HIERARCHICAL SCHEDULING IN AN INTELLIGENT ENVIRONMENTAL CONTROL SYSTEM

Tag Gon Kim
ERLab, University of Arizona

Bernard P. Zeigler
Dept. of Electrical and Computer Engineering, University of Arizona
Tucson, AZ 85721

ABSTRACT

This paper describes design and implementation of the hierarchical scheduling subsystem and graphical user interface in an intelligent environmental control system. The hierarchical scheduling system is capble of managing all environmental events occuring in widely different time scales as specified by the user. We employ a knowledge representation scheme called system entity structure to specify the environmental schedules in a hierarchical fashion. A system entity structure called TAL (Timed Action Language) is developed. TAL organizes a family of all possible long-/mid-/short-term schedules from which a specific schedule can be pruned by the user through graphical interface.

INTRODUCTION

An intelligent environmental control system must be capable of managing environmental activities and conditions in integrated fashion. Such activities range from planning of agricultural activities at highest level to real-time control of environmental parameters at the lowest level. The activities at different levels gives rise to hierarchical schedules which facilitate timing of events to occur at different time scales. Design of such an intelligent control system involves disciplines such as artificial intelligence, automatic control, operations research, discrete-event system theory, and others [1].

This paper describes a hierarchical scheduling subsystem in such an intelligent environmental control system called AIDECS (the AI-Based, Distributed Environmental Control System) which was developed at University of Arizona as outlined in [2]. The AIDECS deals with control over environmental parameters such as temperature and humidity as well as scheduling crop planting and harvesting. The scheduling subsystem in the AIDECS supports specification of a hierarchical schedule, transforms it into a schedule object, and eventually maps it into rule-like activities which are continuously evaluated by an expert system-like evaluator.

The entity structure – a knowledge representation scheme using a labeled tree with attached variable types [5] – is employed to specify time-based schedules in hierarchical fashion. The entity structure, originally used to represent structure knowledge of a system with a family of variant models, can represent sub-components, decomposition, and taxonomic information about a system. In a hierarchical schedule specification, a system entity structure organizes a family of possible schedules from which a particular schedule can be selected by the user through the *pruning* operation. More specifically, our approach to hierarchical schedule specification is: 1) we define a system entity structure which organizes a family of all possible schedules called TAL (Timed Action Language), 2) the user specifies a schedule through the user interface system (either natural language interface or graphical interface), 3) a scheduling system prunes the TAL using information generated by the user interface, and 4) the pruned entity structure is schedule specification. It is transformed into a set of rules for a real-time evaluator.

The paper is organized as follows. First it gives a short review of the system entity structure. It then presents the architecture of the AIDECS and briefly explains its subsystems. The paper presents details of the graphical user interface system and hierarchical scheduling subsystem written in Scheme LISP for execution on PC/ATs or TI Explorers. The conclusion of the paper outlines further work on the AIDECS, particularly on the schedule optimization subsystem.

THE SYSTEM ENTITY STRUCTURE

System entity structure (SES) is a representation scheme which contains the decomposition, coupling, and taxonomy information for a system [5], [6]. Formally, SES is a labeled

Permission to copy without fee all or part of this matertial 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 the copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1989 ACM 0-89791-320-5/89/0006/0310 $1.50

tree with attached variable types that satisfies five axioms – alternating mode, uniformity, strict hierarchy, valid brothers, and attached variables. Detail description of the axioms is available in [5].

There are three types of nodes in SES – *entity, aspect,* and *specialization* – which represent three types of knowledge about systems. The *entity* node, having several *aspects* and/or *specializations*, corresponds to a model component that represents a real world object. The *aspect* node (a single vertical line in the labeled tree of Figure 1) represents one *decomposition*, out of many possible, of an entity. Thus, the children of an aspect node are entities, distinct components of the decomposition. The *specialization* node (a double vertical arrows in the labeled tree of Figure 1) represents a way in which a *general* entity can be specialized into *special* entities.

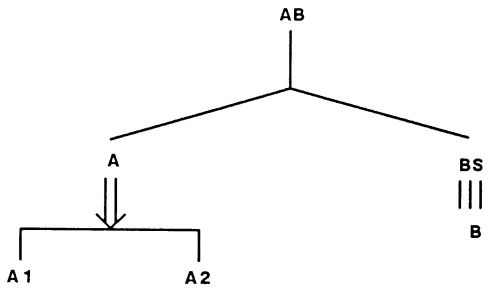

Figure 1. A system Entity Structure.

A *multiple entity* is an entity that represents a collection of homogeneous components. We call such components a *multiple decomposition* of the multiple entity. The aspect of such a multiple entity is called *multiple aspect* (triple vertical lines in the labeled tree of Figure 1). Note that instead of presenting all Bs for BS's components, only one B is placed in the labeled tree.

Pruning extracts a sub-structure of the SES by selecting one aspect and/or one specialization for each entity in the SES. The *pruning* operation also expands multiple entities as well as values of attributes attached to entities in the SES.

SES has been realized in Scheme LISP environment. The realization called ESP-Scheme was described in [3]. ESP-Scheme is originally used as a means of representing structural knowledge for a simulation model in knowledge-based modelling/simulation research. Here, we use ESP-Scheme as a tool for specification of hierarchical schedules in the AIDECS.

SYSTEM ARCHITECTURE OVERVIEW

The AIDECS (Figure 2) consists of Consultation Expert System (CES), User Interface System (UIS), Constraints Checker (CC), Schedule Manager (SM), Schedule Executor (SE), Schedule Decision Expert System (SDES), Control Data Base (CDB), and a Real-Time Control and Data Acquisition System (RTCDAS). The CES contains the Integrated Pest Management Expert System (IPMES) and the Diagnostic Expert System (DES).

The UIS enables the user to specify a short-/mid-/long-term schedule for environmental activities and parameters in hierarchical manner. 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 a hierarchical 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 in conjunction with the SDES; 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) for environmental parameters such as temperature, and location where control actions are taken place. Data points that the RTCDAS collects by sensors are maintained in the CDB which the CES and the SDES might access as required.

The following sections provide details of graphical user interface subsystem and of the schedule generator within the SM subsystem. Description of rest of the components of the AIDECS can be found in [2], [4].

GRAPHICAL USER INTERFACE

Two different types of user interfaces have been developed: natural language interface (NLI) and graphical interface (GI). There exists an one-to-one correspondence between information generated by the NLI and by the GI. The NLI facilitates specifying a time-based schedule with considerable freedom of expression. An augmented transition network is used to implement the syntax and semantics of schedule specification. The NLI, including its grammar, was reported in [2].

The GI is designed to facilitate specifying, showing, modifying, or executing a time-based schedule by help of a mouse. Developed in SCOOPS, an object-oriented programming superset of SCHEME, the GI employs a hypertext approach. The input of the GI is novel in that the user can draw curves graphically by a mouse as well as texts by a keyboard for a schedule specification as shown in Figure 3. The output information generated by the GI is used for automatic pruning of an entity structure representing a family of schedules.

Classes in the GI include a text menu class and a graphic menu class, each of which has its sub-classes. Instance variables and methods for those classes are provided to

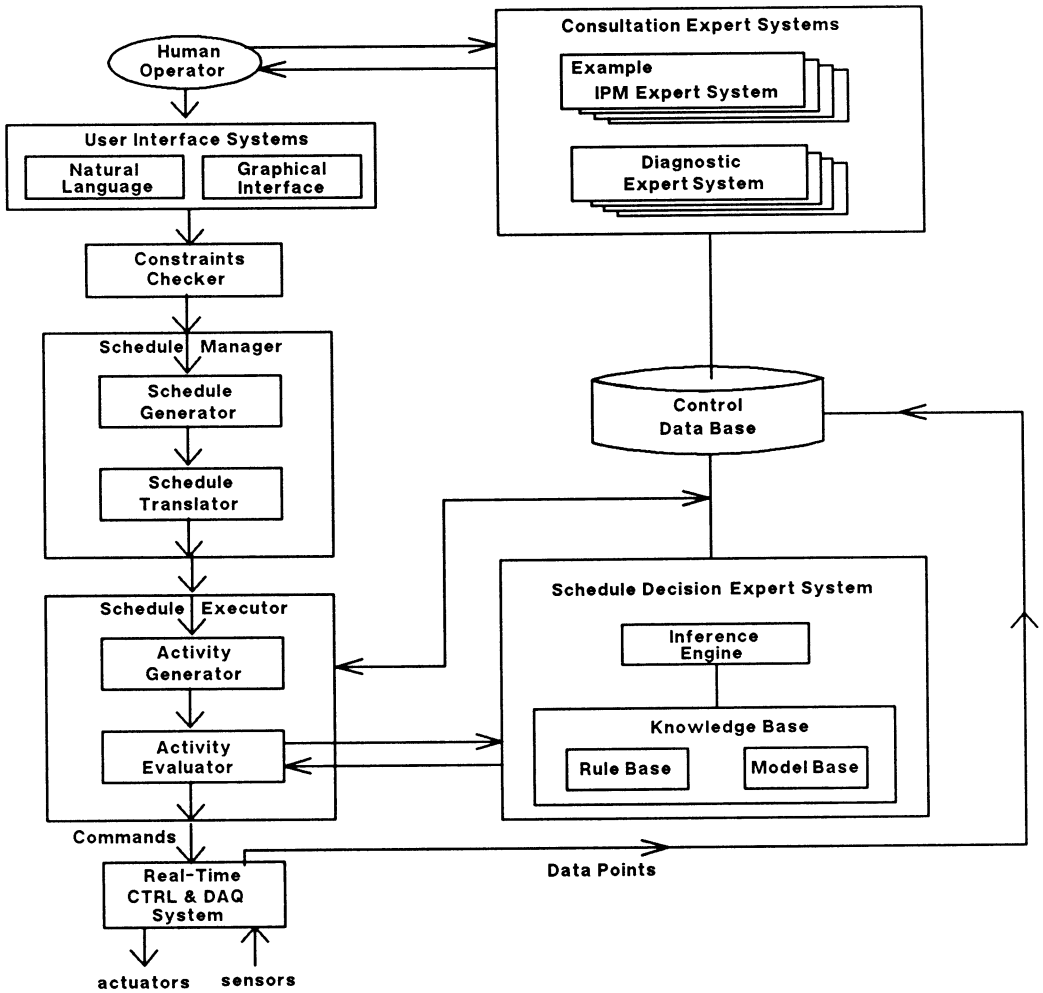

Figure 2. Overall System Architecture.

display, read, or select text menus, draw curves, and save specified information. More specifically, a method *when-receive-click* of graphic menu class performs the following:

- translates a current mouse position into a value of a control variable, and saves it in an instance variable,
- displays the current position, and
- draws a line from a previous position to the current one.

Similarly, the method *when-receive-click* of text menu class is to read a text from a keyboard, display it, and save in an instance variable.

A menu *specify* in Figure 3 displays questions about a schedule one by one to the user, text menus followed by graphic menus. Specifying text information for the schedule consists of answering by typing texts at mouse positions and/or selecting texts displayed in a screen using the mouse. Such information includes start-date, end-date, biome-type, plot-type, and others for the schedule.

Objects of the graphic menu class are used for specifying graphic information about control of environmental variables such as temperature and humidity. Specifying such information involves imputing curves graphically which assign a values of environmental variables and their associated time intervals. The schedule specified can be saved as required or discarded.

A daily schedule for temperature and humidity is shown in Figure 3. The natural language version of schedule specification is as follows.

From 1989/6/1 to 1991/5/31
for all PLOTS in IPM-GREENHOUSE Biome
for every 24 hours period

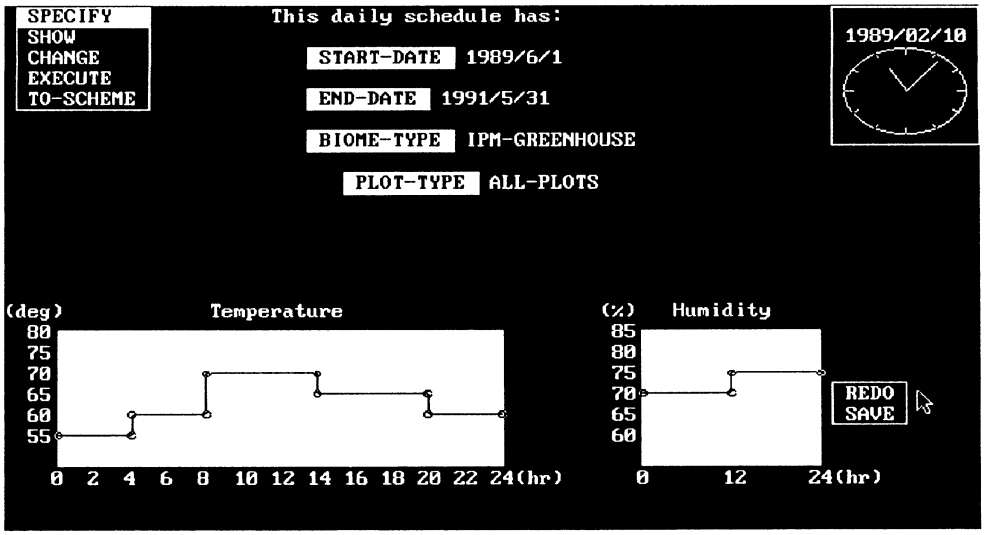

Figure 3. Graphical User Interface.

maintain temperature at 55 degrees for first 4 hrs
60 degrees for next 4 hrs
70 degrees for next 6 hrs
65 degrees for next 6 hrs
60 degrees for last 4 hrs

maintain humidity at 70 % for first 12 hrs
75 % for last 12 hrs.

The GI produces a set of variables with their values assigned as its output for the above input. Such a set of variables and their values, as shown in Figure 4, is used to generate entity structures for corresponding schedules. We shall explain details of such a schedule generation by automatic pruning processes in the following section.

The menu *show* shows schedules saved in a library called schedule base. The schedule shown can be executed as desired. The menus *change* and *execute* can change and execute a schedule shown in a screen, respectively.

```
start-date = 1989/6/1
end-date = 1991/5/31
biome-type = IPM-GREENHOUSE
plot-type = ALL
period = (24)
period-unit = (hrs)
control = ((maintain maintain maintain maintain maintain)
          (maintain maintain))
parameters = ((temperature temperature temperature temperature
          temperature) (humidity humidity))
parameter-values = ((55 60 70 65 60) (70 75))
parameter-units = ((deg deg deg deg deg) (% %))
duration-values = ((4 4 6 6 4) (12 12))
duration-units = ((hrs hrs hrs hrs hrs) (hrs hrs)).
```

Figure 4. Output of Graphic Interface.

SCHEDULE GENERATOR

The schedule generator (SG), a subsystem of the schedule manager, generates hierarchical schedules which facilitate timing of events to occur at coarse- and fine-grained time and space units. An entity structure organizes a family of all possible schedules by its representation scheme; it can represent components, decompositions, and taxonomic knowledge of a schedule. The operation pruning defined on the entity structure allows the SG to generate a pruned entity structure representing a schedule specified by the user.

We develop, using ESP-Scheme, an entity structure called Timed Action Language (TAL) to organize a family of all possible schedules for the environmental control. As shown in Figure 5, entities, decompositions, and specializations with attached attributes in the TAL represent all necessary information for the SG to generate user-specified schedules. For example, *entities* WHERE, START-DATE, END-DATE, CYCLE, and ACT&TIMS along with their attached attributes designate area, start date, end date, period, and a sequence of control actions with associated time intervals for a schedule, respectively. A *specialization* BIOME-SPEC specifies that an area can be one of three types, namely, FOREST, SAVANNAH, or IPM-GREENHOUSE, each of which has its own plots, each of which has its own sectors. Such specialization allows the user to adjust sensitivity of areas in which control actions are taken place for the specified duration.

Note that the TAL has a multiple entity ACT&TIMS whose child entity ACT&TIM specifies a control action with an associated time interval. Since the pruning opera-

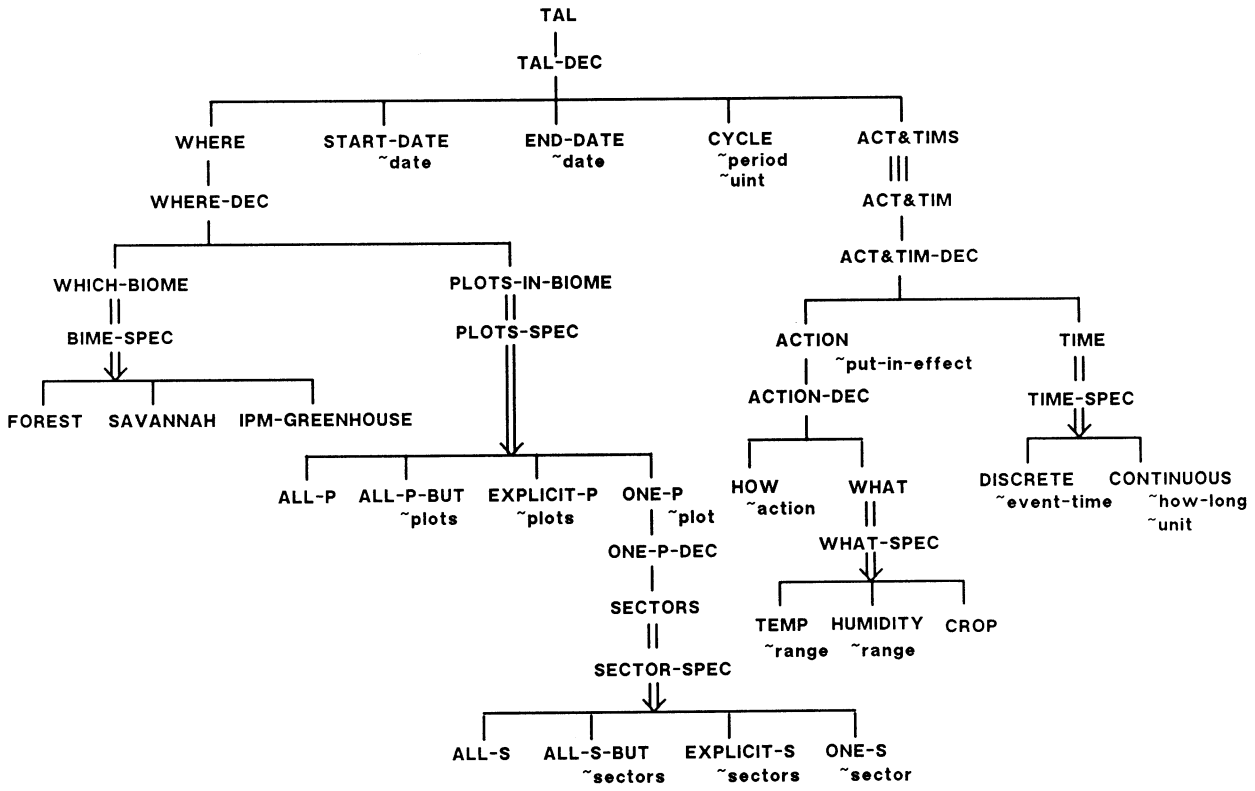

Figure 5. Entity Structure of Timed Action Language (TAL).

tion on a multiple entity can generate unlimited number of multiple children entities, the user can specify any sequences of control actions and their associated time intervals.

A schedule is specified in a hierarchical manner. Such a hierarchical schedule is represented by a set of pruned entity structures organized in hierarchical fashion. An attribute *put-in-effect* attached to the entity ACTION is used to maintain an ordered list of entity structures for an entity structure, which represent sub-schedules to be executed for a sequence of specified intervals. Each sub-schedule, in turn, has its own sub-schedules, recursively. Such a capability allows the SG to generate hierarchical schedules on any level of hierarchy in time units. On the other hand, putting into effect more than one schedule with the same interval results in a joint schedule, a set of schedules being executed together such as temperature and humidity schedules.

Specification of a hierarchical schedule can take advantage of information inheritance; an entity structure inherits all information specified for a schedule except cycle and action and associate timing to its sub-entity structures. For example, a start date specified in an entity structure is inherited to all its sub-structures as default. However, a new start date, if specified in the sub-structure(s), can override the inherited one.

PRUNING: SCHEDULE SPECIFICATION

In a hierarchical structure, pruned entity structures with no sub-structures specify actual control actions, while those with sub-structures maintains a list of the sub-structures with their associated timing intervals. To explain how such actual control actions are specified by pruning the TAL, let us consider a schedule specified in Figure 3 of the GI. The Figure shows a daily joint schedule to control both temperature and humidity.

The SG inputs information in Figure 4 and prunes the TAL. The SG generates two pruned entity structures for temperature and humidity schedules, respectively. Let us call the pruned entity structures for the temperature control p:temp and for the p:humi. To obtain the p:temp shown in Figure 6 involves the following procedures:

- assign values of attributes attached to the entities,
- select specialized entities under generalized ones,
- generate five ACT&TIMs for five control actions, and
- specify generated ACT&TIMs by assigning values of attributes and selecting entities under each such ACT&TIM entity.

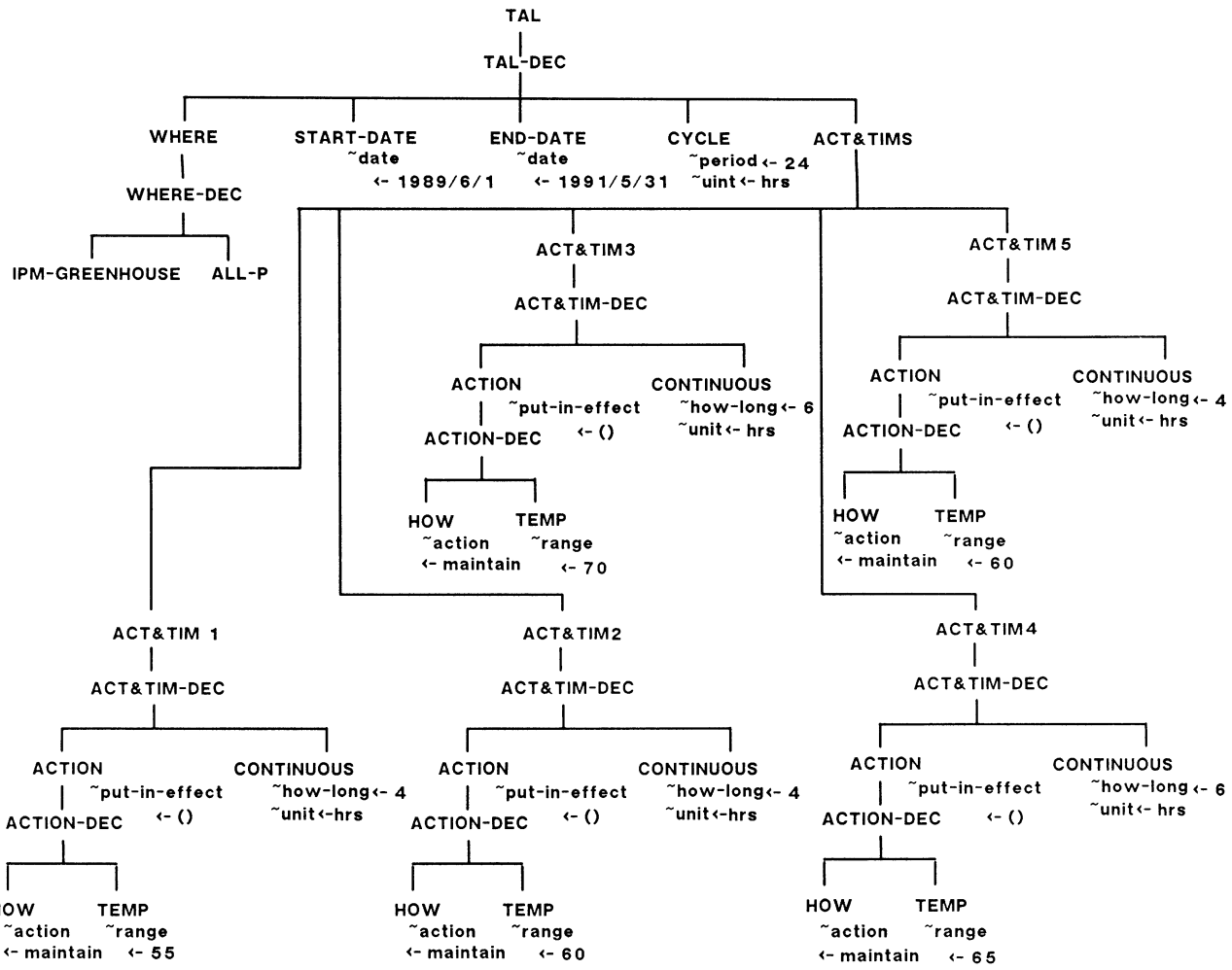

Figure 6. Pruned Entity Structure of TAL for the Schedule Shown in Figure 3.

More specifically, 1989/6/1 in Figure 4 is assigned as the value of the attribute date attached to the entity START-DATE for the p:temp. Similarly, the above values of end-date, period, and period-unit are assigned to values of attribute date attached to the entity END-DATE, and attributes period and unit attached to the entity CYCLE, respectively. The values IPM-GREENHOUSE of biome-type and ALL of plot-type above are used to select entities IPM-GREENHOUSE and ALL-P for the p:temp under BIOME-SPEC and PLOTS-SPEC specializations in the TAL, respectively.

Length five of the first list in the values of variable control in Figure 4, is used to generate five ACT&TIMs for p:temp under the multiple entity ACT&TIMS. A set of the list of five values of the variables in Figure 4 – control, parameters, parameter-values, parameter-units, duration-values, and duration-units – is used to specify five temperature control actions for p:temp. Figure 6 shows all such assigned values of attributes for all entities including those under five ACT&TIMs. Similarly the humidity entity structure, p:humi, can be obtained by using the values of variables in Figure 4.

HIERARCHICAL SCHEDULE SPECIFICATION

We now show how a hierarchical schedule is specified by hierarchically constructing pruned entity structures, such as the p:temp and p:humi, of the TAL. Consider a hierarchical schedule shown in Figure 7.

The schedule has a period of a year, for which three sub-schedules with their associated effective intervals are specified, each of which has its own sub-periods for which its sub-schedules are specified and so on. Note that there are six actual control actions for temperature and/or humidity, which requires six pruning processes. For example, lines (7)-(10) are represented by a pruned entity

structure for the temperature control. We call the six pruned entity structures p:temp1, p:temp2, p:temp3, p:humi1, p:humi2, and p:tem4 accounting for lines (7)-(10), (12)-(14), (16)-(18), (19)-(22), (25)-(27), and (29)-(32), respectively. More specifically, the pruned entity structure p:temp1 has three ACT&TIMs entities representing three temperature control actions with associated timings shown in lines (8), (9), and (10).

```
(1)  For all plots in IPM-GREENHOUSE
(2)  from 1989/7/1 to 1991/6/31, perform the following schedule
(3)  for every year period
(4)     for first 4 months
(5)        for every month period
(6)           for first 15 days
(7)              for every 1 day period
(8)                 for first 8 hrs, maintain temperature at 50 degs
(9)                 for next 8 hrs, maintain temperature at 60 degs
(10)                for last 8 hrs, maintain temperature at 70 degs
(11)          for last 15 days
(12)             for every 1 day period
(13)                for first 12 hrs, maintain temperature at 55 degs
(14)                for last 12 hrs,  maintain temperature at 65 degs
(15)  for next 4 months period perform a following joint schedule
(16)     for every 1 day period
(17)        for first 12 hrs, maintain temperature at 65 degs
(18)        for last 12 hrs,  maintain temperature at 75 degs
(19)     for every 1 month period
(20)        for first 10 days, maintain humidity at 70 %
(21)        for next 10 days,  maintain humidity at 75 %
(22)        for last 10 days,  maintain humidity at 80 %
(23)  for last 4 months
(24)     for first 2 months
(25)        for every 10 days period
(26)           for first 5 days, maintain humidity at 70 %
(27)           for last 5 days,  maintain humidity at 80 %
(28)     for last 2 months
(29)        for every 1 day period
(30)           for first 10 hrs, maintain temperature at 70 degs
(31)           for next 5 hrs, maintain temperature at 75 degs
(32)           for last 9 hrs, maintain temperature at 80 degs.
```

Figure 7. Hierarchical Schedule Specification.

The lines (4), (15), and (23) indicate that the overall schedule has three sub-schedules, each of which lasts for four months. To specify the sub-schedules needs additional three pruned entity structures. We call such pruned entity structures p:schedule1, p:schedule2, and p:schedule3 accounting for lines (4)-(14), (16)-(18), and (23)-(32), respectively. Another pruned entity structure, called p:schedule as a root, organizes the three sub-schedules. The pruned entity structure p:schedule has all information necessary for the schedule such as start date and end date specified in the line (2), area specified in the line (1), and others. Recall the inheritance property among hierarchically constructed pruned entity structures. P:schedule inherits such information to p:schedule1, p:schedule2, and p:schedule3, which inherit the information to their sub-entity structures listed above. Figure 8 shows the hierarchical structure of the pruned entity structures for the schedule of Figure 7. Note that P:temp3 and p:humi1 are in the same

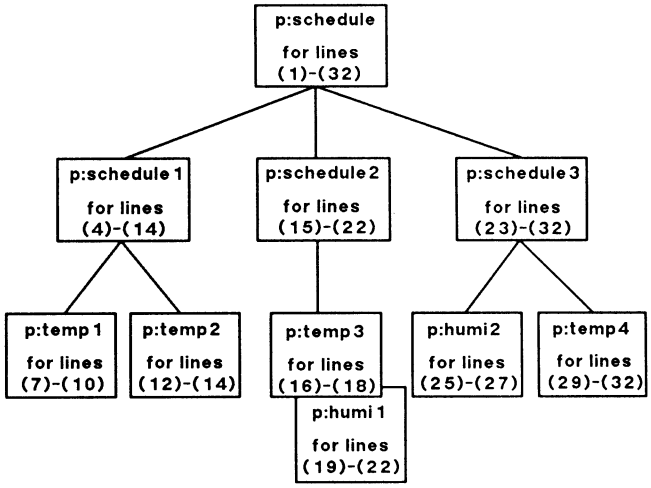

Figure 8. Pruned Entity Structures for Figure 7. Line numbers are those in Figure 7.

interval as a joint schedule, while others are in different intervals. The hierarchical structure of Figure 8 shows that change of schedule can be made locally on any level of the hierarchy in time units.

CONCLUSIONS

We have described subsystems for hierarchical scheduling in the AIDECS under development. Particularly, the graphical user interface and schedule generator written in Scheme have been discussed. The graphical user interface, developed in the object-oriented programming superset of SCHEME, allows the user to specify schedule using hypertexts. The schedule generator generates hierarchical schedules by pruning the TAL with respect to user-specified information output by the graphical user interface. Within the AIDECS hierarchical schedules have been tested on the real-time basis using two PC/ATs coupled in a PC/AT-based computer network.

Development of a real-time decision-making system for schedule optimization is under consideration. The approach to schedule optimization for such a expert system is to combine rule-based expert system techniques for static (or long-term) optimization with modelling and simulation techniques for dynamic (or short-term) optimization. Details of the approach have been outlined in [4].

REFERENCES

[1] *Proceedings of IEEE International Symposium on Intelligent Control,* edited by A. Meystel and J.Y.S. Luh, Jan. 1987, Phil. PA.

[2] Tag Gon Kim, George Mignon, and B.P. Zeigler, "Design of an AI-Based Self-Sustaining Habitats Control System," *Proc. in the First International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, pp. 1059-1065, June 1988, Tullahoma, Tennessee.

[3] Tag Gon Kim and B.P. Zeigler, "ESP-Scheme: A realization of System Entity Structure in a LISP Environment," *Proc in 1989 SCS Eastern Multiconference,* March 1989, Tempa, Florida.

[4] Tag Gon Kim, "Design of Intelligent Control System:vol I," Tech Report, ERLab, University of Arizona, May 1988.

[5] B.P. Zeigler, *Mutifaceted Modelling and Discrete Event Simulation,* Academic Press, 1984.

[6] B.P. Zeigler, "Knowledge Representation from Minsky to Newton and Beyond," *Applied Artificial Intelligence,* vol.1, pp 87-107, Hemisphere Pub., Co. 1987.