Tyler Hardy

twhardy

ECE 6420 Section 400

Takehome #1

Description:

In a critical manufacturing environment, production downtime can result in significant financial losses. It is often necessary to maintain backup equipment ready to take the place of manufacturing assets when machines must be taken offline for preventative or reactive maintenance. Using predictive maintenance techniques, it is possible to predict overhaul availability dates with a high degree of accuracy. When assets number in the hundreds or thousands with various configurations that require lengthy overhaul periods, ensuring adequate production units are available becomes a labor-intensive activity.

This project seeks to develop an expert system that can read an Excel database and determine when expected overhauls may run the risk of having insufficient production assets available to meet a specified baseline work flow.

References:

-CLIPS Reference Manual, Volume 1: Basic Programming Guide
-CLIPS Reference Manual, Volume 2: Advanced Programming Guide
-Head First C by O'Reilly
-ECE6420 Lecture Notes

Strategy:

This project involves the implementation of embedded CLIPS to read data stored in a Comma Separated Values format. This data will be stored as a variable set and exported to CLIPS.

Upon receipt of the variables to analyze, rules will be applied to determine where equipment overhaul periods overlap within a given category of machine.

After determining when machine downtimes will overlap, the next task is to determine when overlapping overhauls will result in insufficient production assets available for the specified baseline workload.

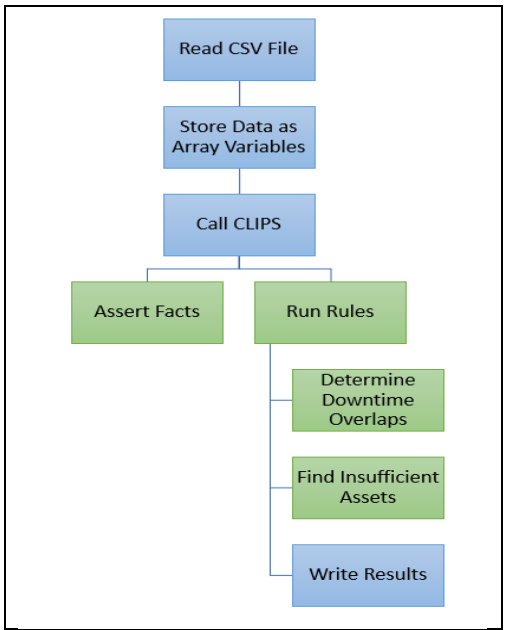

Figure 1 – Proposed Flow Diagram

Outcome/Deliverables: The goal of the project is to quickly provide a list of at-risk asset classes based on projected overhaul dates with minimal labor involvement on the part of the user.

Development:

This project was created to facilitate machine downtime scheduling. The initial concept of the project was to use a C program to read a CSV file and compute facts based on the contents of the file. Error correction and formatting was expected to be necessary based on the expected contents of the file. The C program would then transmit data in the form of an array to CLIPS for processing in a series of rules which would determine risk based on overlapping date schedules.

Due to difficulty in implementing rules which could accurately compare a given fact to all other facts, this concept was abandoned in favor of a categorization system. The categorization system dramatically decreased the complexity of the rules required in order to ascertain machine risk. The previous code for risk flagging was left in place for future updates as necessary.

The code works by first importing necessary header files and declaring variables and functions as required. It then imports data from database.csv and stores it as a structured array. Legacy code is left in place to ensure accurate reading of the CSV file. The CLIPS environment is instantiated and data from the structured array is transferred iteratively through a specific function call developed for that purpose.

The CLIPS code works simply by categorizing each fact and incrementing a global variable based on fact assignment. The negative salience rule which fires last checks global variables for anything exceeding the hardcoded threshold for when excessive machines are down for overhaul. For the purposes of this exercise, the threshold of 4 was chosen for all machine classifications. The periods of risk for each machine classification are then output to the user from CLIPS and the program ends.

Results:

Overall, the project can be said to be successful. Performing hand analysis of the “dummy” CSV file produces results which match the program output. The program can be immediately used with a valid input data file.

Outcomes:

This project met all goals in terms of outcomes and deliverables. All features called for in the proposal were successfully implemented.

Code for C Executable shown in Figure 2.

```
#include<stdio.h>
#include<stdlib.h>
#include "clips.h"

void AddMachineFact(void *, int, int, int, int);

int main(int argc, char *argv[])
{
	struct data_collect
	{
		int machineType;
		int machineSpeed;
		int machineSerial;
		int machineStart;
//		int machineFinish;
	};
	char buffer[21];
	FILE *filetarget;
	char *field;
	int row;
	char *filename = "overhaul.clips";
	int bufferSerial;
	int bufferType;
	int bufferSpeed;
	int bufferStart;
//	int bufferFinish;

	void *theEnv;

	char *cs;

	printf("Opening CSV file...\n");
	filetarget = fopen("database.csv","r");
	printf("Initializing records...\n");
	struct data_collect records[100];

	row = 0;

	while(fgets(buffer,21,filetarget)){

		field = strtok(buffer,",");
		records[row].machineType = atoi(field);
		field = strtok(NULL,",");
		records[row].machineSpeed = atoi(field);
		field = strtok(NULL,",");
		records[row].machineSerial = atoi(field);
		field = strtok(NULL,",");
		records[row].machineStart = atoi(field);
//		field = strtok(NULL,",");
```

```
//                  records[row].machineFinish = atoi(field);
                    ++row;
            }

            fclose(filetarget);
            printf("Storage complete.\n");
            // Duplicated in CLIPS
            cs = "(deftemplate equipment"
            " (slot serial)"
            " (slot type"
            " (default 0)"
            " (type INTEGER))"
            " (slot speed"
            " (default 0)"
            " (type INTEGER))"
            " (slot start"
            " (type INTEGER))"
            " (slot risk"
            " (default no)"
            " (allowed-symbols yes no)))";

            /* CLIPS Implementation */

            theEnv = CreateEnvironment();

            EnvLoad (theEnv, filename);

            printf("Preparing to build environment\n");
            EnvBuild(theEnv,cs);
            printf("Environment built.\n");

            EnvReset (theEnv);


            int outputcount;
            for ( outputcount = 1; outputcount < 100; outputcount++)
            {
                    //printf("Machine Serial: %i\n", records[outputcount].machineSerial);

                    bufferSerial = records[outputcount].machineSerial;
                    //printf("Machine Type: %i\n", records[outputcount].machineType);
                    bufferType = records[outputcount].machineType;
                    //printf("Machine Speed: %i\n", records[outputcount].machineSpeed);
                    bufferSpeed = records[outputcount].machineSpeed;
                    //printf("Machine Start: %i\n", records[outputcount].machineStart);
                    bufferStart = records[outputcount].machineStart;
                    //printf("Machine Finish: %i\n\n", records[outputcount].machineFinish);
//                  bufferFinish = records[outputcount].machineFinish;
                    AddMachineFact(theEnv, bufferSerial, bufferType, bufferSpeed, bufferStart);
            }
```

```
	printf("Overhaul expert system loaded \n");
	printf("This will calculate machines at risk of running out of spares assuming 4 spares
of each type and speed. \n");
	printf("Outputting results...\n");
	//EnvAgenda (theEnv, "wdisplay", NULL);
	EnvRun (theEnv, -1);



	//EnvFacts (theEnv, "wdisplay",NULL,-1,-1,-1);

	printf("Cleaning up...\n");
	DestroyEnvironment (theEnv);

	printf("Press enter to exit...\n");
	getchar();

	return (1);
}
void AddMachineFact(void *environment, int factSerial, int factType, int factSpeed, int
factStart)
{
	void *newFact;
	void *templatePtr;
	void *theMultifield;
	DATA_OBJECT theValue;

 //IncrementGCLocks(environment);

 /*==================*/
 /* Create the fact. */
 /*==================*/

	templatePtr = EnvFindDeftemplate(environment,"equipment");
	newFact = EnvCreateFact(environment,templatePtr);
	if (newFact == NULL) return;

 /*=============================*/
 /* Set the value of the serial slot. */
 /*=============================*/

	theValue.type = INTEGER;
	theValue.value = EnvAddLong(environment,factSerial);
	EnvPutFactSlot(environment,newFact,"serial",&theValue);

 /*=============================*/
 /* Set the value of the type slot. */
 /*=============================*/

	theValue.type = INTEGER;
	theValue.value = EnvAddLong(environment,factType);
```

```
        EnvPutFactSlot(environment,newFact,"type",&theValue);

 /*============================*/
 /* Set the value of the speed slot. */
 /*============================*/

        theValue.type = INTEGER;
        theValue.value = EnvAddLong(environment,factSpeed);
        EnvPutFactSlot(environment,newFact,"speed",&theValue);

 /*============================*/
 /* Set the value of the start slot. */
 /*============================*/

        theValue.type = INTEGER;
        theValue.value = EnvAddLong(environment,factStart);
        EnvPutFactSlot(environment,newFact,"start",&theValue);

 /*============================*/
 /* Set the value of the finish slot. */
 /*============================*/

//      theValue.type = INTEGER;
//      theValue.value = EnvAddLong(environment,factFinish);
//      EnvPutFactSlot(environment,newFact,"finish",&theValue);

 /*============================*/
 /* Set the value of the risk slot. */
 /* Currently this is not used.    */
 /*============================*/

        theValue.type = SYMBOL;
        theValue.value = EnvAddSymbol(environment,"no");
        EnvPutFactSlot(environment,newFact,"risk",&theValue);


 /*================================*/
 /* Assign default values since all */
 /* slots were not initialized. */
 /*================================*/

        EnvAssignFactSlotDefaults(environment,newFact);
 /*=========================================================*/
 /* Enable garbage collection. Each call to IncrementGCLocks */
 /* should have a corresponding call to DecrementGCLocks. */
 /*=========================================================*/
 //EnvDecrementGCLocks(environment);
 /*=================*/
 /* Assert the fact. */
 /*=================*/
```

```
        EnvAssert(environment,newFact);
}
```

Figure 2 – C Code

CLIPS LISP Code shown in Figure 3.

```
;; Implementation of a Rules Database
;; File: overhaul.clips

;; Global Variables

(defglobal ?*categoryA* = 0)
(defglobal ?*categoryB* = 0)
(defglobal ?*categoryC* = 0)
(defglobal ?*categoryD* = 0)
(defglobal ?*categoryE* = 0)
(defglobal ?*categoryF* = 0)
(defglobal ?*categoryG* = 0)
(defglobal ?*categoryH* = 0)
(defglobal ?*categoryI* = 0)
(defglobal ?*categoryJ* = 0)
(defglobal ?*categoryK* = 0)
(defglobal ?*categoryL* = 0)
(defglobal ?*categoryM* = 0)
(defglobal ?*categoryN* = 0)
(defglobal ?*categoryO* = 0)
(defglobal ?*categoryP* = 0)


;; Template for Equipment

(deftemplate equipment
        (slot serial)
        (slot type
                (default 0)
                (type INTEGER))
        (slot speed
                (default 0)
                (type INTEGER))
        (slot start
                (type INTEGER))
        (slot risk
                (default no)
                (allowed-symbols yes no)))

;; RULES

(defrule categoryCheckA
        (logical (equipment (type 1)(speed 1)(start 42736)))
        =>
```

```
        (bind ?*categoryA* (+ ?*categoryA* 1)))

(defrule categoryCheckB
        (logical (equipment (type 1)(speed 1)(start 42767)))
        =>
        (bind ?*categoryB* (+ ?*categoryB* 1)))

(defrule categoryCheckC
        (logical (equipment (type 1)(speed 1)(start 42795)))
        =>
        (bind ?*categoryC* (+ ?*categoryC* 1)))

(defrule categoryCheckD
        (logical (equipment (type 1)(speed 1)(start 42826)))
        =>
        (bind ?*categoryD* (+ ?*categoryD* 1)))

(defrule categoryCheckE
        (logical (equipment (type 1)(speed 2)(start 42736)))
        =>
        (bind ?*categoryE* (+ ?*categoryE* 1)))

(defrule categoryCheckF
        (logical (equipment (type 1)(speed 2)(start 42767)))
        =>
        (bind ?*categoryF* (+ ?*categoryF* 1)))

(defrule categoryCheckG
        (logical (equipment (type 1)(speed 2)(start 42795)))
        =>
        (bind ?*categoryG* (+ ?*categoryG* 1)))

(defrule categoryCheckH
        (logical (equipment (type 1)(speed 2)(start 42826)))
        =>
        (bind ?*categoryH* (+ ?*categoryH* 1)))

(defrule categoryCheckI
        (logical (equipment (type 2)(speed 1)(start 42736)))
        =>
        (bind ?*categoryI* (+ ?*categoryI* 1)))

(defrule categoryCheckJ
        (logical (equipment (type 2)(speed 1)(start 42767)))
        =>
        (bind ?*categoryJ* (+ ?*categoryJ* 1)))

(defrule categoryCheckK
        (logical (equipment (type 2)(speed 1)(start 42795)))
        =>
        (bind ?*categoryK* (+ ?*categoryK* 1)))
```

```
(defrule categoryCheckL
	(logical (equipment (type 2)(speed 1)(start 42826)))
	=>
	(bind ?*categoryL* (+ ?*categoryL* 1)))

(defrule categoryCheckM
	(logical (equipment (type 2)(speed 2)(start 42736)))
	=>
	(bind ?*categoryM* (+ ?*categoryM* 1)))

(defrule categoryCheckN
	(logical (equipment (type 2)(speed 2)(start 42767)))
	=>
	(bind ?*categoryN* (+ ?*categoryN* 1)))

(defrule categoryCheckO
	(logical (equipment (type 2)(speed 2)(start 42795)))
	=>
	(bind ?*categoryO* (+ ?*categoryO* 1)))

(defrule categoryCheckP
	(logical (equipment (type 2)(speed 2)(start 42826)))
	=>
	(bind ?*categoryP* (+ ?*categoryP* 1)))

(defrule risk-check
	(declare (salience -10))
=>
	(if (> ?*categoryA* 3)
		then
		(printout t "Type 1 Speed 1 Machines are at risk in January" crlf))

	(if (> ?*categoryB* 3)
		then
		(printout t "Type 1 Speed 1 Machines are at risk in February" crlf))

	(if (> ?*categoryC* 3)
		then
		(printout t "Type 1 Speed 1 Machines are at risk in March" crlf))

	(if (> ?*categoryD* 3)
		then
		(printout t "Type 1 Speed 1 Machines are at risk in April" crlf))

	(if (> ?*categoryE* 3)
		then
		(printout t "Type 1 Speed 2 Machines are at risk in January" crlf))

	(if (> ?*categoryF* 3)
		then
```

```
            (printout t "Type 1 Speed 2 Machines are at risk in February" crlf))

        (if (> ?*categoryG* 3)
            then
            (printout t "Type 1 Speed 2 Machines are at risk in March" crlf))

        (if (> ?*categoryH* 3)
            then
            (printout t "Type 1 Speed 2 Machines are at risk in April" crlf))

        (if (> ?*categoryI* 3)
            then
            (printout t "Type 2 Speed 1 Machines are at risk in January" crlf))

        (if (> ?*categoryJ* 3)
            then
            (printout t "Type 2 Speed 1 Machines are at risk in February" crlf))

        (if (> ?*categoryK* 3)
            then
            (printout t "Type 2 Speed 1 Machines are at risk in March" crlf))

        (if (> ?*categoryL* 3)
            then
            (printout t "Type 2 Speed 1 Machines are at risk in April" crlf))

        (if (> ?*categoryM* 3)
            then
            (printout t "Type 2 Speed 2 Machines are at risk in January" crlf))

        (if (> ?*categoryN* 3)
            then
            (printout t "Type 2 Speed 2 Machines are at risk in February" crlf))

        (if (> ?*categoryO* 3)
            then
            (printout t "Type 2 Speed 2 Machines are at risk in March" crlf))

        (if (> ?*categoryP* 3)
            then
            (printout t "Type 2 Speed 2 is at risk in April" crlf))
)
```

Figure 3 – CLIPS Code

The CSV file format follows the general theme set forth in Figure 4.

Type	Speed	Serial	Overhaul Scheduled
1	2	1	1/1/2017
2	1	2	2/1/2017
1	2	3	3/1/2017

Figure 4 – Example CSV Format

As seen in Figure 5, the code compiles with no errors and only throws one warning upon execution.

```
c:\clips_core_source_630\core>gcc overhaul.c -L./ -lclips -lm -o overhaul

c:\clips_core_source_630\core>overhaul
Opening CSV file...
Initializing records...
Storage complete.
Preparing to build environment

[CSTRCPSR4] Cannot redefine deftemplate equipment while it is in use.

ERROR:
(deftemplate MAIN::equipment
Environment built.
Overhaul expert system loaded
This will calculate machines at risk of running out of spares assuming 4 spares of each type and speed.
Outputting results...
Type 1 Speed 1 Machines are at risk in January
Type 1 Speed 1 Machines are at risk in April
Type 1 Speed 2 Machines are at risk in January
Type 2 Speed 1 Machines are at risk in January
Type 2 Speed 1 Machines are at risk in April
Type 2 Speed 2 Machines are at risk in January
Cleaning up...
Press enter to exit...
```

Figure 5 – Runtime Results

Overall, the project could benefit from several improvements. Data checking and input sanitization is always of paramount concern when dealing with imported file data. Reliance on hardcoded date selection reduced the code flexibility by a significant margin. Additionally, the way the code is written currently does not easily lend itself to expansion without a subsequent duplication of significant amounts of code.

One potential area of improvement is in the recognition of dates. Currently, the code requires CSV files to use the integer equivalent of human-readable, delimited data. Instead, it should be able to parse out days, months, and years then import that data into the CLIPS engine.

Another area of improvement is in actual overlap validation. As the expected data from an actual manufacturing production system will be coming in with a period of downtime rather than a single downtime date, it will be necessary to revise the rules somewhat to account for this.

Currently, a warning is thrown based on the syntax requirements of both CLIPS and the C code. Both require definitions of the equipment template; however, the equipment template can only be defined once. As this is a warning that does not impact code performance, it can be moved to a future bugfix.