Game of life, Specification

Behzad Khamneli

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

GameBoard ADT Module

Template Module

Game

Uses

N/A

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
new Game	string	Game	out_of_range, invalid_argument
status			
getCoordinates		seq of (seq of string)	
setAlive	$\mathbb{N}, \mathbb{N}$		out_of_range
setDead	$\mathbb{N}, \mathbb{N}$		out_of_range

Semantics

Environmental Variable

fileName: Input file with alive cells.

State Variables

coordinates: seq of (seq of string)

State Invariant

$|coordinates| = 15$x15

Assumption & Design Decisions

- The format of the input file should be 'x,y' on each line. x represents row and y represents column.

- The top left cell is 0,0. Row starts from 0 and ends at 14. Column starts from 0 and ends at 14.
- The Game constructor is called before any other access routine is called on that instance. Once a Game has been created, the constructor will not be called on it again.
- For better scalability, this module is specified as an Abstract Data Type (ADT) instead of an Abstract Object. This would allow multiple games to be created and tracked at once by a client.
- The getter function is provided, through violating the property of being essential, to give a would-be view function easily(output) and integrated with a game system in the future.
- Size of the board is 15x15.

Access Routine Semantics

Game(FileName):

- transition: Read the points from the file and use those points as index of the array. The input file has location of alive cells in the following format (on each line) : x,y x represents row and y represents column.
  coordinates := new seq of (seq of string) such that for each line in the input file: (Row_col_range(x,y) ⇒ coordinates[x][y] = "yes")
- exception: exc := (File does not exist ⇒ invalid_argument | not_row_col(x,y) ⇒ out_of_range)

status():

- transition : coordinates := new seq of (seq of string) coordinates such that $(\forall x, y : \mathbb{N} | Row\_col(x, y) : (count\_cells(x, y) < 2 \Rightarrow coordinates[x][y] = "dead" | count\_cells(x, y) = 3 \Rightarrow coordinates[x][y] = "yes" | count\_cells(x, y) > 3 \Rightarrow coordinates[x][y] = "dead"))$

getCoordinates():

- output: out := coordinates
- exception : none

setAlive(x,y):

- transition: coordinates[x][y] := "yes"
- exception: exc := (not_row_col(x, y) ⇒ out_of_range)

setDead(x,y):

- transition: coordinates[x][y] := "dead"
- exception: exc := (not_row_col(x, y) ⇒ out_of_range)

Local Types

Local Functions

Row_col: $\mathbb{N} \times \mathbb{N} \to B$
Row_col(row,col) $\equiv (row \geq 0 \wedge row < 15 \wedge col \geq 0 \wedge col < 15)$

not_row_col: $\mathbb{N} \times \mathbb{N} \to B$
not_row_col(row,col) $\equiv (row < 0 \wedge row \geq 15 \wedge col < 0 \wedge col \geq 15)$

count_cells: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$
count_cells(row,col) = count such that $(+count : \mathbb{N} | (\exists i, j : \mathbb{N} | i \in [row - 1...row + 1] \wedge j \in [col - 1...col + 1] : \neg(i = 0 \wedge j = 0) \wedge copy[i][j] = "yes" \wedge Row\_col(row + i, col + j)) : 1)$
copy $\equiv$ new seq of (seq of string) COPY such that $(\forall i, j : \mathbb{N} | Row\_col(i, j) : COPY[i][j] = coordinates[i][j])$

Display Module

Template Module

Show

Uses

N/A

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
output	seq of (seq of string)		
printer	seq of (seq of string)		

Semantics

Environmental Variable

output: output file with alive cells.

State Variables

None

State Invariant

None

Assumptions & Design Decisions

- Both output and printer functions take a double pointer to a string(coordinates) as a parameter.

Access Routine Semantics

output(seq of (seq of string)):

- output: Creates a new file "output.txt". Output file has coordinates of alive cells with the following format(on each line): x,y
  x represents row and y represents column.

- exception: none

printer(seq of (seq of string)):

- output: $(\forall i, j : \mathbb{N} | i \geq 0 \wedge i < 15 \wedge j \geq 0 \wedge j < 15 : (coordinates[i][j] = "yes" \Rightarrow print"0") \wedge (coordinates[i][j] = "dead" \Rightarrow print"."))$

- exception: none