University of Sheffield
Zahra hasan a alhilal (acb20za)

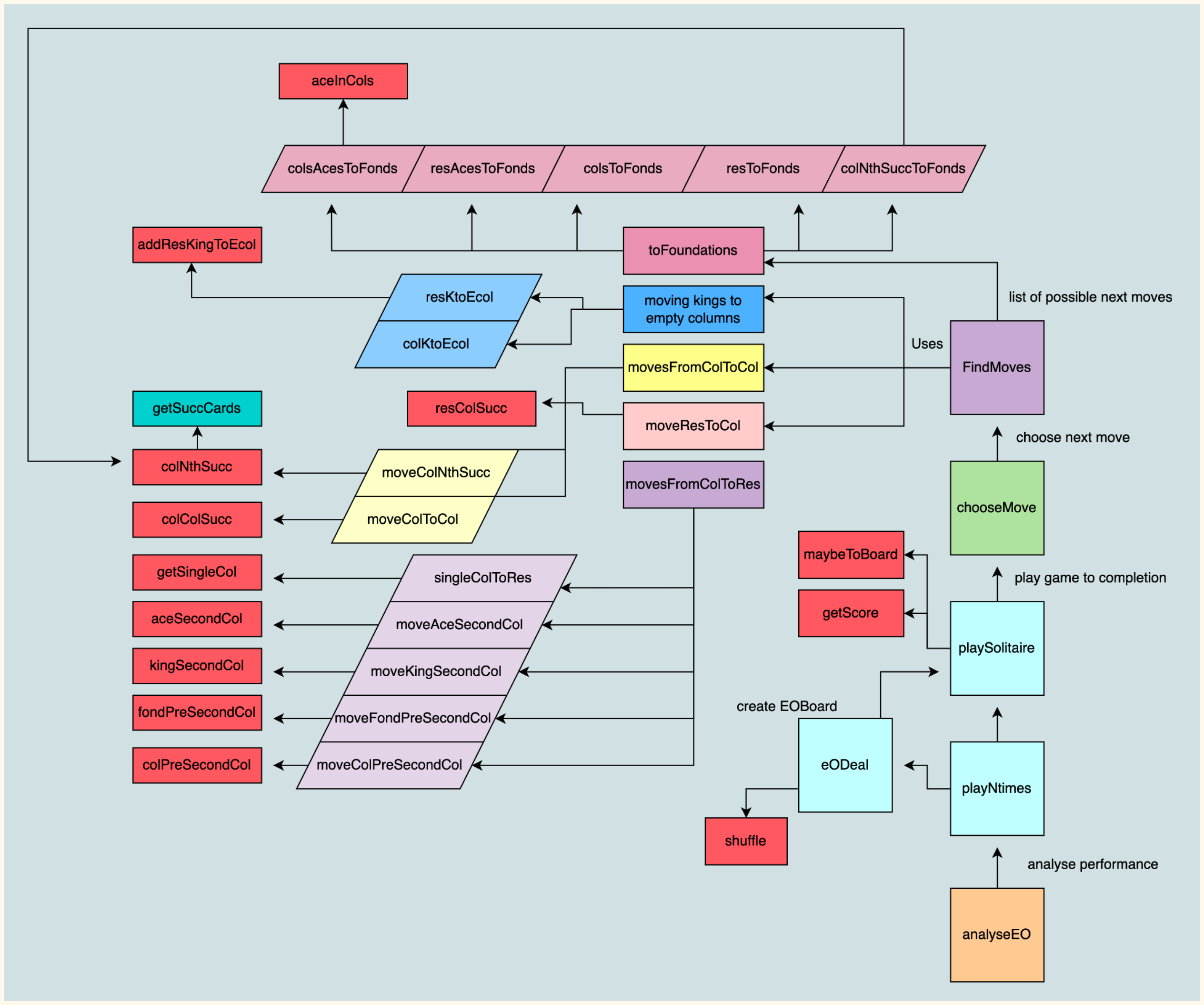
# COM2108 Assignment report.

—

## Eight Of/Spider  Solitaire games data structures.

| Data Type | Definition |
|---|---|
| Suit | Enum of all the four possible suits |
| Pip | Enum of all the 13th possible pips |
| Card with two constructors | |
| UCard | Faced up card that points to a pair of pip and suit |
| DCard | Faced down card that points to a pair of pip and suit |
| Deck | List of cards |
| Foundation (fond) | List of cards (13 at most) |
| Column [[col]] | List of list of cards |
| Reserve (res) | List of cards |
| Stock (stk) | List of cards |
| Board with two constructors | |
| EOBoard | Eight of board has (fond, [[col]], res) |
| SBoard | Spider board has (fond, [[col]], stk) |

# Eight Of Solitaire game design in haskell.



## Game Starter functions:

- eODeal
  - takes a number of the type (StdGen) and returns a randomly shuffled deck layed out as an eight of solitaire  board.
- playSolitaire
  - Takes a board and play it to completion then returns a score calculated by how many cards were moved to the foundations
- playNtimes
  - Takes an initial seed and number of games to play, then seed this to eODeal n times and play each board and play it to completion (playSolitaire)

## Game results analysing function:

analyseEO : takes a seed and number of games to play then apply playNtimes and calculate how many games are winning games and the average score.
Results of running analyseEO:

| seed | Number of games | wins | average |
|---|---|---|---|
| | | | |

Notes: this part could not be completed because the code runs into infinite loops and stops working.

## Game Playing Top functions:

- findMoves
  - takes a board and returns a list of all board states after performing a single move from that initial board.
  - The list is ordered so that it has the best moves “in my opinion and trial and error” in order.
- chooseMove
  - Chooses the next move to take which would be the first in the list of findMoves.

## Breaking down findMoves by defining all the possible single moves directions :

- Moving an ace from column to foundations
- Moving an ace from reserved to foundations
- Moving an card from to column foundations
- Moving a card from reserved to foundations
- Moving a king to an empty column
- Moving a card from reserved to column
- Moving  a card from column to column
- Moving a list of successors cards from column to column
- Moving a card from column to reserved

## Breaking toFoundations steps:

- colsAcesToFonds
  - colHeads (Get all column heads)
  - aceInCols (Find an ace in colHeads)
  - Add it to fonds (taking the head form aceInCols)
- resAcesToFonds
  - Filter all aces in res and add them to fonds
- colsToFonds
  - When a card in fonds have its successor  in colHeads, move that successor card to fonds
- resToFonds
  - When a card in fonds have its successor in res, move that successor card to fonds
- Return EOBoard

## Breaking down moving a King to an empty column:

- resKtoEcol (moving a king from res to Ecol , should be called when there is only one Ecol)
  - isEcol (Check if there is an empty column)
  - resKings (Get all kings from res)
  - Check if there is a king in res (resKings /= [])
  - addResKingToEcol (Move head resKings to the empty column)
  - Remove head resKings from res
- colKtoEcol (moving a king from colHeads to Ecol, should be called when there is only one empty column)
  - isEcol (Check if there is an empty column)
  - isKingCol (Check if there is a king in colHeads)
  - addColKingToEcol (Move a head king to an empty column)

## Breaking down reserved to column:

- resColSucc (Check if a card in res has its successor in colHeads)
- moveResToCol (Move the card in res and place it on top of its predecessor in colHeads)

## Breaking down column to column:

There can be a move of a single card or a move of a list of successor cards

1. Move of a single card
    a. colColSucc (Check if a card in colHeads has its successor in colHeads)
    b. moveColToCols (Move the card in colHeads and place it on top of its predecessor in colHeads)
2. Move of a list of successors cards
    a. colNthSucc uses getSuccCards to get the ordered successors cards of a column in cols
    b. moveColNthSucc (Move the list of successor cards and place them on top of their predecessor head, length res+length of successor cards  MUST be <= 8 )

## Breaking down column to reserved:

When moving a card from column to reserved it's best to move the card that is in a column of length one and it is not a king using  singleColToRes, then second best is to move the card that expose:

1. aceSecondCol (An ace) using moveAceSecondCol
2. kingSecondCol (A king and there is an empty column) using  moveKingSecondCol
3. fondPreSecondCol (A card that has its predecessor in foundation) using moveFondPreSecondCol
4. colPreSecondCol (A card that has its predecessor in colHeads) using moveColPreSecondCol

## Spider Of Solitaire game design in haskell.

Game starter function:
sDeal : deals two pack of cards to a spider of solitaire board.
Basic functions :
Switch : takes a card and change it’s type, if it’s faced up then turn it down and if it's faced down then turn it up.
switchCol: takes a column and turns all cards down except for the head.