Oregon State University
College of Engineering

CS Capstone Progress Report

May 16, 2020

Fall 2019

Code in Sheep's Clothing

Prepared for

Oregon State University

Martin Erwig

Prepared by

Group 41b

Tom and Friends

Johannes Freischuetz
Fern Bostelman-Rinaldi
Ben Warschauer
Thomas Croll
Jackson Bizjak

Abstract

The goal of the project is to make a DSL that is able to introduce middle scholars to programming. This term on the project our group has been able to work on the definitions of the project and some basic implementations for the project. The main part of this project was defining the syntax for the language, which we also have a basic implementation for. While there were some challenges in finding an good definition for the language, we were able to make a compromise.

Contents

1 Purposes and Goals

Our system is addressing the need to introduce and teach middle school students core programming concepts who will not have any prior experience with programming and may be lacking technical knowledge needed for lower level languages. The goal of this language is to give these students a gradual introduction into core concepts without focusing on heavily technical syntax. This language These will be sixth and seventh grade students who will be learning this language at a summer camp and at their middle school.

2 Current Progress

Most of our progress has been towards defining the scope of our language, defining the concrete syntax, and building prototype parse rs and interpreters. A lot of our work doubled back on itself, our concrete syntax went through multiple implementations and the scope of the language went from broad to narrow to broad again, but we have functional versions of those three things.

3 Roadblocks and Solutions

The biggest challenge this term was figuring out the concrete syntax for our language. Finding a middle ground where the language introduces programming concepts while not making it too complicated for the user. We went through many iterations including having the language have turns, board, and pieces implicit in the language. We settled on a syntax shown below, where the user can still make something like Tic Tac Toe much more easily than in a language like C, while still needing to understand if statement, loops, and functions. The syntax we have now is not completely final, as there is a continuous question of what works best for teaching students, and what is needed for the best implementation.

4 Work Samples

```
-- Board and input type definitions
--
type Board = Grid(3,3) of Player|Empty
type Input = Position


-- Game setup
--
initialBoard : Board
initialBoard(positions) = Empty

goFirst : Player
goFirst = A
```

```
-- Game ending: game-over condition and game outcome
--
gameOver : Board -> Bool
gameOver(b) = or(threeInARow(b),isFull(b))

outcome : (Board,Player) -> Player|Tie
outcome(b,p) = if inARow(3,A,b) then A else
               if inARow(3,B,b) then B else
               if isFull(b)     then Tie

threeInARow : Board -> Bool
threeInARow(b) = or(inARow(3,A,b),inARow(3,B,b))


-- Predefined operations
--
isValid : (Board,Position) -> Bool
isValid(b,p) = if b(p) == Empty then True else False


-- Game loop
--
tryMove : (Player,Board) -> (Board,Player)
tryMove(p,b) = let pos = input in
                   if isValid(b,pos) then (place(p,b,pos),next(p))
                                     else (b,p)

loop : (Player,Board) -> (Player,Board)
loop(p,b) = while not(gameOver(b)) do tryMove(p,b)

play : (Player,Board) -> Player|Tie
play(a,b) = outcome(loop(a,b))
-- play(a,b) = outcome(while not(gameOver(b)) do tryMove(p,b))

result : Player|Tie
result = play(initialBoard,goFirst)
```

game TicTacToe

This code is what we want Tic Tac Toe to look like in our language. The language is similar to Haskell, which is

intended as it is supposed to bridge the gap before using a real programming language. The language forces the user to use a concrete syntax, while still doing a lot of work for the user through built-ins like nxn boards, and InARow, which finds if there are a specified number of pieces in a row anywhere on the board.

5 Retrospectives

Overall we did very well in the development of the project. There was a lot that was up in the air with this project and the main accomplishment of this part of the development was getting clear terms of what the expectations are for the project. It is an added bonus that we were able to get a functional prototype by the end of the period.

The only improvements that we wish were were able to do was get a clear definition of the scope of the project sooner. This would have allowed us to begin work on the project sooner. Regardless we made very good progress.

	Positives	Deltas	Actions
Week 4	We made a document with all of the board games that are on a grid and the constructs that they would require to implement. This will be useful for deciding what parts of the language need to be built in.	We will continue to work on this document adding more games and further categorizing what built ins the game needs to be functional	We will get together and work on adding and categorizing games together
Week 5	We designed a syntax that we can present to our client to work from. We finished the required documents, and worked with our client to make sure they matched his expectations.	We will iterate on the syntax from our clients feedback and decide on specific things that are required for the language and what can be optional	We will meet with our client to determine what aspects of the language he likes and work to modify our language to fit his expectations
Week 6	Finished the tech review and met again with our client with our refined syntax. We designed forms that students could use to develop in the language.	Our client told us to wait to design more on the language until he can further define the scope. We will instead get ahead on our documents.	We will meet to work on the documents.
Week 7	We will met with the client this week to finalize the DSL and finished the design document	We will work on implementing the DSL so that we can have a prototype to present to the client	We will talk about about major design characteristics of implementing the compiler
Week 8	We finished the design document as well as returning to characterizing games and their requirements. We need to be able to determine what different types of games we want to support to determine what built ins need to exist in the DSL.	Further characterizing games so that it will be easy to determine the built ins that are needed to implement the language.	Meet again to add more games to the list that we have and further categorize games that already exist
Week 9	We finished categorizing the games that we need to support and began to work on a flex and bison implementation of the DSL	We would like to finish a demo for next week that we can present to our client at the end of the week	We need to subdivide the tasks required to finish this prototype to show our client and finish work on them
Week 10	We finished the flex and bison implementation prototype for the DSL and demoed to the client for feedback.	We need to make changes to the DSL implementation based on his feedback	Divide the work that needs to be done into tasks and begin working on those tasks