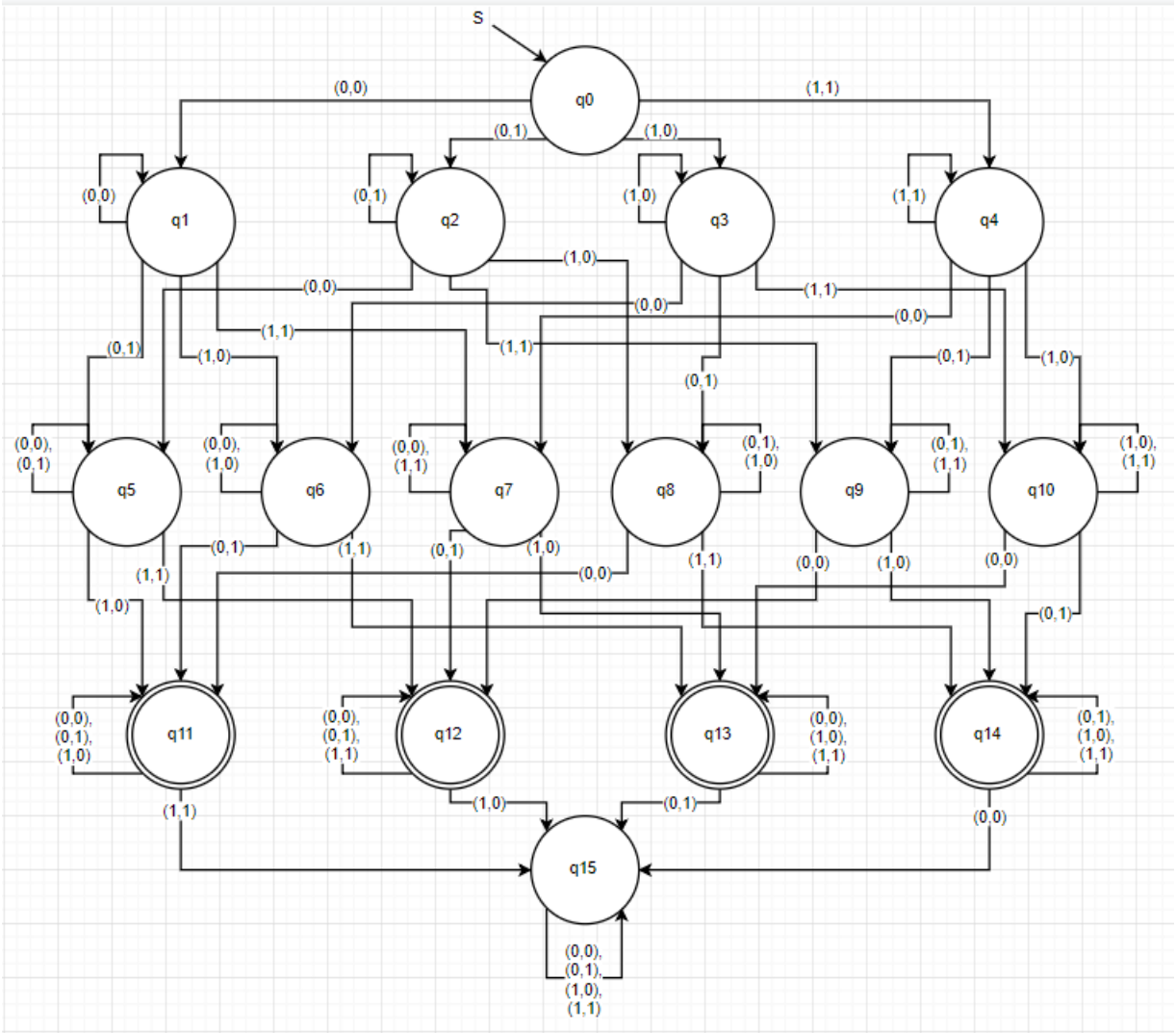
Philip Kirwin

CMPT440

The Complexity of Minesweeper

Often in software development, it is necessary to add layers of abstraction to an application so that the user does not need to understand complex underlaying processes to use the application properly. An excellent example of this principle is the game Minesweeper, wherein the player navigates through a series of board states without having to consider how the game processes their input and makes the necessary state transition. From the point of view of the user, they need only click on a tile and the application handles the rest silently until returning with an updated board state.

In order to make a DFA to model the potential states in a game of Minesweeper on a 2 by 2 board, the alphabet should be defined as the set of potential moves. That is, each transition on the DFA will be made based on an ordered pair of coordinates that each represent the position of a single tile on the board that can be clicked. Next, a generic DFA can be made that does not assume the location of the mine on the board. This means that in this DFA, all moves are legal and do not result in a game over, thereby creating four possible accepting states wherein the player has clicked on three separate tiles. The DFA therefore represents the transitions between having one tile selected, then two, then three where the game is won. It can also be assumed that selecting the fourth tile from one of these accepting states will always result in a game over (q15), as the board will always contain a single mine. Additionally, in every state other than q0, selecting a set of coordinates that has already been selected will simply loop back to the same state, as it is technically not an illegal move, though it does not advance the game in any way.



State	(0,0)	(0,1)	(1,0)	(1,1)
q0	q1	q2	q3	q4
q1	q1	q5	q6	q7
q2	q5	q2	q8	q9
q3	q6	q8	q3	q10
q4	q7	q9	q10	q4
q5	q5	q5	q11	q12
q6	q6	q11	q6	q13
q7	q7	q12	q13	q7
q8	q11	q8	q8	q14
q9	q12	q9	q14	q9
q10	q13	q14	q10	q10
q11	q11	q11	q11	q15
q12	q12	q12	q15	q12
q13	q13	q15	q13	q13
q14	q15	q14	q14	q14
q15	q15	q15	q15	q15

State	Description
q0	Game Start
q1	(0,0)
q2	(0,1)
q3	(1,0)
q4	(1,1)
q5	(0,0), (0,1)
q6	(0,0), (1,0)
q7	(0,0), (1,1)
q8	(0,1), (1,0)
q9	(0,1), (1,1)
q10	(1,0), (1,1)
q11	(0,0), (0,1), (1,0)
q12	(0,0), (0,1), (1,1)
q13	(0,0), (1,0), (1,1)
q14	(0,1), (1,0), (1,1)
q15	Game Over

The next step of this project would be to implement this DFA into a Java program using a table-driven approach. Having done this, logic would need to be implemented that would allow a mine to be placed onto the board at random. This should be somewhat simple, as a placeMine() method could be written to select the index of one member of the alphabet at random, then alter the delta array such that every transition made on the chosen alphabet member will become a transition to q15, the game over state. This will then be a better representation of an actual game of Minesweeper since only one of the four accepting states in the generic DFA will still be reachable after this method is called.

The program will also need to be able to properly parse a sequence of coordinates as input, then output whether this sequence results in a won game. The game is won if the DFA ends in an accepting state, but the program should be able to tell if the game has been lost by the selection of the tile containing the mine or if the game is incomplete by not selecting enough different tiles.