N89-15579

A CLIPS Prototype for Autonomous Power System Control

James M. Vezina Leon Sterling
Center for Automation & Intelligent Systems Research
Computer Engineering & Science Dept
Case Western Reserve University
Cleveland, Ohio 44106

August 1988

The model of the system assumes a constant power source and loads (experiments) whose power demands exceed the supply. Experiments are described by their: name, power consumption, time for a complete run, present status and the state of the load. The power consumption of each load is set at a constant level but can be dynamically modified by the operator. The status specifies if the experiment is running, paused, completed or failed. The state compensates for the lack of actual feedback sensor data, by signifying the stability of the load. Experiments are scheduled to keep as many running as possible with the current system limitations. A graphics oriented user interface is embedded into the rule-based system to enable an operator to easily experiment with the system.

1 INTRODUCTION

With limited resources and the high cost of electrical power on the space station, it is essential to have a highly reliable and robust system to manage the scheduling and restoration of the power system. An expert system, embedded with the tried and true terrestrial algorithmic decision aids, will be able to maintain the power system with little assistance from airborne or ground support. CLIPS (C Language Integrated Production System), developed by NASA, is used to demonstrate how a rule-based system could be used to achieve this lofty goal. The prototype simulates a simple model of the power system and adaptively schedules the various loads. Once a configuration is planned, the system monitors the loads for variation in power consumption, potential failure and

completion. Upon entering one of these states, the system is reconfigured to compensate for the change.

A team maintains a city's electrical power with various feedback sensors and mathematical algorithms. Conventional methods summarize power flow data for the entire network. Conventional techniques also provide avenues to investigate the affect of possible enhancements to the system. A power system could be maintained (at least in part) by capturing the team's expertise and integrating it with the appropriate traditional techniques. This will not replace the team, but will greatly aid them in recognizing and responding to problems that arise. People can be removed from the day-to-day problems of maintenance, thus enabling them to concentrate on improving the efficiency of the system.

This project demonstrates how a rule-based expert system, embedded with user defined functions (written in C), could autonomously control the space station power system. The actual model has been greatly simplified, but still can be used to explore various control strategies. C Language Integrated Production System (CLIPS)[2][3], developed by NASA, provides a low cost, yet highly portable, expert system shell to carry out our initial endeavors. The ease of integrating algorithmic routines in CLIPS, enables the system to access traditional tools and techniques. This simple, yet robust system provides a platform to explore new concepts in autonomous space station power system control.

The current model consists of a single power source, primary buss and various loads (although only four are displayed on the screen). The power consumption of each experiment is divided into four classes: **normal, warning, critical, failure. Warning** indicates the experiment is using more power than anticipated. If the experiment's state is unstable, then this level notifies the operator that a potential problem may exist. Once an unstable experiment enters a **critical** level, it is assumed that it will fail and is therefore aborted. To prevent harmful side-effects to the rest of the system, an aborted experiment will never be rescheduled. If a stable experiment reaches the **critical** level, the operator is notified that a *good* experiment is consuming an abnormally high amount of power. The actual model of the system will be discussed in more detail, along with an explanation of the expert system.

2 PROBLEM DESCRIPTION

The model uses a constant source of power to supply the demands of the experiments. The system enables the operator to vary the available power, but it does not automatically simulate the various fluctuations that would occur in an orbiting space station. The more detailed aspects of a power distribution system, such as a spike on a line or insufficient line load[1], were beyond the scope

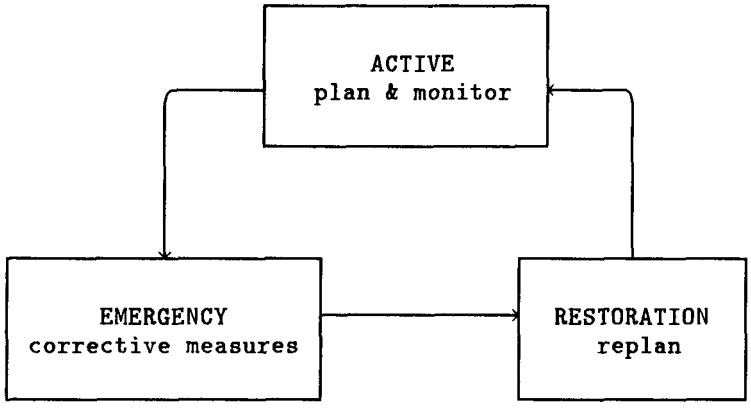

Figure 1: Power System Modes of Operation

of this work. The demonstration considers a primary power buss with auxiliary paths going to the loads. Later we will show how a user can modify the map that specifies the system configuration. Figure 1 shows the three basic modes of operation: **normal, emergency** and **restoration.** Normal mode consists of scheduling the operation of loads and monitoring the system. If a failure occurs (or is predicted to occur) the system enters emergency operation. Drastic measures must be taken to insure the integrity of the power system as a whole. An unexpected variation in the available power, or an unexpected event in the system could trigger an emergency situation. Restoration operations get the system back on track and operating normally. The system must decide which loads should be started, aborted, temporarily stopped or just left alone.

The primary goal of the expert system is to supply power to as many experiments (loads) as possible. This is not the best optimal goal for scheduling[4][5]. but was sufficient to demonstrate a type of scheduler embedded into the rule-based system. The system monitors each active load and notifies the operator if an experiment *moves* toward potential failure. A load can operate at one of five levels: **dormant, normal, warning, critical,** and **failure.** If the experiment is not running, due to a problem or scheduling constraint, it is considered dormant. The anticipated amount of power consumption is, of course, the normal level. The warning level is set to indicate an experiment is approaching failure. If the experiment consumes enough power to be *close* to failing, it becomes critical. Depending on how the system views the *correctness* of the experiment, this level would push the system into emergency mode. Failure is beyond the set limits of the breaker for the particular experiment.

In this model, the various levels are the same for all of the experiments (although this does not have to be the case). The switches (breakers) are considered

to be intelligent and able to measure a potentially failing experiment. If the experiment is operating correctly, but happens to exceed the breaker capacity, it is assumed that this condition was an accident and will not cause any problems. The experiment will continue running (the switch is closed). However, if an intelligent sensor determines that the power consumption is unstable, it will terminate the experiment before the failure level is reached, avoiding possible damage to other experiments. This method now moves the system into restoration operations. During restoration, the experiments are scheduled to take advantage of any available power. The experiments themselves are considered to be too varied and specialized to be maintained by the on-board crew, therefore diagnosing and repairing a failed experiment is considered infeasible. With this assumption, the pertinent information for each experiment is its power requirements and time-to-run. The system is also robust enough to take advantage of interruptable experiments. During restoration, the experiments are scheduled to take advantage of any available power.

The scheduler dynamically runs as many experiments as possible. If power is available, the scheduler will iteratively look for a waiting experiment that can be supported under the current constraints and start it running. This continues until there is not enough power to support any of the remaining loads. If, for some reason, the running experiments exceed the power limits of the system, unsupportable experiments are put back into the wait state or if necessary, aborted. The waiting experiments would later continue from the point they were paused. When an experiment completes successfully or is aborted, it is stopped and will not be scheduled again. At each step, the power consumption values are simulated for the entire system.

The expert system monitors the space station power system simulation. If an operating experiment nears a potentially fatal level, the operator is warned of the approaching problem. The state of the experiment can be considered good or bad. An operator will be warned if a *good* experiment enters a critical state (nearing the maximum level of the breaker). If this experiment blows the breaker, it is assumed that the load will not affect any part of the system, and the breaker is closed. When a *bad* experiment enters the warning level, the operator is notified. Upon entering a critical state, the experiment is aborted to prevent possible degradation to the system. When an experiment successfully terminates or fails, the operator is notified and appropriate action is taken. The diagnostics in this system are straight-forward, *if it fails (or is predicted to fail) then it is aborted.* This approach is disastrous in most applications, but here it is quite reasonable. Groups from all over the world will have their experiments running on the space station. The crew in the space station will not have the time or expertise to be capable of repairing failed equipment in an experiment. The availability of parts will also limit an operator's ability to repair the transient

experiments.

The user-interface (as shown in figure 2) displays the current state of the system. It was intented to display as much information as possible in an easy to understand format. In an actual application, it would not be practical to continuously display all of the data, but this project was designed as an learning tool or experimental platform. At the start of each time interval, the operator can elect to modify any of the parameters of the experiments. In this way, the user can induce potential problems into the power system and experiment with the corrective actions taken by the expert system.

As can be seen in figure 2, there is a **dynamic help bar** containing the commands or responses that are possible at a given time (the commands themselves prompt the user for all input). When modifying (editing) an experiment, the dynamic help bar displays all of the possible commands and alters the prompt accordingly. If the operator elects to increase the power consumption of an experiment, the system prompts for the new value, while the help bar indicates the appropriate response (an integer value) and range. The operator can also access a brief on-line tutorial describing the entire system.

The user-interface is primarily written in C, using the curses library. These commands were embedded into the CLIPS shell and are called via rules. The changes made by the user are asserted directly into the knowledge base of the shell. Rules exist to take these changes and appropriately *edit* the simulation database.

3 DETAILS ON THE EXPERT SYSTEM

3.1 Facts - Data Structures

The data structures are broken down into four groups: switches, experiments, power supply and general facts. The facts are in the form of:
(type-of-node node-name slot-name value).
The type-of-node indicates what type of node the fact is concerned with (e.g. experiment), and the node-name specifies the particular node (e.g. name of the experiment). The slot-name determines what the fact is about and its value. An example of a slot-name would be *status* with the value of *wait* or
(experiment exper_1b status wait).

The switches have a fact for their electrical current and another for their state (i.e. open, closed or blown). Each experiment has 5 slots of information (facts): current load, time, time-to-end, status and state. Figure 3 contains an example of experiment 2 and its corresponding switch. The time slot contains

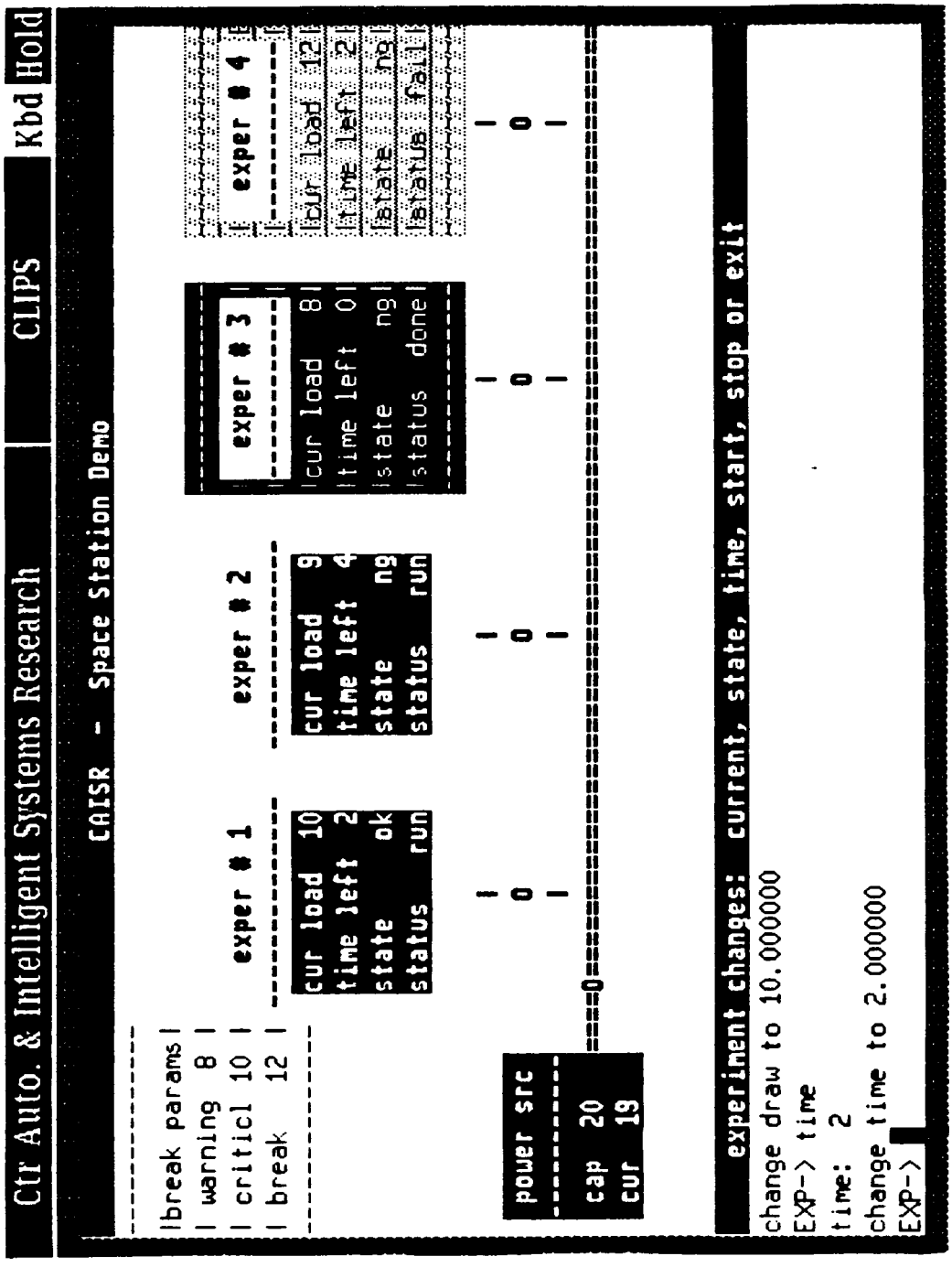

Figure 2: An example session

```
(switch  s1  state     open)
(switch  s1  current   0)

(experiment  exper_2  draws  10)
(experiment  exper_2  time   5)
(experiment  exper_2  time_to_end  unknown)
(experiment  exper_2  state  ok)
(experiment  exper_2  status wait)

(connect s1  exper_2)
```

Figure 3: An example of the facts describing an experiment and its switch

the total amount of time till the experiment is finished. In the figure, the time-to-end has a value of **unknown** to indicate that is has no scheduled end time (because it is not running). Status of the experiment can be: wait, run, done or failed. The state of the experiment is used to simplify the simulation of the power system. As mentioned earlier, a switch can measure the reliability of the experiment. Instead of simulating this operation, each experiment is listed as: OK or NG (No Good for not operating as expected). This is all the information necessary schedule and maintain the experiments.

The configuration of the system is also given by the facts. There is a single fact for each connection in the power system. Figure 3 shows the connection between the switch and the experiment. By altering these simple facts, the power system network can easily be redesigned.

3.2 Rules

The rules of the system control the simulation, the user interface and any algorithmic functions embedded in the system. The knowledge base simulates the power system model to enable the user to easily reconfigure the network by altering a few facts. A traditional algorithmic technique could prove to be more accurate and much faster, but would lose the flexibility and ease of the current method. The system traverses the network calculating the power level at each node. The user interface was embedded directly into CLIPS and can be viewed as a group of new shell functions. CLIPS greatly simplifies the integration of the traditional methods with the expert system. Data can easily be passed to and from the user defined function by simple *get* and *assert* type commands.

The rule format is shown in the following figure (figure 4). If the current

passing through a switch is above the indicated failure level, then the switch must not be operating correctly, otherwise it would have blown. The semi-colon (";") delineates comments on the rule. The rule name is **Switch_bad** and is used if the breaker appears to be fused closed. The first six lines after the rule name are the antecedents of the rule. These must all be satisfied in order for the rule to be fired. The question mark ("?") indicates that the following is a variable and should be set to the appropriate value of a matched fact (CLIPS uses the data driven Rete matching algorithm). The ampersand-colon indicates that the match must also satisfy the additional constraints that appear in the parentheses. The three lines after the arrow ("*arrow*") are the consequences, executed when the rule is fired. In this example, the consequences assert three new facts into the knowledge base. The first and second will be sent to the operator (via the error handler and then the user-interface). The last line will fire another rule to halt (abort) the experiment. Although this example does not have a specified priority, any rule can be given a priority by adding: **(declare (salience 100))**. This line declares that the priority of the rule is 100 and will internally adjust the order for inferencing the rules.

The rules for monitoring and restoration can be broken down into the seven basic rules listed in table 1. The table has the rules indexed in each row of the table. The first three columns show the more pertinent information for deciding if an experiment "looks" as if it will present any danger to the system. SWITCH CURRENT gives the symbolic value for the level of power the experiment is consuming. SWITCH STATE represents the expected state of the network and the experiment (should the experiment be connected to the network, i.e. switch *closed*). The EXPERIMENT STATE, as mentioned earlier, indicates the feedback information on the power flow to the experiment. This is either good or bad (OK or NG). The last two columns are the consequences to the scenarios. The REACTION represents the expert system response to the operator or the power system. The user could be notified or appropriate action might immediately be taken on the network itself (e.g. disconnect the experiment from the rest of the system). The NEW EXP. STATE indicates the new status of the experiment, **fail** (if the experiment is aborted) or **same** (if it is not altered). Emergency handling rules and a variety of system maintenance rules complete the rule-based controller, and the discussion on this project.

4 ACKNOWLEDGEMENTS

This work was supported in part by a grant from the NASA Lewis Research Center (grant number: NAG 3-787). I appreciate all of the help I received from NASA Lewis, especially Les Burke and Jim Kish, and the continued help and

```
;
;*********************************************
; RULE:  6
;  Switch:     FAILURE
;              closed
; Experiment: NG
; DO:         NOTIFY / OPEN_SWITCH
;
(defrule Switch_bad
   (switch ?switch_name current ?i_value)
   (switch current failure
        ?failure_value&:(>= ?i_value
                            ?failure_value))
   (switch ?switch_name ?experiment_name)
   (experiment ?experiment_name status run)
   (experiment ?experiment_name state ng)
=>
   (assert (info experiment ?experiment_name
                            halted))
   (assert (info switch ?switch_name fused))
   (assert (stop_experiment ?experiment_name
                            fail)))
```

Figure 4: An example of a rule in CLIPS

Diagnostic & Restoration Rule Table					
Rule number	Switch current	Switch state	Experiment state	Reaction	New Exp. state
1	ok	closed	ok	——	same
2	warn	closed	ng	notify	same
3	critical	closed	ok	notify	same
4	critical	closed	ng	notify open-switch	fail
5	failure	closed	ok	notify	same
6	failure	closed	ng	notify open-switch	fail
7	——	blown or open	(running)	notify	fail

support from Eric Bobinsky. I would like to thank the Cleveland Advanced Manufacturing Program for their aid in supporting this project. Finally, I would like to acknowledge the computer resources provided by the Center for Automation and Intelligent Systems Research at Case Western Reserve University.

5 REFERENCES

[1] DyLiacco, TE *The Adaptive Reliability Control System* IEEE PAS-104(12) pp 3423-2427, Dec. 1985

[2] Giarratano, Joseph C. **Clips User's Guide** Sept 13, 1987

[3] Giarratano, Joseph C. **Clips Reference Manual** Sept 13, 1987

[4] Touchton, Robert A *Common Module Dynamic Payload Scheduler Expert System* **Proceedings of the Twenty-first Intersociety Energy Conversion Engineering Conference**, August 1986, vol III pp 1785-1790

[5] Washington, Sylvia **Optimal Load Management for the Space Station Power System** Masters Thesis, Case Western Reserve University, 1987