INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

UNIVERSITY OF WESTMINSTER

6SENG005C.1 Formal Methods

Coursework

Structure Diagram for the Asteroids arcade game & Invariant Descriptions

Author – Nethmi Mohotti

IIT ID – 20200486

UoW ID – w1830188

Module Leader: Dr. Thilini Piyatilake

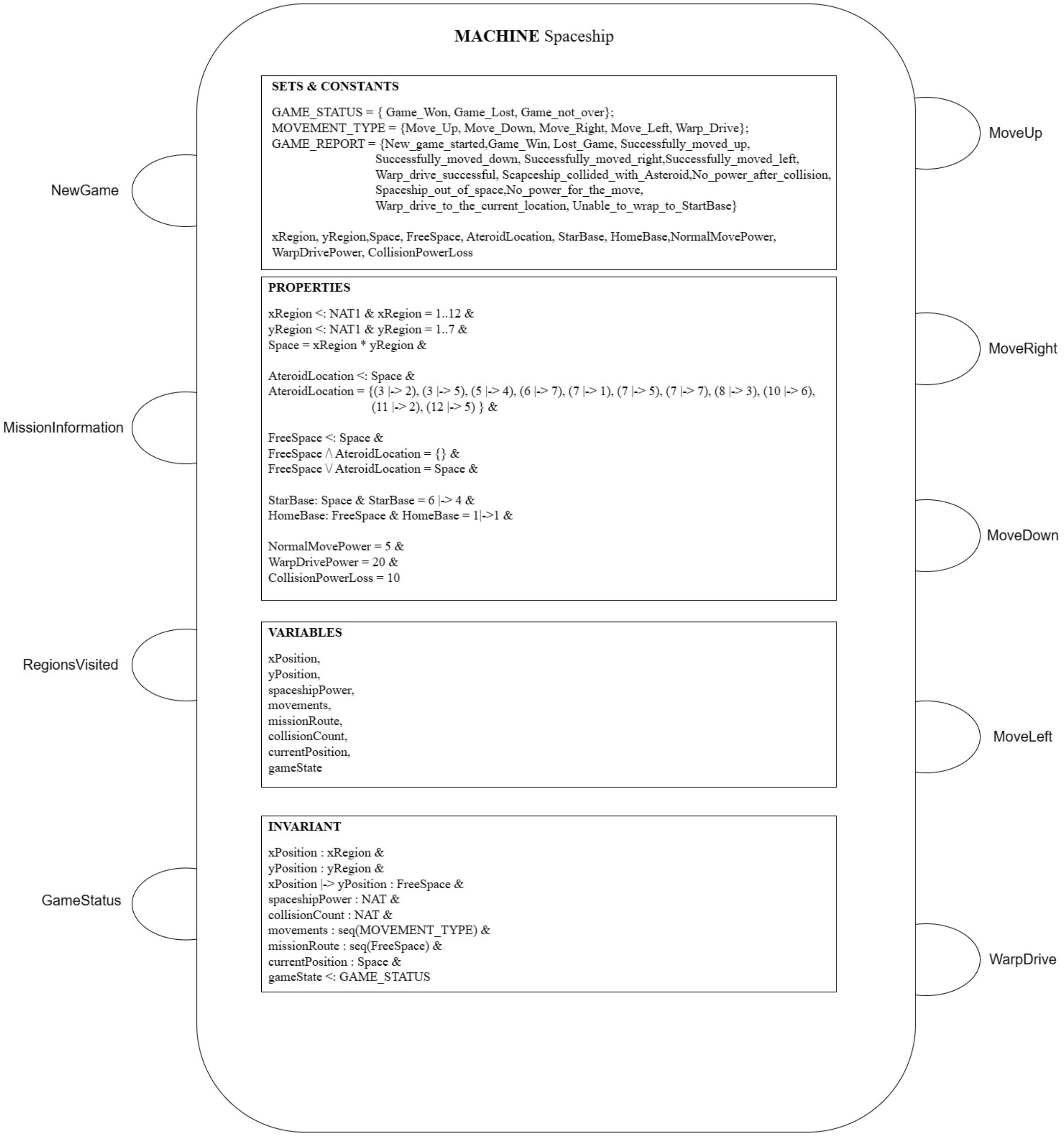
Submission Date: 09th January 2024

Contents

List of Tables

List of Figures

1.0 Abstract Machine Structure Diagram



```
MACHINE Spaceship

SETS & CONSTANTS

GAME_STATUS = { Game_Won, Game_Lost, Game_not_over};
MOVEMENT_TYPE = {Move_Up, Move_Down, Move_Right, Move_Left, Warp_Drive};
GAME_REPORT = {New_game_started,Game_Win, Lost_Game, Successfully_moved_up,
                    Successfully_moved_down, Successfully_moved_right,Successfully_moved_left,
                    Warp_drive_successful, Scapceship_collided_with_Asteroid,No_power_after_collision,
                    Spaceship_out_of_space,No_power_for_the_move,
                    Warp_drive_to_the_current_location, Unable_to_wrap_to_StartBase}

xRegion, yRegion,Space, FreeSpace, AteroidLocation, StarBase, HomeBase,NormalMovePower,
WarpDrivePower, CollisionPowerLoss

PROPERTIES

xRegion <: NAT1 & xRegion = 1..12 &
yRegion <: NAT1 & yRegion = 1..7 &
Space = xRegion * yRegion &

AteroidLocation <: Space &
AteroidLocation = {(3 |-> 2), (3 |-> 5), (5 |-> 4), (6 |-> 7), (7 |-> 1), (7 |-> 5), (7 |-> 7), (8 |-> 3), (10 |-> 6),
                   (11 |-> 2), (12 |-> 5) } &

FreeSpace <: Space &
FreeSpace /\ AteroidLocation = {} &
FreeSpace \/ AteroidLocation = Space &

StarBase: Space & StarBase = 6 |-> 4 &
HomeBase: FreeSpace & HomeBase = 1|->1 &

NormalMovePower = 5 &
WarpDrivePower = 20 &
CollisionPowerLoss = 10

VARIABLES

xPosition,
yPosition,
spaceshipPower,
movements,
missionRoute,
collisionCount,
currentPosition,
gameState

INVARIANT

xPosition : xRegion &
yPosition : yRegion &
xPosition |-> yPosition : FreeSpace &
spaceshipPower : NAT &
collisionCount : NAT &
movements : seq(MOVEMENT_TYPE) &
missionRoute : seq(FreeSpace) &
currentPosition : Space &
gameState <: GAME_STATUS
```

Figure 1.1: Structure Diagram

2.0 System Invariant Explanation

When a variable is declared in a B machine the name of the variable must be provided, its properties and type should be defined, and finally the variable must be provided with an initial value. In order to the satisfy these requirements there are different clauses defined in the machine to supply each requirement as following:

- Declaration of a variable is done under the VARIABLES clause which provide the name of the variable.
- The information such as the type and the properties are listed under the INVARIANT clause.
- The initial value assignment for the variables is done under the INITIALISATION clause.

The following provide the explanation of the INVARIANTs specified in the Spaceship abstract machine:

Invariant	**Explanation**

`xPosition: xRegion &`
`yPosition: yRegion &`
`xPosition|-> yPosition: Space`
	The current X, Y position of the spaceship where the `xPosition` is an element of the constant defined as `yPosition` which in the set of Natural numbers from 1 to the boundary 12 and `xRegion` is an element of the constant defined as `yRegion` which in the set of Natural numbers from 1 to the boundary 7.
	The invariant is specified that the ordered pair, simplified as the (`xPosition`, `yPosition`) must be an element of the Space.

`spaceshipPower : NAT`	The `spaceshipPower` should be an element of the Natural number set from 0. Which explains as the power of the spaceship could take any value from 0 to infinity.

`collisionCount : NAT`	The `collisionCount` should be an element of the Natural number set from 0. Which explains as the number of collisions could be any whole number starting from 0 according to the collisions occurred while playing the game.
`movements: seq(MOVEMENT_TYPE)`	The movement types that are taking place while a game is played is saved into the `movements` variable as a list of elements, where all the elements should be of the MOVEMENT_TYPE set.
`missionRoute: seq(FreeSpace)`	The mission routes that are taking place while a game is played is saved into the `missionRoute` variable as a list of elements, where all the elements are of constant FreeSpace.
`currentPosition: FreeSpace`	The current position of the SpaceShip should be inside the FreeSpace set where the spaceship has avoided Asteroids.
`gameState <: GAME_STATUS`	The `gameState` variable is specified so that it can hold values that are members of the `GAME_STATUS`.

Table 2.1: Invariant Description