) | (ONCEILING VLV) |
| (TYPE VLV VALVE) | (ONCEILING TAP1) |
| (TYPE TAP1 TAP) | (ORIENTATION BKT UP) |
| (TYPE RBT-MU MOBILITYUNIT) | (CAPACITY BKT 100) |
| (TYPE RBT-ARM ARM) | (CONTENT BKT 0) |
| (HASASPART RBT RBT-MU) | (CONTROL VLV TAP1) |
| (HASASPART RBT RBT-ARM) | (MAXRATE VLV 10) |
| (ALARM OFF CLK) | (RATE VLV 0) |
| (PUSHABLE CLK) | [(TURNRATE VLV 0) |
| (PUSHABLE BKT) | (MAXTURNRATEABS VLV 5)] |

Used only in section 9.

FIGURE 1

An Initial State of a Sample Model World

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 (a k Cstime r n). The symbol Cstime is an example of the C-variables mentioned earlier. The value of Cstime (written =Cstime when it is important to distinguish the value from the variable) 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 (if the alarm is indeed to be set) the robot must choose to actively participate in the alarm setting process, the first symbolic IC guarantees that the robot =r has elected to allocate the environment affecting resource =a to the task of setting the clock =k to sound at time =Cstime. Since only arms may set alarm clocks, the second symbolic IC guarantees that resource =a indeed is an arm. The third 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.

Several other symbolic conditions might be included in the ICs but are not necessary, being given implicitly by those actually stated. For example, (TYPE k CLOCK) is not needed since =k must be a clock if (ALARM OFF =k) is true. Also, (HASASPART r a) is not needed since =a must be a part of =r if =r ALLOCATED-ACTIVATED =a. In the numeric equations, ¢ 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 ¢. The first numeric condition is written in the scenario as

$$\text{¢} < \text{Cstime}.$$

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, ¢. In an actual computer implementation, the condition would be presented in some more convenient form such as (GREATERP Cstime ¢). 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 (ALLOCATED-ACTIVATED =r =a SETALARM =a =k =Cstime) will not be deleted by SETALARM, it will be deleted by the process which determines the robot's resource allocations.

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

| | | |
|---|---|---|
| <u>Scenario name</u>: | SOUNDALARM | An alarm clock k sounds at time Cstime and continues to sound. |
| <u>Parameters</u>: | (k Cstime /) | |

Initiation conditions:

| | |
|---|---|
| symbolic: | ((ALARM SET k Cstime)) |
| numeric: | ¢ = 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 an ALLOCATED-ACTIVATED condition in the ICs. Once set, the clock does not need the robot's help to sound at the proper time and keep sounding until some other process (such as OFFALARM) stops it. Note that the numeric IC allows the alarm to begin sounding only at the proper moment. (That is, when the process initiation time, ¢, is equal to the time at which the alarm is set to begin sounding, Cstime.) The mechanism which initiates this scenario at the proper moment is the subject of the next subsection.

The reader is encouraged to try his hand at writing a few scenarios for instantaneous processes. Scenarios for such processes as OFFALARM, SLEEPROBOT, and AWAKENROBOT (robot is awakened when an alarm clock sounds) are very similar to those just described.

## 7.1 How the process monitor keeps track of time

To evaluate the numeric initiation condition of the process scenario just presented, it is obvious that the process monitor must have some model of time. The system's notion of time is described by a real variable which is manipulated by the process monitor. Through the use of this model time, the process monitor keeps track of the sequence of alterations in the model world.

The principal components of the process monitor are a set of process control blocks (which are data structures modeling individual processes), and a monitor executive. The job of the monitor executive is to determine when (in model time) new processes should be initiated and when old processes should be interrupted. To perform this task, the process monitor uses information associated with the control blocks. Process control blocks are maintained only for duration processes. (Blocks are temporarily created for 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 with arm a to rate Crate. |
| Parameters: | (a v Crate / r Cmaxrate n) | |

Initiation conditions:

| | |
|---|---|
| symbolic: | ((ALLOCATED-ACTIVATED r a TURNVALVE a v Crate) (TYPE a ARM) (MAXRATE v Cmaxrate) (AT r n) (AT v n)) |
| numeric: | $0 \leq$ Cmaxrate - Crate<br>$0 \leq$ 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<br>$0 <$ Ccapacity - Cinitialcontent |

Effects - G:

| | |
|---|---|
| symbolic: | ((CONTENT b Ycontent)) |
| numeric: | Ycontent = Cinitialcontent + Crate · $ |

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 ALLOCATED-ACTIVATED 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, Crate, 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. Hence, the explicit relation (CONTENT =b =Cinitialcontent) 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 as the "sustains list." Each pointer on this 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

$$\text{Ycontent} = \text{Cinitialcontent} + \text{Crate} \cdot \$$$

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

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

$$\text{Ycontent} = \text{Cinitialcontent} + \text{Crate} \cdot \$$$

$$\text{Ycontent} = \text{Ccapacity}$$

for \$, a critical point is determined for FILLBUCKET which indicates that the numeric continuance condition will be broken when $\$ = (\text{Ccapacity} - \text{Cinitialcontent})/\text{Crate}$. 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 precondition.

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 through 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 all primary variables. Thus, no two FILLBUCKET processes may simultaneously be filling the same bucket from the same tap. Of course, multiple FILLBUCKET processes may still occur simultaneously so long as each process fills a different bucket and uses a different tap. (Difficulties begin to arise if a bucket may simultaneously be beneath two taps or if multiple buckets are beneath a single tap. Such situations may still be modeled, but a more microscopic analysis would be required. The simple FILLBUCKET scenario presented here assumes that if a bucket is below a tap then there is exactly one bucket below the tap and exactly one tap above the bucket.)

## 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 Cdesiredflowrate 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," the flow rate.) 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:

| | | |
|---|---|---|
| <u>Scenario name</u>: | TURNVALVE | Robot r uses arm a to turn valve v at turnrate Cturnrate until flow rate Cdesiredflowrate is achieved. |
| <u>Parameters</u>: | (a v Cturnrate Cdesiredflowrate / r Cinitialflowrate Cmaxflowrate Cmaxturnrate n) | |
| <u>Initiation conditions</u>: | | |
| symbolic: | ((ALLOCATED-ACTIVATED r a TURNVALVE a v Cturnrate Cdesiredflowrate) (TYPE a ARM) (RATE v Cinitialflowrate) (MAXRATE v Cmaxflowrate) (MAXTURNRATEABS v Cmaxturnrate) (AT r n) (AT v n)) | |

| | |
|---|---|
| numeric: | $Cdesiredflowrate \leq Cmaxflowrate$<br>$Cdesiredflowrate \geq 0$<br>$Cturnrate \leq Cmaxturnrate$<br>$Cturnrate \geq -Cmaxturnrate$<br>$0 < (Cdesiredflowrate - Cinitialflowrate) \cdot Cturnrate$ |

<u>Effects - I</u>:

| | |
|---|---|
| delete list: | ((TURNRATE v *)) |
| add list: | ((TURNRATE v Cturnrate)) |

<u>Effects - G</u>:

| | |
|---|---|
| symbolic: | ((RATE v Yflowrate)) |
| numeric: | $Yflowrate = Cinitialflowrate + Cturnrate \cdot \$$ |

<u>Continuation conditions</u>:

| | |
|---|---|
| symbolic: | ((ALLOCATED-ACTIVATED r a TURNVALVE a v Cturnrate Cdesiredflowrate) (AT r n)) |
| numeric: | $Yflowrate \neq Cdesiredflowrate$ |

<u>Effects - P</u>:

| | |
|---|---|
| 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, (Cdesiredflowrate - Cinitialflowrate) $\cdot$ Cturnrate must be positive. If Cdesiredflowrate is less than Cinitialflowrate, then Cturnrate must be negative. Hence, (Cdesiredflowrate - Cinitialflowrate) $\cdot$ Cturnrate is the product of two negative numbers and is again positive. Thus, 0 < (Cdesiredflowrate -

Cinitialflowrate) • Cturnrate guarantees that the valve will be turned in the proper direction for achieving Cdesiredflowrate. Further, if Cdesiredflowrate = Cinitialflowrate, then the process is meaningless. But in such a case (Cdesired-flowrate - 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 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 its arm is ALLOCATED-ACTIVATED to turning the valve. However, if a second robot is introduced, the second might forcibly 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 Vflowrate 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 (TURN-RATE =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, (ALLOCATED-ACTIVATED =r =a TURNVALVE =a =v =Cturnrate =Cdesiredflowrate) 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 Cdesiredflowrate and/or Cturnrate are entered in the SWM (by adding (ALLOCATED-ACTIVATED =r =a TURNVALVE =a =v =Cturnrate' =Cdesiredflowrate')) 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: | $0 < \text{Cinitialflowrate}^2 + \text{Cturnrate}^2$ <br> $\text{Cinitialcontent} < \text{Ccapacity}$ | |
| 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:            Ycontent < 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

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

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, Cdesiredflowrate 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 Scenarios for conventional robot operations

The example scenarios presented in sections 7 through 9 have modeled processes which are rather foreign to conventional robot worlds. This section considers how such conventional operations as GOTO and PUSH may be modeled in the scenario formalism.

The sample world used in the preceding sections faithfully treated time as a continuous phenomenon. However, since the sample world was depicted as having only six discrete locations (A through F), space has not received a realistic, continuous treatment. This situation may, of course, be remedied by describing locations in terms of a spatial coordinate system.

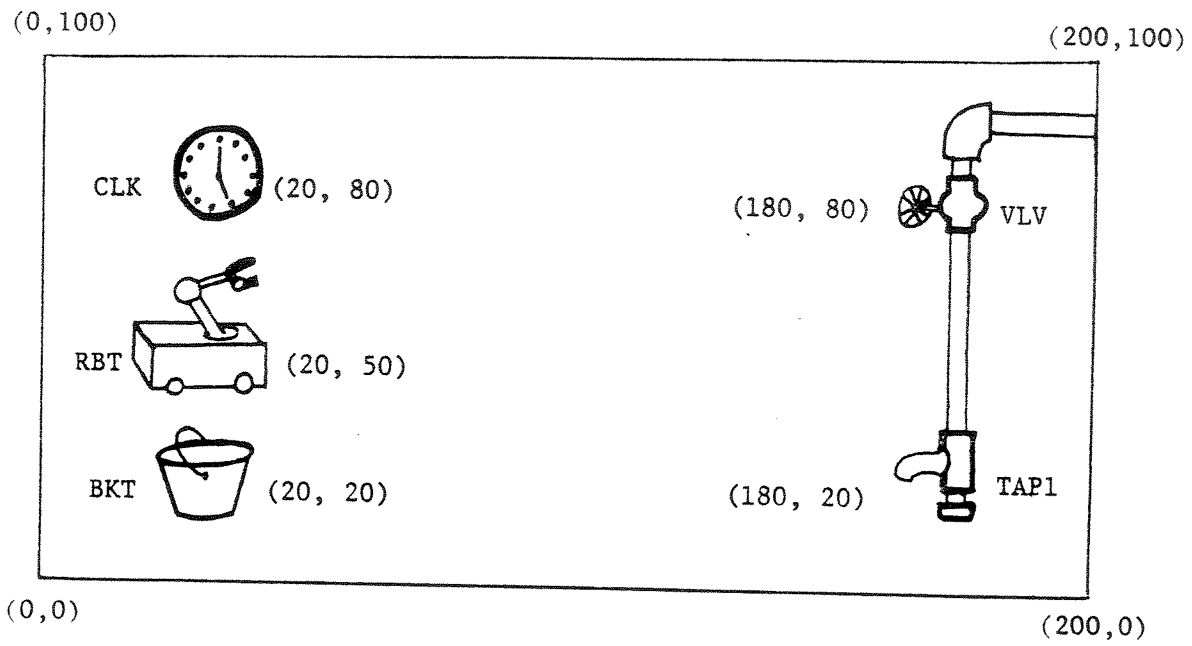
Generally speaking, the inclusion of a coordinate system in a world model can lead to a variety of complications. For example, since objects in the world may have quite curious shapes, equations describing the space occupied by an object may be incredibly complex. Lest the focus of the discussion become diffused in a clutter of engineering difficulties, such problems will be ignored here by assuming

1) all objects in the model world occupy only a single point, and
2) any number of objects may occupy the same point at the same time.

In constructing a model for a real robot, several compromises may be struck between these oversimplifying assumptions and the rigors of an exact system. For example, each object, no matter what its shape, might have associated with it a "sphere of influence." Thus, the location of an object with unknown or irregular shape might be represented as the smallest cube or sphere containing the object. The robot would be able to grasp or push another object when its "sphere" intersects (tangentially) the "sphere" of the object.

## 10.1 A new description of the sample world

A new description of the initial state of the sample world is given in Figure 2. Note particularly that AT relationships are now expressed in the



```
(TYPE CLK CLOCK)                    (CONTROL VLV TAP1)
(TYPE RBT ROBOT)                    (MAXRATE VLV 10)
(TYPE BKT BUCKET)                   (RATE VLV 0)
(TYPE VLV VALVE)                    (AT CLK 20 80)
(TYPE TAP1 TAP)                     (AT RBT 20 50)
(TYPE RBT-MU MOBILITYUNIT)          (AT BKT 20 20)
(TYPE RBT-ARM ARM)                  (AT VLV 180 80)
(HASASPART RBT RBT-MU)              (AT TAP1 180 20)
(HASASPART RBT RBT-ARM)             (SPEEDLIMIT RBT 100)
(MOVABLE CLK)                       (GRASPABLE CLK)
(MOVABLE RBT)                       (GRASPABLE BKT)
(MOVABLE BKT)                       (GRASPABLE VLV)
(IMMOVABLE VLV)                     (GRASPABLE TAP1)
(IMMOVABLE TAP1)                    (NOTGRASPED CLK)
(ALARM OFF CLK)                     (NOTGRASPED BKT)
(ORIENTATION BKT UP)                (NOTGRASPED VLV)
(CAPACITY BKT 100)                  (NOTGRASPED TAP1)
(CONTENT BKT 0)
(FREE RBT-MU)
(FREE RBT-ARM)
```

FIGURE 2

Revised Initial State of Model World

form (AT object x y), where (x,y) is a point in the coordinate system occupied by "object." (The third dimension has been omitted for simplicity. If a bucket and tap occupy the same position, assume the tap may fill the bucket.) In anticipation of movement by the robot, the relationship (SPEEDLIMIT RBT 100) has been included to indicate that the robot RBT may move at a maximum speed of 100 length units per time unit.

Since the coordinate system is being introduced under the assumption (only for simplicity) that two objects may occupy the same point at the same time, it follows that one object (e.g., the robot) may pass through another object without pushing or otherwise affecting it. So that the robot may interact with other objects, assume that if the robot and another object occupy the same point then the robot may "grasp" the object. If the robot is grasping an immovable object, then the robot itself cannot move. If the robot is grasping a movable object, then that object will move with the robot if the robot moves.

## 10.2 GRASPING and RELEASING scenarios

To model the grasping and releasing of objects, two new scenarios may be defined as follows:

```
    Scenario name:          GRASP                       Robot r grasps object b
                                                           with its arm a while
    Parameters:             (a b / r Cx Cy)                at point (Cx, Cy).

    Initiation conditions:

         symbolic:          ((ALLOCATED-ACTIVATED r a GRASP a b)
                             (TYPE a ARM) (GRASPABLE b) (NOTGRASPED b)
                             (AT r Cx Cy) (AT b Cx Cy))

         numeric:           ∅

    Effects - I:

         delete list:       ((NOTGRASPED b))

         add list:          ((GRASPING r a b))

                                   ** ** **
```

| | | |
|---|---|---|
| Scenario name: | RELEASE | Robot r releases object b held by arm a. |
| Parameters: | (a b / r) | |
| Initiation conditions: | | |
| symbolic: | ((ALLOCATE-ACTIVATE r a RELEASE a b) (TYPE a ARM) (GRASPING r a b)) | |
| numeric: | Ø | |
| Effects - I: | | |
| delete list: | ((GRASPING r a b)) | |
| add list: | ((NOTGRASPED b)) | |

According to these scenarios, a robot arm may grasp any number of objects but an object may be grasped by at most one robot arm. Of course, other scenarios may be written to describe other situations. A scenario which limits both the total number of objects and the total weight of the objects which a robot may grasp may easily be written and is recommended as an exercise.

(Very interesting problems begin to arise if robots are allowed to grasp robots or if two robots are allowed to grasp the same object at the same time. For example, if two or more robots become physically united by grasping one another or common objects, what will happen if they attempt to move in different directions? Several outcomes are possible. One possibility is that some robot or object will be broken. Another possibility is that the entangled robots and objects will follow a path representing the vector sum of the paths which the robots would have followed individually.)

## 10.3 Scenarios of dynamic inference

Since, in the sample world, a robot's ability to move is determined by whether or not it is grasping an immovable object, it seems reasonable to set up the following special scenario whose job it will be to determine dynamically whether a robot is itself movable or immovable in a particular world state.

| | | |
|---|---|---|
| Scenario name: | MOVABILITY | The movability of robot b is determined dynamically. |
| Parameters: | ( r / a b) | |
| Initiation conditions: | | |
| symbolic: | ((GRASPING r a b) (IMMOVABLE b)) | |
| numeric: | $\emptyset$ | |
| Effects - I: | | |
| delete list: | ((MOVABLE r)) | |
| add list: | ((IMMOVABLE r)) | |
| Effects - G: | $\emptyset$ | |
| Continuation conditions: | | |
| symbolic: | ((GRASPING r a b) (IMMOVABLE b)) | |
| numeric: | $\emptyset$ | |
| Effects - P: | | |
| delete list: | ((IMMOVABLE r)) | |
| add list: | ((MOVABLE r)) | |

Because of the rule concerning identical bindings of primary variables (see section 8.6), only one MOVABILITY process block will be set up to prevent the movement of any one robot. This will be true no matter how many immovable objects the robot may be grasping. However, if the robot releases the object =b, other immovable objects grasped by the robot are no longer inhibited from activating MOVABILITY and one of them will succeed in restoring the IMMOVABLE condition.

A close look at the MOVABILITY scenario reveals a rather peculiar structure. The initiation conditions and continuation conditions are identical. The post effects are the inverse of the instantaneous effects. Further, there are no gradual effects.

Hence, MOVABILITY acts simply as a binary switch. Whenever there exists a =b for which (GRASPING =r =a =b) and (IMMOVABLE =b) are true, the switch is in the (IMMOVABLE =r) position. Whenever no such =b can be found, the switch is in the (MOVABLE =r) position.

## 10.4 The definition of GOTO

Now that a simplified coordinate system and the notions of MOVABLE and IMMOVABLE as applied to robots have been introduced, a scenario for GOTO may be presented.

```
Scenario name:          GOTO         Robot r goes to point (Cxto, Cyto) from
                                       point (Cxfrom, Cyfrom) at speed Cspeed.
Parameters:             (r Cxto Cyto Cspeed / m Cxfrom
                         Cyfrom Cspeedlimit Ehyp Exrate
                         Eyrate)

Initiation conditions:

     symbolic:          ((MOVABLE r) (ALLOCATED-ACTIVATED
                          r m GOTO r Cxto Cyto Cspeed)
                         (TYPE m MOBILITYUNIT)
                         (SPEEDLIMIT r Cspeedlimit)
                         (AT r Cxfrom Cyfrom))

     numeric:           0 ≤ Cxto
                        0 ≤ Cyto
                        0 ≤ 200 - Cxto
                        0 ≤ 100 - Cyto
                        0 < Cspeed
                        0 ≤ Cspeedlimit - Cspeed
                        ______
```

$$\text{Ehyp} = \text{SQRT}((\text{Cxto} - \text{Cxfrom})^2 + (\text{Cyto} - \text{Cyfrom})^2)$$

$$\text{Exrate} = (\text{Cxto} - \text{Cxfrom}) \cdot \text{Cspeed} / \text{Ehyp}$$

$$\text{Eyrate} = (\text{Cyto} - \text{Cyfrom}) \cdot \text{Cspeed} / \text{Ehyp}$$

```
Effects - I:

     delete list:       ((XRATE r *) (YRATE r *))

     add list:          ((XRATE r Exrate) (YRATE r Eyrate))
```

Effects - G:

symbolic: ((AT r Yx Yy))

numeric: Yx = Cxfrom + Exrate · \$
Yy = Cyfrom + Eyrate · \$

Continuation conditions:

symbolic: ((ALLOCATED-ACTIVATED r m GOTO r Cxto Cyto Cspeed) (MOVABLE r))

numeric: $0 \neq$ Yx - Cxto

Effects - P:

delete list: ((XRATE r *) (YRATE r *))

add list: ((XRATE r 0) (YRATE r 0))

The GOTO scenario introduces the E-variable. E-variables, like C-variables, are bound to constants at initiation time. However, E-variables are derived from C-variables rather than being gleaned from symbolic relations.

Most of the numerical portions of the scenario come from a straightforward application of the Pythagorean theorem. The constants 100 and 200 are the dimension of the model world. Upon reflection, it should be clear that the Effects - I and Effects - P sections of the scenario could have been omitted. However, as will be seen in the next section, this information will be useful in determining what happens to movable objects the robot may be grasping.

## 10.5 The notion of subordinate actions

The STRIPS robot effectively has two GOTO operators. The first, called GOTO, models the solo movement of the robot. The second, called PUSH, models the joint movement of the robot and a box or wedge. Now the STRIPS robot can only push one object at a time so these two GOTO operations are adequate for the STRIPS world. However, if the robot could push from 0 to n objects, then n + 1 distinct operators would be necessary. In addition to pushing, the robot might carry a variety of objects in its pockets or in its hands. Further, the robot might be wearing clothes, false teeth, contact lenses and a toupee. Since all these objects must move along with the robot, the number of necessary operators would quickly proliferate.

This proliferation of operators may be curbed by thinking of the robot's solo movement as a central process with numerous subordinate processes (based on a very small number of scenarios) accounting for the movement of objects which are somehow swept along with the robot. That is, the central process is the solo movement of the robot. If a box is in the path of the robot, then the box is pushed. If an object is held in the robot's hand or is in the robot's pocket, then the object is carried along.

In the description of the revised sample world, the robot carries along with it those movable objects which it is GRASPING. (In an expanded world, the robot might also be WEARING clothes and CARRYING objects in its pockets. The model of such an expanded world could use inference scenarios (section 10.3) to predicate (MOVEWITH object robot) whenever the robot was grasping, wearing or carrying an object. In such a case, MOVEWITH would take the place of GRASPING in the scenario which follows.) Thus, the location of an object grasped by the robot is a function of the movements of the robot. Hence the LOC scenario:

| | | |
|---|---|---|
| <u>Scenario name</u>: | LOC | Object b, being grasped by robot r, is moved from point (Cxfrom, Cyfrom). |
| <u>Parameters</u>: | (b / r a Cxfrom Cyfrom Cxrate Cyrate) | |
| <u>Initiation conditions</u>: | | |
| symbolic: | ((GRASPING r a b) (MOVABLE b) (AT b Cxfrom Cyfrom) (XRATE r Cxrate) (YRATE r Cyrate)) | |
| numeric: | $0 < \text{Cxrate}^2 + \text{Cyrate}^2$ | |

Effects - G:

symbolic: (AT b Yx Yy)

numeric: $Yx = Cxfrom + Cxrate \cdot \$$
$Yy = Cyfrom + Cyrate \cdot \$$

Continuation conditions:

symbolic: ((GRASPING r a b)
(XRATE r Cxrate) (YRATE r Cyrate))

numeric: $\emptyset$

In this scenario the movement of an object grasped by a robot is computed in terms of the robot's rates of displacement with respect to the X and Y coordinates. These rates of robot displacement are, of course, computed by the GOTO scenario, which models the solo movement of the robot. (Indeed, the only reason rates were introduced into the GOTO scenario was for use in LOC.) It is important to note that the starting point ("from" position) of the movement of an object is not necessarily the starting point of the movement of the robot. It is quite possible for the robot to grasp and subsequently release an object with the robot in motion all the while. The reader is invited to consider how the motion of an object grasped by a robot is affected by changes in the robot's speed or course.

## 11.0 Modeling the execution of plans

Sections 5 through 10 have discussed a methodology for modeling a world containing simultaneous actions and continuous processes in terms of an SWM-maintenance procedure (or world simulator). Conspicuously absent from the discussion has been any mention of procedures which allow a robot to plan interactions with its environment. While no robot planner has yet been designed to operate over worlds containing simultaneous actions and continuous processes, the scenario formalism is itself adequate to model the execution of a preformulated plan. However, before considering the actual execution of plans, it will be necessary to gain some notion of what constitutes a plan.

## 11.1 Robot plans as resource utilization schedules

As has been stated previously, a robot engages itself in an activity (and hence precipitates changes in its world) by allocating and activating certain of its resources to the accomplishment of the activity at a time when it is physically possible (given the environmental conditions) for the activity to be set in motion. In attempting to accomplish its goals, it is necessary for a robot to carefully coordinate its environment affecting resources. This coordination is realized through the adoption of a comprehensive resource utilization schedule. Such a schedule is known as a robot plan.

(In conventional robots, all the robot's resources (i.e., the total robot) must (effectively) be allocated to the performance of each operation. Hence, the sequence of desired operations is enough to specify the sequence of resource allocation.)

## 11.2 The representation of a plan

As an example of a robot plan, suppose that the robot is to move from point (20, 50) to point (20, 10), grasping the bucket as it passes through point (20, 20), and releasing the bucket as it passes through point (20, 15). (See Figure 2.) One possible plan for this sequence of events is the following:

Step 1 Allocate and activate the resources of the robot's mobility unit to the task of moving to point (20, 10). Plan to withdraw the resources when the robot reaches the point.

Step 2 When the robot is at point (20, 20) allocate the resources of the arm to the grasping of the bucket. Withdraw the resources once the bucket is grasped.

Step 3 When the robot is at point (20, 15) allocate the resources of the arm to the releasing of the bucket. Withdraw the resources once the bucket is released.

Notice that steps 2 and 3 have three basic parts. First, wait for the proper conditions. Second, make an allocation and activation of resources. Third, make arrangements for withdrawing the resources. Plan step 1 may also be given in three parts by adding a dummy wait condition such as "Wait for the robot to be a robot."

Recognizing each step to be a triple, the plan for a particular robot, RBT, may be formalized as follows.

```
(DOWHEN    RBT  1  TYPE RBT ROBOT)                 ]
(DOWHAT    RBT  1  RBT-MU GOTO RBT 20 10 50)       ]  Step 1
(DOUNTIL   RBT  1  AT RBT 20 10)                   ]
(DOWHEN    RBT  2  AT RBT 20 20)                   ]
(DOWHAT    RBT  2  RBT-ARM GRASP RBT-ARM BKT)      ]  Step 2
(DOUNTIL   RBT  2  GRASPING RBT RBT-ARM BKT)       ]
(DOWHEN    RBT  3  AT RBT 20 15)                   ]
(DOWHAT    RBT  3  RBT-ARM RELEASE RBT-ARM BKT)    ]  Step 3
(DOUNTIL   RBT  3  NOTGRASPED BKT)                 ]
```

The first three relations indicate that the first step in RBT's plan is to make (ALLOCATED-ACTIVATED RBT RBT-MU GOTO RBT 20 10 50) true as soon as (TYPE RBT ROBOT) becomes true. (ALLOCATED-ACTIVATED RBT RBT-MU GOTO RBT 20 10 50) is to be withdrawn when (AT RBT 20 10) becomes true. Other steps in the plan may be interpreted analogously.

Hopefully, it is clear that if a robot's plan has been completely determined before it is set in operation, then the plan is itself a part of the state of the world preceding the execution of the plan. Thus, for simulation purposes, a plan may be represented by a set of initial explicit relations such as those above. To indicate that RBT's plan is to start with step 1, an additional relation such as (PLANSTEP RBT 1) may be included in the set of initial explicit relations.

## 11.3 Scenarios for the execution of a plan

The following two scenarios may be used to model the allocation and activation of a robot's resources during the execution of a plan.

| | | |
|---|---|---|
| <u>Scenario name</u>: | NEXTSTEP | Robot r allocates and activates resources for the next step in its plan. |
| <u>Parameters</u>: | (r / resource Cstepnumber *1 *2 *3) | |
| <u>Initiation conditions</u>: | | |
| symbolic: | ((PLANSTEP r Cstepnumber) (DOWHEN r Cstepnumber *1) (*1) (DOWHAT r Cstepnumber resource *2) (FREE resource)) | |
| numeric: | no <u>C</u>-variable restrictions<br><br>Enextstep = Cstepnumber + 1 | |
| <u>Effects - I</u>: | | |
| delete list: | ((PLANSTEP r Cstepnumber) (FREE resource)) | |
| add list: | ((ALLOCATED-ACTIVATED r resource *2) (PLANSTEP r Enextstep) (INPROGRESS r Cstepnumber)) | |

** ** **

| | | |
|---|---|---|
| <u>Scenario name</u>: | WITHDRAWRESOURCE | The allocation and activation of a resource is withdrawn. |
| <u>Parameters</u>: | (resource / r Cstepnumber *1 *2) | |
| <u>Initiation conditions</u>: | | |
| symbolic: | ((INPROGRESS r Cstepnumber) (DOUNTIL r Cstepnumber *1) (*1) (DOWHAT r Cstepnumber resource *2)) | |
| numeric: | ∅ | |
| <u>Effects - I</u>: | | |
| delete list: | ((INPROGRESS r Cstepnumber) (ALLOCATED-ACTIVATED r resource *2)) | |
| add list: | ((FREE resource)) | |

In these scenarios, symbols of the form "*n" (where n is an integer) are used as variables which can match a string of relation terms of arbitrary length. Thus (DOWHEN r Cstepnumber *1) will match the relation (DOWHEN RBT 2 AT RBT 20 20) with string variable *1 bound to the string "AT RBT 20 20." With this convention in mind the scenarios may be examined more closely.

Consider the NEXTSTEP scenario. The first symbolic initiation condition determines that the next step in the plan of robot r is step number Cstepnumber. The second symbolic IC indicates that the step is to be taken when the relation (*1) is true. The third IC guarantees that the relation (*1) is indeed true. The fourth IC indicates the resource allocation-activation to be made in taking the step while the fifth IC determines whether or not that resource is free for allocation. If the ICs are met, then the allocation and activation of the resource is made and the plan step number is advanced by one.

Scenario WITHDRAWRESOURCE is used to return a resource when the resource has accomplished its purpose with respect to the plan.

## 11.4 More general plans

The scenarios presented above are for use in a planning scheme in which a robot allocates only one resource at each step. There is, of course, no reason why a robot may not make multiple simultaneous allocations in a single step. This is easily modeled by adding an extra variable, such as Cinterstepindex, to the DOWHAT, DOWHEN, DOUNTIL and INPROGRESS relations. The plan execution scenarios must be appropriately modified to insure agreement of Cinterstepindex numbers.

Moving into another area of generality, it is conceivable that a robot might impose a partition over its set of resources and have an independent plan for each subset. (Thus, for example, each hand of a robot octopus might perform its own task using its own plan.) This situation may be conveniently modeled by including a Cpartitionindex in PLANSTEP, DOWHEN, DOWHAT, DOUNTIL and INPROGRESS relations. For each robot, there must be as many PLANSTEP relations as there are subsets in its resource partition.

## 11.5 Loops, jumps and decisions in a plan sequence

As indicated in section 11.3, the NEXTSTEP scenario is responsible for advancing a plan's step number. There is, however, no reason to exclude other processes from affecting the number. Indeed, it is quite easy to write a scenario which changes a plan step number when an arbitrary set of conditions hold. Such a scenario greatly resembles the conditional jumps familiar to computer programmers.

The ability to model jumps in plan sequences opens up a galaxy of tantalizing possibilities. In particular, a robot may engage in plans containing loops, reusable subplans (subroutines), recursive subplans and interrupts. In the light of the model's inherent parallelism, these abilities seem capable of accomodating an infinite variety of robot plans.

## 11.6 Formulating plans

As stated earlier, no planner has yet been devised to operate over a world containing simultaneous actions and continuous processes. However, work in this area has just begun. Since the scenario formalisms and other features of the modeling system were designed with planning in mind, it is hoped that most of the experience gained from constructing conventional robot planners will be applicable in multiple, continuous process planning.

The presence of real variables and numeric constraints, of course, adds a new dimension of complexity to the task of planning over continuous processes.

To meet this challenge, conventional planning must be supplemented by such activities as finding feasible solutions to systems of simultaneous equations.

The sophistication of numeric ability required for planning interactions with the real world may very well turn out to be much less than that required for simulation. To see this, consider the example of a man filling a bucket. The man simply places the bucket beneath a tap and turns on the water. Perhaps while performing other tasks, the man keeps an eye on the process and eventually shuts off the water at the proper moment.

To simulate this scene it would be necessary to know the exact formula for computing the content of the bucket with respect to time. In general, this formula may be very complex. Obviously, the man in this example does not perform a precise simulation to determine the instant at which to cut off the water. Rather, he apparently makes a quick, conservative guess as to when he should examine the progress of the filling process. Several cycles of examination and reestimation later, he turns off the water as he watches it reach the desired level.

The important thing to recognize in this example is that the man has the ability to interact with an external world and hence can update and refine his model. The fillbucket scenario used by a planning robot should be kept simple. The robot might have some rough idea of the water flow rate and the bucket's capacity. (These rough approximations could be given by pairs of numbers expressing upper and lower limits. (See Fikes et al. (3)) In turn, the scenario may provide a range of possible fill times. The robot may plan to examine the state of the filling process at the earliest time. From the inspection, information may be gained which will provide a better second estimate, etc.

The interplay between action, perception and estimation is clearly an important tool in real task performance. The ability to perceive and to refine estimates removes a considerable burden from the numeric considerations of planning.

## 12.0 Implementation

An experimental micro-world simulation system based on the ideas presented has been implemented in GROPE on the CDC 6600 at the University of Texas. GROPE (Friedman (4), Slocum (12)) is a graph processing language which provides very general data structures, including lists and graphs, and compatibility with FORTRAN.

All constructs presented in previous sections have been faithfully implemented with two exceptions. First, the string variables (Section 11.3) have not been implemented. The scenarios of Section 11.3 were realized by introducing dummy terms in some relations so that all relations referenced by string variables were of the same length. The string variables were then replaced by a fixed length sequence of ordinary variables. Second, the monitor has no mechanism for finding the critical points of a system of simultaneous equations. (Such mechanisms do exist for sufficiently well-behaved systems. See Wagner (16) and Zangwill (17).) Rather, for each scenario with Effects - G, a FORTRAN function must be supplied which computes the earliest self-interruption time, given the bindings of parameters stored in the process control block. Tailor-made FORTRAN functions are also used in handling the numeric considerations of initiation conditions.

## 12.1 Interaction with the computer

In one mode of operation, a user may interact with the simulation system by adding or deleting relations from the set of explicit relations whenever model time is frozen at some interesting, critical time. By adding and deleting ALLOCATED-ACTIVATED relations, the user may provide his own simulation of a robot's mind while the system simulates other world components, including the robot's brawn.

## 13.0 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 ongoing 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 SWM-maintenance program using the multiple, continuous process modeling scheme has been given.

The SWM-maintenance program is actually a simulation system. While previous simulators and simulation languages (Gordon (14), Naylor (15)) have been designed primarily to simulate, the system presented in this paper was designed as a theoretical basis for the understanding of a dynamic world by artificially intelligent entities. Care has been taken to represent both state and process knowledge by constructs which an intelligent computer system may conveniently examine and manipulate.

Much attention has been paid to the use of multiple, continuous process modeling in the area of robotics. In particular, the interplay between a robot's brain and brawn has been discussed.

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. Specifically, the new methodology should prove very useful to question answerers and to CAI. Indeed, the system presented was actually developed in an attempt to broaden the scope of the author's robot-like natural language system (Hendrix et al. (5)). That system builds a model of a simple narrative text, using a sequence of STRIPS robot-like operators to depict the sequence of events given in the text.

The breadth of applicability of the new methodology has been left largely to the imagination of the reader. But clearly, the world of robots has been brightened by the ability to model growing flowers, running streams and a sun which wanders gradually across the heavens.

## 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. Fikes, R. E., Hart, P. E. and Nilsson, N. J., "Some New Directions in Robot Problem Solving," in Meltzer, B. and Michie, D. (Eds.), Machine Intelligence 7, John Wiley and Sons, New York, 1972.

4. Friedman, Daniel P., "GROPE: A Graph Processing Language and its Formal Definition, " Technical Report TR-20, August, 1973, Department of Computer Sciences, The University of Texas at Austin.

5. Hendrix, G.G.,"Language Processing Via Canonical Verbs and Semantic Models," Advance Papers of 3IJCAI, Stanford, California, 1973, 262-269.

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

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

8. Siklossy, L. and Dreussi, J., "An Efficient Robot Planner which Generates its Own Procedures," Advance Papers of 3IJCAI, Stanford, California, 1973, 423-430.

9. Siklossy, L. and Roach, J.,"Proving the Impossible is Impossible is Possible: Disproofs Based on Hereditary Partitions," Advance Papers of 3IJCAI, Stanford, California, 1973, 383-387.

10. 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.

11. Simmons, R. F., "Mapping English Strings into Meanings," Technical Report NL-10, January, 1973, Dept. of Comp. Sci., The University of Texas at Austin.

12. Slocum, J., "The Graph Processing Language GROPE 2.0," Masters's thesis in preparation, The University of Texas at Austin.

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

## PERIPHERAL REFERENCES

14. Gordon, Geoffrey, System Simulation, Prentice-Hall, Englewood Cliffs, New Jersey, 1969.

15. Naylor, Thomas H., "Bibliography 19. Simulation and Gaming," Computing Reviews, X, 1 (January, 1969), 61-69.

16. Wagner, Harvey M., Principles of Operations Research, Prentice-Hall, Englewood Cliffs, New Jersey, 1969.

17. Zangwill, Willard I., Nonlinear Programming: A Unified Approach, Prentice-Hall, Englewood Cliffs, New Jersey, 1969.