

# Assignment 4, Specification

SFWR ENG 2AA4

April 13, 2019

Joshua Guinness, guinnesj, 400134735

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

In applying the specification, there will be cases that involve undefinedness. We will interpret undefinedness following [?]:

If  $p : \alpha_1 \times \dots \times \alpha_n \rightarrow \mathbb{B}$  and any of  $a_1, \dots, a_n$  is undefined, then  $p(a_1, \dots, a_n)$  is False. For instance, if  $p(x) = 1/x < 1$ , then  $p(0) = \text{False}$ . In the language of our specification, if evaluating an expression generates an exception, then the value of the expression is undefined.

# Cell ADT Module

## Module

Cell

## Uses

N/A

## Syntax

### Exported Constants

None

### Exported Types

Cell = ?

### Exported Access Programs

Routine name	In	Out	Exceptions
new Cell		Cell	none
new Cell	boolean	Cell	none
get_life		boolean	none
get_neighbours		int	none
set_life	boolean		none
set_neighbours	int		out_of_range

## Semantics

### State Variables

$S$ : boolean # *Alive or Dead*

$N$ : int # *Number of neighbors*

### State Invariant

$n \leq 8$

## Assumptions and Design Decisions

- The  $\text{Cell}(S)$  or  $\text{Cell}()$  constructor is called for each object instance before any other access routine is called for that object. The constructor can only be called once.

## Access Routine Semantics

$\text{new Cell}()$ :

- $S, N := \text{false}, 0$
- output:  $\text{out} := \text{self}$
- exception: none

$\text{new Cell}(s)$ :

- $S, N := s, 0$
- output:  $\text{out} := \text{self}$
- exception: none

$\text{get\_life}()$ :

- output:  $\text{out} := S$
- exception: none

$\text{get\_neighbours}()$ :

- output:  $\text{out} := N$
- exception: none

$\text{set\_life}(s)$ :

- transition:  $S := s$
- output: none
- exception: none

$\text{set\_neighbours}(n)$ :

- transition:  $N := n$
- output: none
- exception:  $\text{exc} := (n > 6 \Rightarrow \text{out\_of\_range})$

# Game Board ADT Module

## Template Module

BoardT

## Uses

Cell

View

## Syntax

### Exported Constants

None

### Exported Types

BoardT

### Exported Access Programs

Routine name	In	Out	Exceptions
new BoardT		BoardT	
new BoardT	seq of (seq of C)	BoardT	invalid_argument
next			
view			

## Semantics

### State Variables

$C$ : seq of (seq of Cell) #2D Array of Cells

### State Invariant

$|seq\ of\ Cell| = |seq\ of\ (seq\ of\ Cell)|$  #2D array is a perfect square

## **Assumptions & Design Decisions**

- The BoardT constructor is called before any other access routine is called on that instance. Once a BoardT has been created, the constructor will not be called on it again.
- The seq of (seq of C) that is passed to the constructor is a perfect square. This means that both sequences are of the same length.
- 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.

## **Access Routine Semantics**

## **Local Types**

## **Local Functions**

## Critique of Design