Stephen Duke

OCAML - SCHEDULER

For my project, I sought out to develop a scheduler for the current company I work for. Currently, scheduling takes hours per week and is unnecessarily costly in both time and effort. This has been something on my mind for the last six months and decided it would be a perfect opportunity to get started. I initially thought that it would be interesting to use the concepts of constraint satisfaction within the scheduling algorithm, as it seemed to make sense within the context. After developing the program, it became increasingly clear that the reason scheduling was so costly was the perspective of management. There is no reason why their scheduling process should take so long, it is just a perspective fallacy. Other than the concept of priority of employees, the algorithm is almost identical to any other scheduling algorithm.

Currently, tours are scheduled by neighborhood and many employees may be capable of running tours in different neighborhoods. At first, this seemed like the issue that I would be solving. However, after laying out the pseudocode and running through the thought process I realized that this is simply the same concept of an employee both being a server and a bartender at a restaurant. As such, it was easily solved.

I initially sought to create or use a library for constraint satisfaction. This concept is the process of finding a solution to a set of constraints that impose conditions that the variables must satisfy. [1] Coming from this perspective, solving the problem set was very simple. I created a set of Boolean functions for testing constraints. The Booleans in place exist within a main function titled “no_conflict”. This function checks every possible constraint and if

constraints are satisfied it returns True, otherwise, if any constraint is not satisfied it returns false. A better way to make this code more

```
let no_conflict emp schedule =
  if ((exists_in schedule.neighborhood emp.duties) != true) then false else
  if (employee_has_reached_max_shifts emp) then false else
  if (exists_in schedule.date emp.days_off) then
    (if ((exists_in schedule.time emp.days_off_times) ||
         (List.hd emp.days_off_times = "ALL")) then false else true)
  else if (exists_in schedule.day_of_week emp.days_off) then false else true
```

robust would have been to directly program logic and use symbols to represent each variable. In this case, I could have created a module to then be used in further applications. As it stands, this code is not modular in that regard.

Because I felt as though I had not done enough work, I sought out to create a graphical interface to implement the program as the end user would never use the terminal. I initially wanted to use the ocaml library lablgtk, because it allows for much more interesting graphics, but after repeated attempts in mac OS Mojave, I was unable to properly install the library. It installed for linux, but if I relied on my PC, I would have been unable to finish in time as I'm frequently not home. I eventually got a different library lablтk downloaded and created an interface for the program. It simply allows the user to select file input and passes the file into the main scheduling algorithm.

The main scheduling algorithm works by reading csv files and parsing them into records. Once the files are parsed into records I am able to easily pass and manipulate the date through various functions. With a list of employees with their constraints and a given schedule, I am able to match employees to a given schedule. If no employee exists to satisfy the schedule, a new empty employee is created with the name "NONE_AVAILABLE".

While I wish that I had chosen a more interesting project, my company is interested in buying the software. Therefore, I did satisfy part of my initial goal. I learned so much from the

process and the project solidified more knowledge of the benefits of functional programming. I appreciate how secure the code feels. If I were to do the project again, I would definitely attempt to integrate ocaml with some sort of embedded systems. I'm very interested in electronics and very much wish that I had instead created a system to interface with the serial port and built a physical project.

CITATIONS

1. Edward Tsang (13 May 2014). Foundations of Constraint Satisfaction: The Classic Text. BoD – Books on Demand. ISBN 978-3-7357-2366-6.