Assignment 4, Part 1, Specification

SFWR ENG 2ME4

April 13, 2019

This Module Interface Specification (MIS) document contains modules, types and methods for implementing the state of a game of Life.

GameBoard Module

Module

GameBoard

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

GameBoard

Exported Access Programs

Routine name	In	Out	Exceptions
new GameBoard	File		badbit, failbit, invalid_argument
get		Seq of Seq of bool	
size_c		$\mathbb{N}$	
size_r		$\mathbb{N}$	
update			

Semantics

State Variables

raw: seq of bool board: seq of raw

State Invariant

None

Assumptions & Design Decisions

- The GameBoard constructor is called before any other access routine is called for that object. The constructor can only be called once. The constructor reads configuration from file. File should have format

  ***##**

  ****

  ##*****

  Where # - cell with life, * - without each raw should have the same amount of cells

Access Routine Semantics

new GameBoard(f):

- transition: $board := while(line! = EOF), raw.clear()(\forall i \in line.size() : line[i] == \#?raw.add(true) : raw.add(false)), board.add(raw)$
- output: none
- exception: $exc := (file.open(f) exeptions)||(line[i]! = \#||line[i]! = *||\exists i, j \in board.size() : board[i].size()! = board[j].size()) \implies invalid\_argument$

get():

- output: $out := board$
- exception: none

size_c():

- output: $out := board.size()$

size_r():

- output: $out := board[0].size()$

update():

- transition: $board := updated\_buffer()$

Local Variables

temp_board : seq of seq of bool

Local Functions

updated_buffer : void → seq of seq of bool
updated_buffer() ≡ $\forall i \in board.size() : \forall j \in board[i].size() : temp\_board = update\_sell(i, j)$
update_sell : $\mathbb{N} \times \mathbb{N} \rightarrow$ bool
updated_sell(i, j) ≡

board[i][j] == true	neighbors_number(i,j) < 2	false
	neighbors_number(i,j) > 3	false
	neighbors_number(i,j) > 1 and neighbors_number(i,j) < 4	false
$board[i][j] == false$	neighbors_number(i,j) == 3	true
	neighbors_number(i,j) ! = 3	false

- exception: i < 0 ‖ i > board.size() ‖ j < 0 ‖ j > board[i].size() → out_of_range

neighbors_number : $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$
neighbors_number(i, j) ≡ $(+\forall c \in i \pm 1 : \forall d \in j \pm 1 : (c! = i \&\& d! = j \&\& c < board.size() \&\& c > 0 \&\& d < board[c].size() \&\& d > 0) : if(board[c][d]) : 1)$

- exception: i < 0 ‖ i > board.size() ‖ j < 0 ‖ j > board[i].size() → out_of_range