Brian Jahnke

Final Write Up

Imleagues DFA

Abstract

This document shall outline my implementation of a DFA within my chosen subject, intramural sports at Marist. It shall first give the motivations behind the project. Then, it will discuss the user interface and the various commands that are available to different users. Along with an illustration of the actual DFA there will be a description of how it specifically works from state to state and how the implementation of the DFA improves upon the problems inherent in the Imleagues site.

Introduction

The purpose of intramurals at college should be to get students involved in a fun activity during their spare time. As someone who has acted as a commissioner for intramurals at Marist for over a year, one of the main issues I have come across is that there are many people who attempt to get involved in intramural sports but do not end up playing due to shortcomings in Marist's intramural website, Imleagues.com. Every semester, there are plenty of people who create teams but are just shy of the minimum roster requirement and are not able to play because of it. There are also many people who sign up for a league by themselves as "free agents". This feature is suppose to allow people to get involved even if they don't personally know anyone that's making a team. However, since the process of including a free agent into an existing team is

entirely manual, in many cases free agents are never put on a team. Other difficulties come from the fact that many of the rules that define what a team ought and ought not have must be regulated personally by intramural commissioners. It is for these reasons that I have sought to apply a DFA to this subject.

Detailed System Description

The program I designed is made for 2 types of users in mind, players and admins. Before logging in to player or admin, the user only has access to a few commands:

User Commands:

Command	**Description**
help	Prints all commands available to the User
login	Prompts the user to enter their marist email to login to that account. The user gains access to the player commands upon login.
Register	Prompts the user to enter first name, last name, and grade/class. Stores this info in a player object. The user can now sign in with x.y1@marist.edu where x = first name entered and y = last name entered.
admin	Gives access to admin commands

Once a player logs into the system via their marist email, they will have access to various commands:

Player Commands:

Command	**Description**

my info	Prints info about the current user. Including first name, last name, class, current team, and email.
create team	Creates a Team object with a name specified by the user. The user who creates the Team is stored as the team “Captain” within the team object. The Team object is also stored in the current Player object as their current team.
Create legal team	Executes the DFA which will attempt to put the user’s team into accepting state.
Add free agent	If the current user is a Captain, this command allows the user to pick a player from the free agent list and add them to the user’s team.
Enter wait list	If the current user is the Captain of a team, this command allows the user to insert their team into the wait list for the league.
Become free agent	If the current user has no team, this command allow the user to insert themselves into the free agent list.
Leave team	If the current user has a team, this command allows the user to leave that team.
Check team	Prints out the names of all members in the user’s team as well as the index that they appear in the team roster list.
Check free agents	Prints out the names of all free agents as well as the index in which they appear in the free agent list.
Help	Prints all of the commands available to the player

The Admin commands are accessed by typing ‘admin’ in the REPL, and give access to commands completely separate from the user:

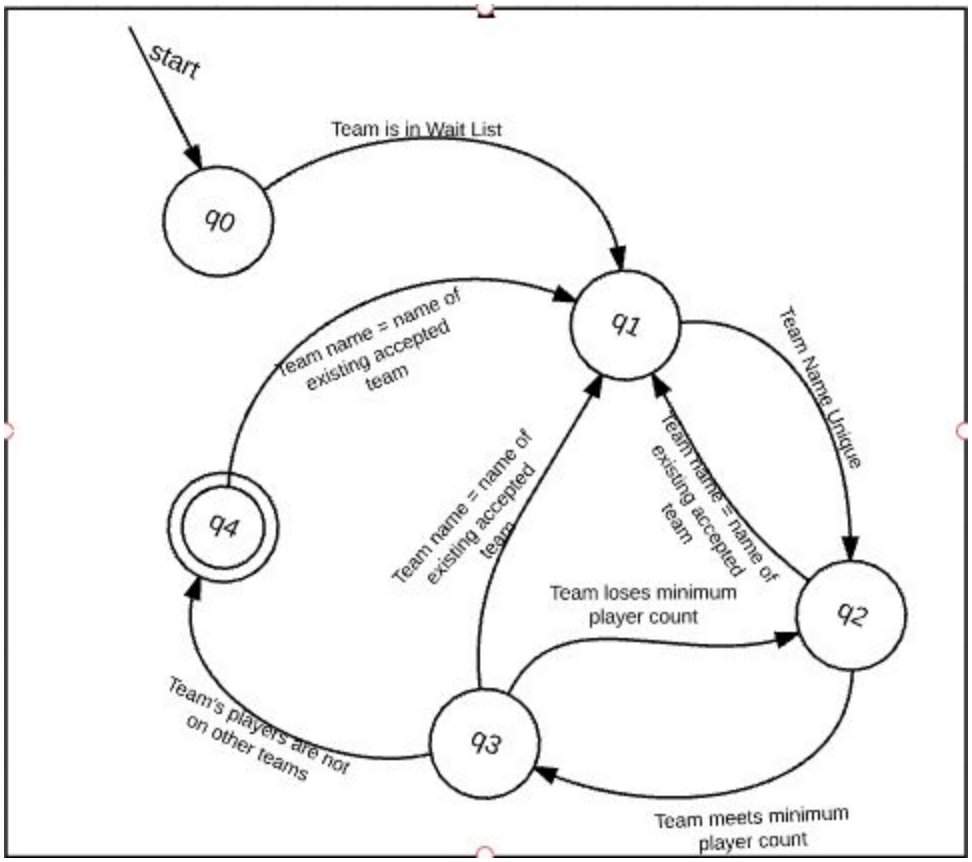
Admin Commands:

Command	**Description**
Free agents	Prints out a list of free agents and their corresponding index in the free agents list
League info	Prints out the league name as well as the league’s max capacity, its current capacity, and a list of all the teams in the

	league.
Wait list	Prints out all the teams in the wait list
View roster	User inputs a team name, the roster of that team is printed.
Fix league	The DFA is applied to every team in the wait list.
help	Prints all the commands available to the admin

At its core, the new system is similar to what Imleagues has in place with a few key differences. Users can still create their own teams and add players entirely of their own choosing. The system also allows for those without a team to join the league as a free agent. However, unlike Imleagues, the leader or “captain” of a team may submit their team to the DFA which will use the resources available to put the team into an accepting state as defined by league parameters. In terms of how this affects free agents, if a team that is submitted to the DFA has below the required roster size as defined by the league, the DFA will add players from the free agent list to the team until that requirement is met. This means that teams that would have normally dropped out due to their roster being too small can still exist and players who go in as free agents are more likely to get picked up by a team. The Admin’s ‘fix league’ command applies the DFA to every team in the wait list. Thus, only teams that are accepted by the DFA as defined by the league are then admitted into the league list.

<u>The DFA:</u>



This DFA outlines the requirements a team must meet in order to be accepted into the league. First, the team must be located within the waitlist. Next, the DFA will check if the team's name is unique. A team's name is considered unique if there is no team in the accepted list (the regular league list) that has a name equal to the team's. This is a requirement because Imleagues currently has no parameters set up to prevent teams from having the same name. This can make it confusing when trying to discern one team from the other on paper. Next, the DFA requires that the team meet the minimum roster size as defined by the league. This is important because there's a finite amount space available in a league; if a team can join the league even though it does not have enough players to play, this prevents other teams, who may have enough players, from joining the league. Finally, the DFA will check if every player on the team's roster is not

already located on any accepted team's roster. Again, Imleagues does not keep track of how many teams a specific player is apart of in one league. First, this prevents other players from joining the league, as there is one person taking up potentially multiple spots. Second, it causes a troublesome situation when players who are on multiple teams have to play against their own teams. When the DFA reaches an accepted state with the team, the team will then be added to the league if there is space available. Otherwise the team will stay on the waitlist.

Requirements

Any modern computer with java installed on it.

Literature Survey

Imleagues not only services Marist but nearly a thousand other colleges across the United States. This system's design bases its functionality off of imLeagues. The solutions implemented in this system came from working with and analysing the shortcomings of the site. As previously stated, my solution to the site comes from the need for more automatic structures that reduces that amount of manual league maintenance required by commissioners and allows more students to be involved in intramurals.

User Manual

Upon first running the code, the user will have an access to a limited amount of commands. From here, one can either sign in as a player or an admin. No matter what commands are being accessed, each command list provides a 'help' command which reveals every available command for that command list. Although many player objects

are generated at the execution of the code, they are somewhat randomly generated so the user would have no way of knowing the login email of any of the existing player objects. Thus, it is recommended that the user use the ‘register’ command to register a player object defined by them. Once the player is registered, the user can gain access to the Player commands by using the ‘login’ command followed by fname.lname1@marist.edu where fname and lname are whatever the user defined as the first and last names. From here, one can create a team, add players, test the DFA, etc. The user can ‘logout’ of the their player account and type ‘admin’ to gain access to the admin commands. From here, you can run ‘fix league’ and execute the various report commands to see the results.

Conclusion

While my example implementation of the imleagues site supports the same basic functionality as Imleagues, the key differences in how my implementation accepts new teams into the league give it the potential to have a much better user experience than the normal site offers. Introducing a DFA into the system allows for more students to have a chance to participate in Intramurals and greatly lessens the tediousness of maintaining leagues. It is clear that adopting a DFA in a system like Imleagues would provide a smoother, more responsive experience.