Design of An AI-Based Self-Sustaining Habitats Control System

Tag Gon Kim and George Mignon
The Environmental Research Lab., The University of Arizona

Bernard P. Zeigler
Electrical & Computer Engineering Dept., The University of Arizona

ABSTRACT

The work described here derives from the implementation of intelligent control for a Bioregenerative Closed Ecological Life Support System. The AI-Based, Distributed Environment Control System (AIDECS) will be capable of supporting, in an integrated fashion, all activities ranging from planning of agricultural activities at the highest level to real-time control of environmental conditions at the lowest level. This paper describes the AIDECS subsystem that deals with control over two widely different time scales: 1) that of scheduling crop planting and harvesting over the extended horizon needed to ensure maintenance of the CO_2/O_2 and other gas balances (typically involving units of weeks, months and years), and 2) that of controlling environmental parameters such as temperature and humidity to be properly correlated with crop requirements and external weather conditions (typically involving units of hours and days).

INTRODUCTION

"Intelligent control," the intersection of artificial intelligence (AI), conventional automatic control, and operations research approaches, is receiving increasing attention in both theory and application[1].

This paper describes the AI-Based, Distributed Environment Control System (AIDECS) under development for a Bioregenerative, closed ecological life support system (BCELSS) at the Environmental Research Lab of the University of Arizona. The AIDECS will be capable of supporting, in an integrated fashion, all activities ranging from planning of agricultural activities at the highest level to real-time control of environmental conditions at the lowest level. This paper describes the AIDECS subsystem that deals with control over two widely different time scales: 1) that of scheduling crop planting and harvesting over the extended horizon needed to ensure maintenance of the CO_2/O_2 and other gas balances (typically involving units of weeks, months and years), and 2) that of controlling environmental parameters such as temperature and humidity to be properly correlated with crop requirements and external weather conditions (typically involving units of hours and days).

The AIDECS subsystem contains a natural language-like interface in which the wide variety of schedules required in the BCELSS may be specified in a uniform manner. Such specifications are mapped into schedule objects that are stored for later implementation and reuse. Such objects may include specification of events which put into effect other schedule objects. This gives rise to hierarchical schedules which facilitate timing of events to occur at coarse- and fine-grained time units. Likewise, several schedule objects may be put into effect at the same time, thus enabling scheduling of concurrent, correlated activities.

A hierarchical schedule is implemented by first interpreting its root schedule object. Such an interpretation is performed by mapping the schedule into a set of elementary, rule-like "activities" to be evaluated by an evaluator, in similar fashion to that of a conventional inference engine. The evaluator continually cycles through its list of activities, checking whether the timing and preconditions of activities are satisfied, and executes the actions of those that are so triggered. Actions at the

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specfic permission.

© ACM 1988 0-89791-271-3/88/0006/1059 $1.50

lowest level are commands to sample environmental variables or to actuate control mechanisms. Actions at higher levels include carrying out the schedule replacements dictated by hierarchical schedules.

The AIDECS subsystem is written in SCHEME for PC or TI Explorer execution. It communicates with C language data acquisition and control software for interfacing to sensors and actuators. It affords much greater flexibility in schedule specification and online modification than would be possible in a pure C language (or other conventional programming approaches). The system design exemplifies general principles by which expert system architectures may be extended to include time-sensitive behaviors. This paper describes the representations employed to realize these principles and details in the overall design of the AIDECS subsystem.

SYSTEM ARCHITECTURE

The AIDECS consists of the Executive Expert System (EES), the Natural Language Interface (NLI), the Constraints Checker (CC), the Schedule Manager (SM), the Schedule Executor (SE), and the Real-Time CTRL & DAQ System (RTCDAS), as shown in Fig. 1.

The EES decides an optimal advisory schedule by asking the user a sequence of questions or by simulating the environment. The NLI enables a person to specify a time-based schedule to the environmental control system. The CC decides whether the schedule is acceptable or not by comparing it with a set of constraints contained within it, and sends it to the SM. The SM generates the specification of the schedule, translates the specification into a schedule object, and sends the object to the SE. The SE transforms the object into a set of activities, each of which has slots for a pair of condition and action, and others. The SE continually evaluates each activity in the set; if the condition of an activity is satisfied, the associated action in the activity is fired, which sends a micro-level control signal(s) to the RTCDAS. This signal has information on setpoint(s) such as temperature, and location within the BCELSS.

EXPERT SYSTEMS

Expert systems for the EES components are being developed using a consultation system shell (CESM, Classification Expert System Maker), which accepts an entity structure-like description of a classification hierarchy and writes a set of production rules to do the specified classification. The uncertainty management module of this system employs a modified version of

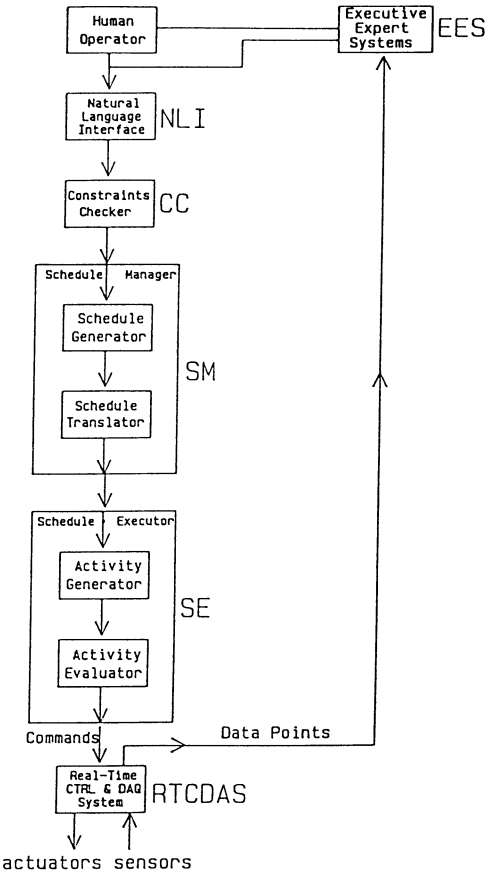


Fig. 1. The AI-Based Environmental Control System

Dempster-Shafer operators to provide adjustable evidence accumulation properties[2]. Forward Chaining, Backward Chaining, and Mixed modes of rule interpretation are provided. Graphic displays are easily incorporated to facilitate user comprehension. Since CESM is written in Scheme, it is readily integrated with other components in the AIDECS system. Commercial shells would present difficulties in this respect.

We have found that it is much easier to keep track of rules being developed for a consultation system when there is a graphical aid to organize the knowledge being employed. CESM can automatically generate rules from such a graphical description, thus saving much development effort and guaranteeing a greater degree of completeness (humans tend to think only in terms of hypothesis-confirming rules, omitting hypothesis-disconfirming rules that narrow the final classification).

For example, working with pest management experts, we developed an expert system for pest identification

and control. Concentrating first on aphids, the most important insect pest problem anticipated, the consultation system provides assistance in distinguishing aphids from other insects, recognizing important species of aphids, deciding what form of action is needed, and recommending environmental controls as appropriate. The system displays pictures of distinguishing body characteristics upon user request. The AIDECS call the pest management expert system whose recommendations may be checked by the CC accessing a crop tolerances knowledge base. The user may override the environmental schedule currently in force to implement emergency pest control environmental enhancers recommended by the pest management expert system.

NATURAL LANGUAGE INTEFACE

The NLI is designed to facilitate specifying a time-based schedule for an environmental control system with considerable freedom of expression. An Augmented Transition Network (ATN)[3] is used to implement the syntax and semantics of schedule specifications.

Consider a simplified example of a schedule specification:

from 1988/1/1 to 1989/12/31
in all plots in intensive-agriculture biome
for 24 hours period
maintain temperature at 70 degrees for first 12 hours
increase temperature by 5 degrees for next 12 hours.

We expect the ATN to produce the following variables with values assigned after parsing:

start-date = 1988/1/1
end-date = 1989/12/31
plot-type = all
biome-type = intensive-agriculture
period-value = 24
period-units = hours
control = (maintain increase)
parameter = (temperature temperature)
parameter-value = (70 5)
parameter-units = (degrees degrees)
duration-time-value = (12 12)
duration-time-units = (hours hours)

Subsequent modules can read the values of the variables and eventually issue corresponding instructions to set the thermostat of the heating-cooling system. Fig. 2

```
command -> when-phrase where-phrase period-phrase act&tims-phrase
         | where-phrase when-phrase period-phrase act&tims-phrase
  when-phrase -> start-phrase end-phrase
    start-phrase -> from-prep start-date
      from-prep -> from | beginning
      start-date -> [read a date]
    end-phrase -> to-prep end-date
      to-prep -> to | ending | until
      end-date -> [read a date]
  where-phrase -> plot-phrase biome-phrase
    plot-phrase -> locator determiner-phrase plot-type-phrase | empty
      locator -> in
      determiner-phrase -> determiner | empty
        determiner -> the
      plot-type-phrase -> plot-type plot-word
        plot-type -> all | plot0 plot-type | plot1 plot-type
                   | plot2 plot-type | plot3 plot-type | plot4 plot-type
        plot-word -> plots | empty
    biome-phrase -> locator determiner-phrase biome-type-phrase
      biome-type-phrase -> biome-type biome-word
        biome-type -> ia | intensive-agriculture | savannah | forest
        biome-word -> biome | empty
  period-phrase -> cycle-prep-phrase cycle-phrase
    cycle-prep-phrase -> cycle-prep cycle
      cycle-prep -> for
      cycle -> every | empty
    cycle-phrase -> period-time-value period-time-units period
      period-time-value -> [read a number]
      period-time-units -> sec | secs | min | mins | hour | hours
                         | hrs | hr | days | day
      period -> period | empty
  act&tims-phrase -> action-phrase duration-phrase act&tims-phrase
                   | duration-phrase action-phrase act&tims-phrase
    action-phrase -> control what parameter-phrase
      control-> maintain | increase | decrease
      what -> temperature | humidity
      parameter-phrase -> designator parameter-value parameter-units
        designator -> to | at | by
        parameter-value -> [read a number]
        parameter-units -> degrees | deg | degs | per-cent | %
    duration-phrase -> sequence-phrase duration-time-value duration-time-units
      seqence-phrase -> for-prep seq-word
        for-prep -> for
        seq-word -> first | next | last | empty
      duration-time-value -> [read a number]
      duration-time-units -> sec | secs | min | mins | hours
                           | hour | hrs | hr | days | day
```

Fig. 2. Context-free Grammar for Schedule Specification

shows the context-free grammar underelying the ATN design. As shown in the figure, humans have several ways to express the same information. For example, the rewrite rules for "command" permit any order of action and timing phrases. Other freedoms are less significant but equally user-friendly: alternative prepositions (to, until), abbreviations (hours, hrs), etc.

The parser reads a schedule specification either from the keyboard or from a file. It has a back-tracking capability that detects syntax errors and allows the user to go back to the point where he/she made an error and continue specifying the schedule from that point. It also has interactive help invoked by "?" that gives all possibilities for specification at the current point.

CONSTRAINTS CHECKER

The CC contains a set of constraints on environmental parameters such as temperature and humidity and tolerances for various crops. The CC compares the set of constraints with a set of values of variables corresponding to the specified schedule, set by the parser of NLI. Only the schedule that satisfies these constraints is sent to the Schedule Manager to prune the entity structure of the Timed Action Language (see below). For example, a

temperature schedule for a crop to "maintain temperature at 60 degrees for 4 hours" can be checked with such constraints as:

1) Is 60 degrees lower than temperature upper limit?
2) Is 60 degrees higher than temperature lower limit?
3) Is 4 hours longer than maximum exposure time at 60 degrees?
4) If so, does 60 degrees for 4 hours result in a tolerable loss of crop?

Violations of constraints are reported to the user who must decide (in conjunction with the EES) how to revise the designed schedule appropriately.

SCHEDULE MANAGER

The SM has two subsystems: a schedule generator and a schedule translator. The generator generates a schedule using the values of variables assigned by the parser of the NLI. To generate the schedule specification, we developed the Timed Action Language (TAL) using the concept of system entity structuring formalism[4]. As shown in Fig. 3, TAL can specify the following information for a schedule:

```
-ENT : TAL     ,attributes -> ((CO-ORDINATOR? ()))
--ASP : TAL-DEC
---ENT : WHERE     ,attributes -> ()
---ASP : WHERE-DEC
----ENT : INTENSIVE-AGRICULTURE     ,attributes -> ((MAX-PLOTS 5))
-----ENT : ALL-P     ,attributes -> ()
---ENT : START-DATE     ,attributes -> ((DATE 1987/11/1))
---ENT : END-DATE     ,attributes -> ((DATE 1988/12/31))
---ENT : CYCLE     ,attributes -> ((PERIOD 24) (UNIT HRS))
---ENT : ACT&TIMINGS     ,attributes -> ()
----ASP : ACT&TIMINGS-ASP
----ENT : BACKGROUND-ACTS     ,attributes -> ()
-----ASP : BACKGROUND-ACTS-DEC
------ENT : DAS     ,attributes -> ()
------ENT : SCREEN     ,attributes -> ()
----ENT : ACT&TIMS     ,attributes -> ((NUM-CHILDREN 9))
-----ASP : ACT&TIMS-ASP
------ENT : ACT&TIM0     ,attributes -> ()
-------ASP : ACT&TIM0-ASP
-------ENT : ACTION0     ,attributes -> ((PUT-IN-EFFECT ()))
--------ASP : ACTION0-ASP
--------ENT : HOW0     ,attributes -> ((ACTION MAINTAIN))
--------ENT : TEMP0     ,attributes -> ((RANGE 55))
------ENT : CONTINUOUS0     ,attributes -> ((HOW-LONG 5) (UNIT HRS))
-----ENT : ACT&TIM1     ,attributes -> ()
------ASP : ACT&TIM1-ASP
-------ENT : ACTION1     ,attributes -> ((PUT-IN-EFFECT ()))
--------ASP : ACTION1-ASP
--------ENT : HOW1     ,attributes -> ((ACTION MAINTAIN))
--------ENT : TEMP1     ,attributes -> ((RANGE 60))
-------ENT : CONTINUOUS1     ,attributes -> ((HOW-LONG 2) (UNIT HRS))
------ENT : ACT&TIM2     ,attributes -> ()
------ASP : ACT&TIM2-ASP
-------ENT : ACTION2     ,attributes -> ((PUT-IN-EFFECT ()))
--------ASP : ACTION2-ASP
--------ENT : HOW2     ,attributes -> ((ACTION MAINTAIN))
--------ENT : TEMP2     ,attributes -> ((RANGE 65))
-------ENT : CONTINUOUS2     ,attributes -> ((HOW-LONG 1) (UNIT HRS))
------ENT : ACT&TIM3     ,attributes -> ()
------ASP : ACT&TIM3-ASP
------ENT : ACTION3     ,attributes -> ((PUT-IN-EFFECT ()))
-------ASP : ACTION3-ASP
--------ENT : HOW3     ,attributes -> ((ACTION MAINTAIN))
--------ENT : TEMP3     ,attributes -> ((RANGE 70))
------ENT : CONTINUOUS3     ,attributes -> ((HOW-LONG 1) (UNIT HRS))
------ENT : ACT&TIM4     ,attributes -> ()
------ASP : ACT&TIM4-ASP
------ENT : ACTION4     ,attributes -> ((PUT-IN-EFFECT ()))
--------ASP : ACTION4-ASP
--------ENT : HOW4     ,attributes -> ((ACTION MAINTAIN))
--------ENT : TEMP4     ,attributes -> ((RANGE 75))
------ENT : CONTINUOUS4     ,attributes -> ((HOW-LONG 7) (UNIT HRS))
-----ENT : ACT&TIM5     ,attributes -> ()
------ASP : ACT&TIM5-ASP
```

Fig. 3. The Entity Strucuture for Timed Action Language

1. Where
   1.1 Which biome (e.g., {intensive-agriculture ...})
      1.1.1Which plots in a biome
         1.1.1.1 Which sectors in a plot
2. Start-date
3. End-date
4. Period
5. Action(s) and Timing(s)
   5.1 Action(s)
      5.1.1 What (e.g., {crop, temp, humidity....})
      5.1.2 How (e.g., {maintain increase....})
   5.2 Time(s) (e.g., {discrete, continuous....})

An actual schedule can be set by pruning the Entity Structure of the TAL with respect to the schedule specified by the user through the NLI. The pruning procedure assigns values of attributes in an entity and makes choices where alternatives exist for the same entity[5]. It also generates as many actions and corresponding timings (ACT&TIMS in Fig. 3) as the schedule specifies. For example, Fig. 8, a pruned entity structure, has biome type of intensive-agriculture and a sequence of 9 different actions and timings about temperature schedule as specified in the schedule of Fig. 7.

TAL specifies a schedule in a hierarchical manner. Each schedule has a sequence of intervals represented by sub-schedules; each sub-schedule in turn has its own sub-schedules, recursively. Such a capability allows the SM to change its schedule locally on any level of the hierarchy. It can also specify more than two different schedules in the same interval (called a "joint schedule"). For example, a temperature schedule and a humidity schedule can be specified in the same interval of a cycle, as shown in (13) through (22) and in (24) through (33) in Fig. 4.

The schedule translator in the SM translates the high-level schedule specification into low-level schedule object(s). The procedure of translation is as follows:

1. Create space object
   1.1 Create biome object(s)
   1.2 Create plot object(s)
   1.3 Create sector object(s)
2. Attach sector(s) to plot(s), and plot(s) to biome(s)
3. Create schedule object(s)
4. Attach the schedule object(s) to space object(s) including biome(s), plot(s), and sector(s)

```
For all plots in Intensive-Agriculture Biome
preform the following schedule from 1988/1/1 to 1990/12/31

(1) for every year period
(2)     for first 4 months
(3)         for every 2 months period
(4)             for first 20 days
(5)                 for every 1 day period
(6)                     for first 8 hours, maintain temp at 50
(7)                     for next  8 hours, maintain temp at 55
(8)                     for last  8 hours, maintain temp at 60
(9)             for next 40 days
(10)                for every 1 day  period
(11)                    for first 12 hours, maintain temp at 50
(12)                    for next 12 hours, maintain temp at 60
(13)    for next 4 months  perform following 2 schedules
(14)        schedule 1 for temp:
(15)            for every 1 day period
(16)                for first 12 hours, maintain temp at 60
(17)                for next 12 hours,  maintain temp at 65
(18)        schedule 2 for humid:
(19)            for every 1 month period
(20)                for first 10 days, maintain humi at 65
(21)                for next 10 days,  maintain humi at 70
(22)                for last 10 days,  maintain humi at 75
(23)    for last 4 months
(24)        for every 2 months period
(25)            for first 20 days perform following 2 schedules
(26)                schedule 1 for temp:
(27)                    for every 1 day  period
(28)                        for first 18 hours, maintain temp at 70
(29)                        for next 6 hours,  maintain temp at 75
(30)                schedule 2 for humid:
(31)                    for every 1 day  period
(32)                        for first 12 hours, maintain humi at 65
(33)                        for next 12 hours,  maintain humi at 70
(34)            for next 40 days
(35)                for every 20 days period
(36)                    for first 10 days, maintain temp at 65
(37)                    for next 10 days,  maintain temp at 70
```

Fig.4. A Hierarchical Schedule Specification

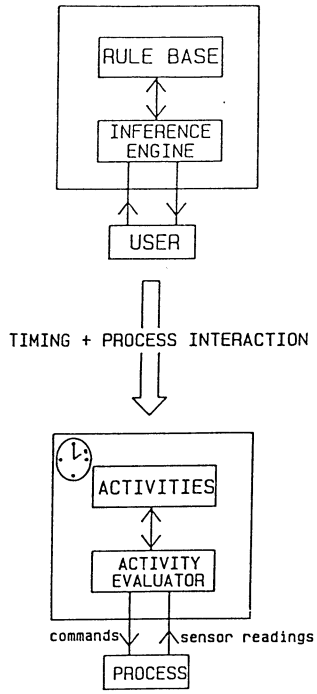


Fig. 5. Upper: A Rule-Based Expert System Architecture
Lower: A Schedule Excutor Architecture

SCHEDULE EVALUATOR

The concept of SE evolved from the canonical architecture of rule-based expert systems as shown in Fig.5. We see that the if-then rules are replaced by activities containing a special section for time management. The Inference Engine is called an Activity Evaluator (AE) since inferencing is now only a part of the effect of rule evaluation. The AE achieves its rule evaluation by referencing the clock and the timing section of activities, in addition to the usual rule conditions. The user is replaced by the real world process to be managed. The usual question and response interaction between the Inference Engine and the user is replaced by a more complex interaction in which the AE exerts actions upon the process and receives sensorial feedback from it. The structure of an activity is as shown in Fig. 6.

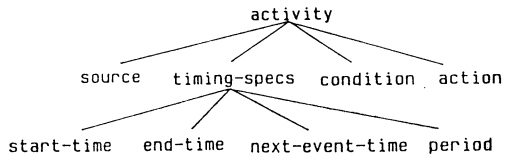
An activity will be fired, i.e., its action part will be executed by the AE, under the following circumstances:

1) its timing requirements are met
2) its condition is satisfied
3) the activities of its source schedule are not overriden.

In one cycle of rule evaluation, more than one activity can be fired. This is similar to the classifier systems approach[6] and contrasts with the usual inference engine cycle in which a conflict resolution scheme selects only one rule to fire of those that are triggered.

As depicted in Fig. 6, in each cycle the AE scans the current set of activities and writes a new set of activities for the next cycle. At the start of a cycle, the AE reads the clock. Then for each activity, it may perform one or more of the following actions:

- fire the activity: if the above conditions 1), 2) and 3) hold
- reschedule the activity: modify its timing specification and place it in the activities set of the next cycle. This occurs whenthe timing requirements 1) are satisfied, whether or not requirements 2) or 3) are also met.
- retain the activity: place it unmodified in the activities set of the next cycle. This occurs when the timing requirements 1) are not met.
- remove the activity: do not retain the activity for the activities set of the next cycle. This occurs when the timing requirements 1) have become impossible to satisfy.
- handle user interrupt: respond to the user interrupt by calling the EES for consultation to possibly override the current schedule in the period to be specified.



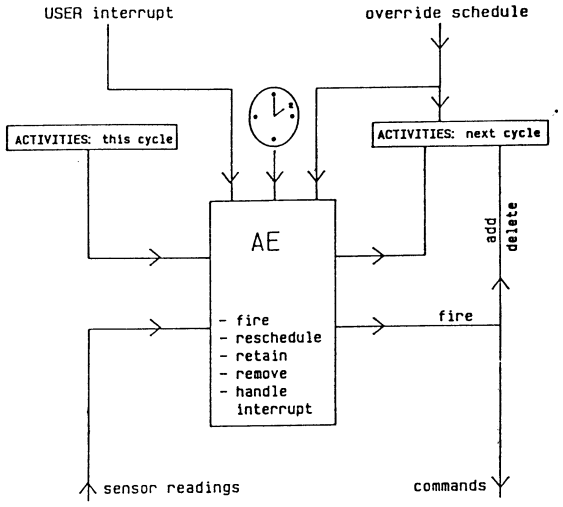


Fig. 6. Upper: Structure of Activity
Lower: Overview of Activity Evaluation

Note that the action part of a fired activity may cause additions or deletions of activities for the next cycle. This reprogramming of the activity set is a powerful capability that is employed in hierarchical scheduling (see Fig. 4) among other possibilities.

In more detail, the timing requirements of an activity are met if all the following hold:

1) its starting time has arrived, i.e.,
   start-time < = current-time
2) its life time has not expired, i.e.,
   current-time < = end-time
3) its next-event-time has arrived, i.e.,
   current-time = next-event-time.

When an activity is rescheduled, its next-event-time is increased by its period. That is, the next-event-time slot is modified by next-event-time : = (next-event-time + period) and is placed in the activities for the next cycle. Because rescheduling occurs when the timing requirements are met, an activity gets a chance to be fired periodically while it is alive. An activity is removed from consideration when its life time has expired, i.e., when current-time end-time.

TEST OF AIDECS

The AIDECS under development has been tested on the real-time basis under a PC/AT-based computer network. Two computers have been used for two dedicated jobs: one (computer 1) for the RTCDAS component (Fig. 1)

;;;; temp and humid schedule for winter ;;;;

from 1987/11/1 to 1988/12/31
in all plots in intensive-agriculture biome
for every 24 hrs period
    maintain temperature at 55 degs for first 5 hrs
    maintain temperature at 60 degs for next 2 hrs
    maintain temperature at 65 degs for next 1 hrs
    maintain temperature at 70 degs for next 1 hrs
    maintain temperature at 75 degs for next 7 hrs
    maintain temperature at 70 degs for next 1 hrs
    maintain temperature at 65 degs for next 1 hrs
    maintain temperature at 60 degs for next 1 hrs
    maintain temperature at 55 degs for last 5 hrs

    maintain humidity at 70 % for 24 hrs.

Fig. 7. Actual Schedule Tested in Real-Time Basis

This File is accepted by the Parser in NLI

```
-ENT : TAL    ,attributes -> ((CO-ORDINATOR? ()))
--ASP : TAL-DEC
--ENT : WHERE    ,attributes -> ()
---ASP : WHERE-DEC
----ENT : INTENSIVE-AGRICULTURE    ,attributes -> ((MAX-PLOTS 5))
----ENT : ALL-P    ,attributes -> ()
--ENT : START-DATE    ,attributes -> ((DATE 1987/11/1))
--ENT : END-DATE    ,attributes -> ((DATE 1988/12/31))
--ENT : CYCLE    ,attributes -> ((PERIOD 24) (UNIT HRS))
--ENT : ACT&TIMINGS    ,attributes -> ()
---ASP : ACT&TIMINGS-ASP
----ENT : BACKGROUND-ACTS    ,attributes -> ()
-----ASP : BACKGROUND-ACTS-DEC
------ENT : DAS    ,attributes -> ()
------ENT : SCREEN    ,attributes -> ()
----ENT : ACT&TIMS    ,attributes -> ((NUM-CHILDREN 9))
-----ASP : ACT&TIMS-ASP
------ENT : ACT&TIM0    ,attributes -> ()
-------ASP : ACT&TIM0-ASP
--------ENT : ACTION0    ,attributes -> ((PUT-IN-EFFECT ()))
---------ASP : ACTION0-ASP
----------ENT : HOW0    ,attributes -> ((ACTION MAINTAIN))
----------ENT : TEMP0    ,attributes -> ((RANGE 55))
---------ENT : CONTINUOUS0    ,attributes -> ((HOW-LONG 5) (UNIT HRS))
------ENT : ACT&TIM1    ,attributes -> ()
-------ASP : ACT&TIM1-ASP
--------ENT : ACTION1    ,attributes -> ((PUT-IN-EFFECT ()))
---------ASP : ACTION1-ASP
----------ENT : HOW1    ,attributes -> ((ACTION MAINTAIN))
----------ENT : TEMP1    ,attributes -> ((RANGE 60))
---------ENT : CONTINUOUS1    ,attributes -> ((HOW-LONG 2) (UNIT HRS))
------ENT : ACT&TIM2    ,attributes -> ()
-------ASP : ACT&TIM2-ASP
--------ENT : ACTION2    ,attributes -> ((PUT-IN-EFFECT ()))
---------ASP : ACTION2-ASP
----------ENT : HOW2    ,attributes -> ((ACTION MAINTAIN))
----------ENT : TEMP2    ,attributes -> ((RANGE 65))
---------ENT : CONTINUOUS2    ,attributes -> ((HOW-LONG 1) (UNIT HRS))
------ENT : ACT&TIM3    ,attributes -> ()
-------ASP : ACT&TIM3-ASP
--------ENT : ACTION3    ,attributes -> ((PUT-IN-EFFECT ()))
---------ASP : ACTION3-ASP
----------ENT : HOW3    ,attributes -> ((ACTION MAINTAIN))
----------ENT : TEMP3    ,attributes -> ((RANGE 70))
---------ENT : CONTINUOUS3    ,attributes -> ((HOW-LONG 1) (UNIT HRS))
------ENT : ACT&TIM4    ,attributes -> ()
-------ASP : ACT&TIM4-ASP
--------ENT : ACTION4    ,attributes -> ((PUT-IN-EFFECT ()))
---------ASP : ACTION4-ASP
----------ENT : HOW4    ,attributes -> ((ACTION MAINTAIN))
----------ENT : TEMP4    ,attributes -> ((RANGE 75))
---------ENT : CONTINUOUS4    ,attributes -> ((HOW-LONG 7) (UNIT HRS))
------ENT : ACT&TIM5    ,attributes -> ()
-------ASP : ACT&TIM5-ASP
--------ENT : ACTION5    ,attributes -> ((PUT-IN-EFFECT ()))
---------ASP : ACTION5-ASP
----------ENT : HOW5    ,attributes -> ((ACTION MAINTAIN))
----------ENT : TEMP5    ,attributes -> ((RANGE 70))
---------ENT : CONTINUOUS5    ,attributes -> ((HOW-LONG 1) (UNIT HRS))
------ENT : ACT&TIM6    ,attributes -> ()
-------ASP : ACT&TIM6-ASP
--------ENT : ACTION6    ,attributes -> ((PUT-IN-EFFECT ()))
---------ASP : ACTION6-ASP
----------ENT : HOW6    ,attributes -> ((ACTION MAINTAIN))
----------ENT : TEMP6    ,attributes -> ((RANGE 65))
---------ENT : CONTINUOUS6    ,attributes -> ((HOW-LONG 1) (UNIT HRS))
------ENT : ACT&TIM7    ,attributes -> ()
-------ASP : ACT&TIM7-ASP
--------ENT : ACTION7    ,attributes -> ((PUT-IN-EFFECT ()))
---------ASP : ACTION7-ASP
----------ENT : HOW7    ,attributes -> ((ACTION MAINTAIN))
----------ENT : TEMP7    ,attributes -> ((RANGE 60))
---------ENT : CONTINUOUS7    ,attributes -> ((HOW-LONG 1) (UNIT HRS))
------ENT : ACT&TIM8    ,attributes -> ()
-------ASP : ACT&TIM8-ASP
--------ENT : ACTION8    ,attributes -> ((PUT-IN-EFFECT ()))
---------ASP : ACTION8-ASP
----------ENT : HOW8    ,attributes -> ((ACTION MAINTAIN))
----------ENT : TEMP8    ,attributes -> ((RANGE 55))
---------ENT : CONTINUOUS8    ,attributes -> ((HOW-LONG 5) (UNIT HRS))
end of display
```

Fig. 8. Puned Entity Structure of Fig. 3 with respect the Temperature Schedule in Fig. 7

using C language, and the other (computer 2) for EES, NLI, CC, SM, and SE using SCHEME language. Computer 2 was used to specify schedule shown in Fig. 7 through the NLI. The specification accepted by the CC was sent to the SM which generated the pruned entity structure of the TAL (Fig. 8) with respect to the schedule, which in turn was transformed into schedule object(s) by the SE. The SE successfully sent commands to the RTCDAS in computer 1. The RTCDAS, having a set of control software written by C, issued micro-level signals to the control devices. Conversely, data acquired by the RTCDAS was been periodically sent to computer 2, where the EES stores current data for late use. Work is continuing to install and test the AIDECS in a prototype life support system.

REFERENCES

1. Proceedings of IEEE International Symposium on In telligent Control, edited by A. Meystel and J.Y.S. Luh, Jan. 1987, Phil, PA.

2. B.P. Zeigler, "Some Properties of Modified Dempster-Shafer Operations in Rule-Based Inference System," Tech Report AIS-5, Electrical and Computer Engineering Department, University of Arizona, April, 1987.

3. P.H. Winston, Artificial Intelligence, 2nd ed. Additional Wesley, Reading, MA., 1984

4. B.P. Zeigler, Multifaceted Modeling and Discrete Event Simulation, Academic Press, London and Orlando, FL., 1984.

5. B.P. Zeigler, "Knowledge Representation From Newton to Minsky and Beyond," Applied Artificial Intelligence, vol.1 87-107, Hemisphere Pub. Co., 1987.

6. J.H. Holland, K.J. Holyoak, R.E. Nesbit and P.R. Thagard, Induction: Process of Inference, Learning, and Discovery, The MIT Press, Cambridge MA, 1986.