Informatics Institute of Technology

Formal Methods

6SENG005C.1

Course Work

Report

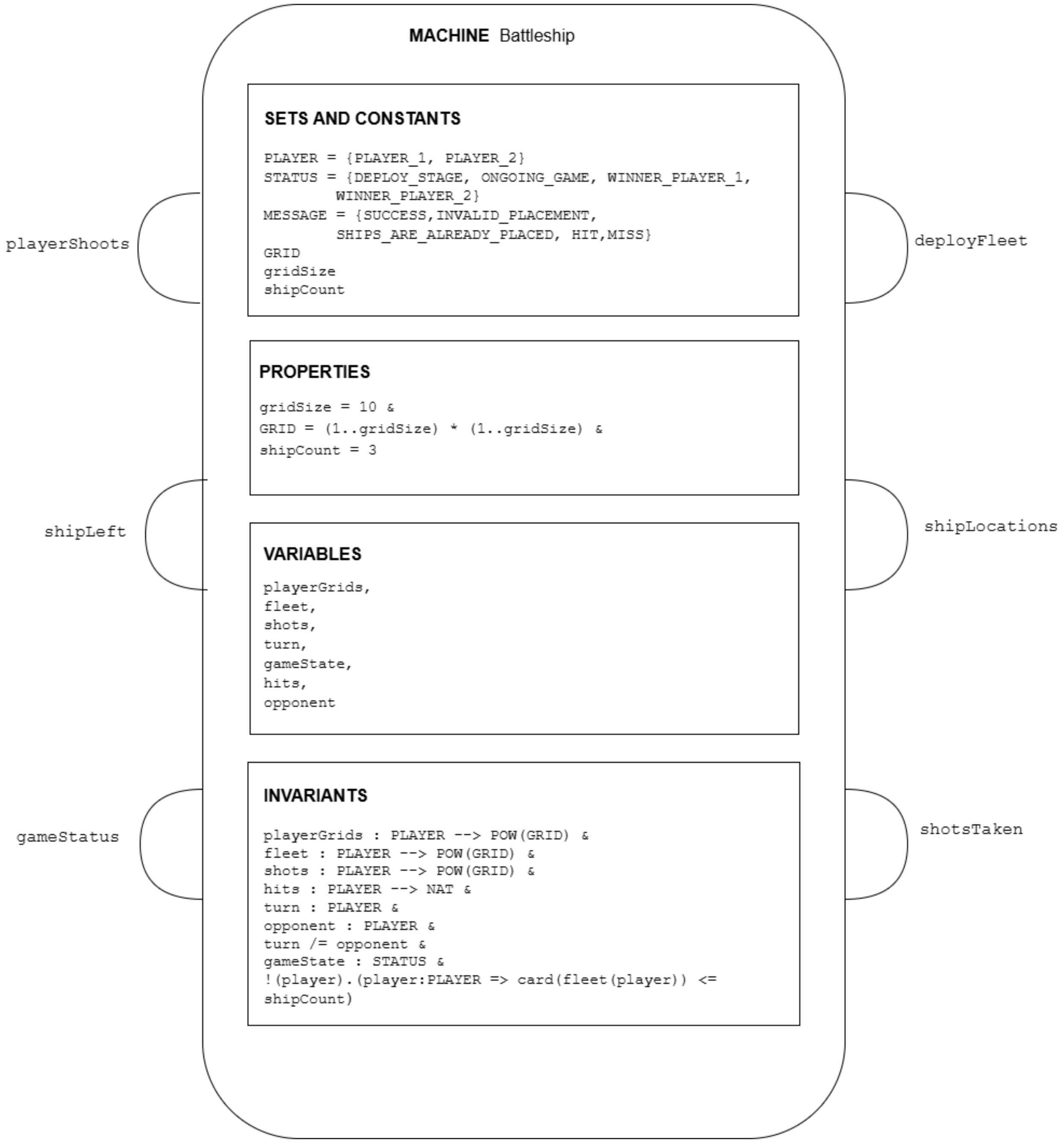
Pasindu Anuradha Godakanda

Uow Number – w1871482

IIT Number – 20212026

Table of Contents

1 Machine Structure Diagram



SETS AND CONSTANTS

```
PLAYER = {PLAYER_1, PLAYER_2}
STATUS = {DEPLOY_STAGE, ONGOING_GAME, WINNER_PLAYER_1,
        WINNER_PLAYER_2}
MESSAGE = {SUCCESS,INVALID_PLACEMENT,
        SHIPS_ARE_ALREADY_PLACED, HIT,MISS}
GRID
gridSize
shipCount
```

PROPERTIES

```
gridSize = 10 &
GRID = (1..gridSize) * (1..gridSize) &
shipCount = 3
```

VARIABLES

```
playerGrids,
fleet,
shots,
turn,
gameState,
hits,
opponent
```

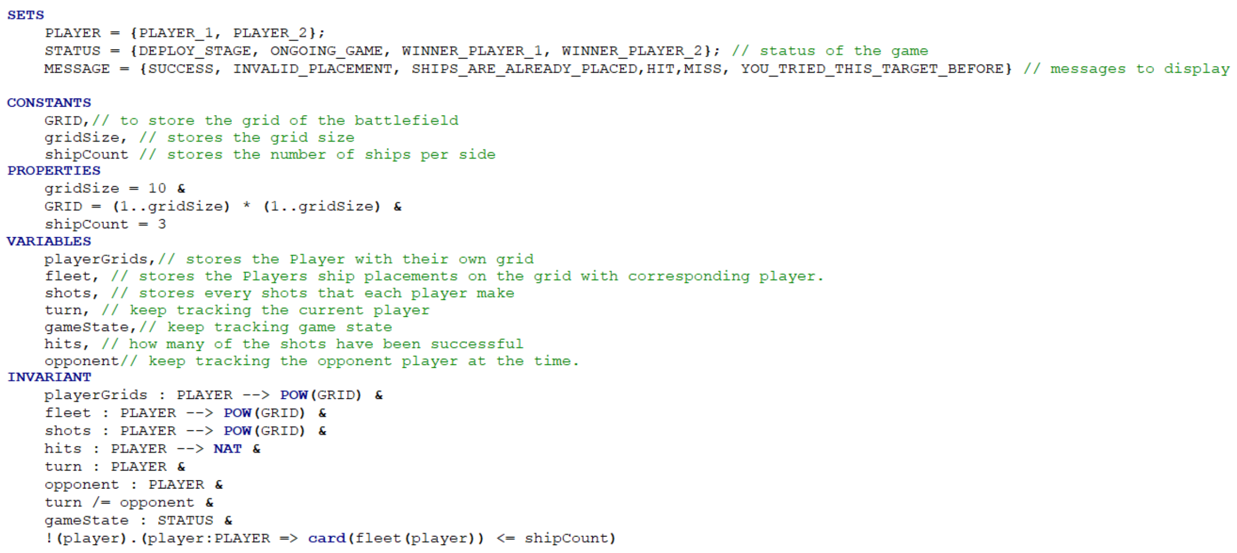
INVARIANTS

```
playerGrids : PLAYER --> POW(GRID) &
fleet : PLAYER --> POW(GRID) &
shots : PLAYER --> POW(GRID) &
hits : PLAYER --> NAT &
turn : PLAYER &
opponent : PLAYER &
turn /= opponent &
gameState : STATUS &
!(player).(player:PLAYER => card(fleet(player)) <=
shipCount)
```

Figure 1: Machine Structure Diagram

2 Justifications:

Detailed explanations for the assigned and utilized SETS, VARIABLES and INVARIENTS are provided below.



```
SETS
    PLAYER = {PLAYER_1, PLAYER_2};
    STATUS = {DEPLOY_STAGE, ONGOING_GAME, WINNER_PLAYER_1, WINNER_PLAYER_2}; // status of the game
    MESSAGE = {SUCCESS, INVALID_PLACEMENT, SHIPS_ARE_ALREADY_PLACED,HIT,MISS, YOU_TRIED_THIS_TARGET_BEFORE} // messages to display

CONSTANTS
    GRID,// to store the grid of the battlefield
    gridSize, // stores the grid size
    shipCount // stores the number of ships per side
PROPERTIES
    gridSize = 10 &
    GRID = (1..gridSize) * (1..gridSize) &
    shipCount = 3
VARIABLES
    playerGrids,// stores the Player with their own grid
    fleet, // stores the Players ship placements on the grid with corresponding player.
    shots, // stores every shots that each player make
    turn, // keep tracking the current player
    gameState,// keep tracking game state
    hits, // how many of the shots have been successful
    opponent// keep tracking the opponent player at the time.
INVARIANT
    playerGrids : PLAYER --> POW(GRID) &
    fleet : PLAYER --> POW(GRID) &
    shots : PLAYER --> POW(GRID) &
    hits : PLAYER --> NAT &
    turn : PLAYER &
    opponent : PLAYER &
    turn /= opponent &
    gameState : STATUS &
    !(player).(player:PLAYER => card(fleet(player)) <= shipCount)
```

Figure 2: SETS, VARIABLES and INVARIENTS

2.1 SETS & CONSTANTS

SET & CONSTANTS	Explanation
`PLAYER = {PLAYER_1, PLAYER_2}`	This Represents the two participants in the game. Battleship game involves two players, requiring this set to distinguish between them.
`STATUS = {DEPLOY_STAGE, ONGOING_GAME, WINNER_PLAYER_1, WINNER_PLAYER_2}`	This Represents different game status. This is requires to keep track the game progression throughout the different stages of the game.
`MESSAGE = {SUCCESS, INVALID_PLACEMENT, SHIPS_ARE_ALREADY_PLACED,HIT,MISS, YOU_TRIED_THIS_TARGET_BEFORE}`	This Represents the feedback messages. This Requires to provide relevant game notifications during the play.
`GRID`	This is the game board coordinates which defines the playable area for ship placement and targeting.

`gridSize`	This constant used for set the grid dimensions. As for this specification it uses 10 x 10 grid, hence the grid size would be 10.
`shipCount`	This is the number of ships that each player can deploy. This fixes ship count for balanced gameplay.

2.2 VARIABLES & INVARIENTS

VARIABLES & INVARIENTS	**Explanation**

`playerGrids`
`playerGrids : PLAYER --> POW(GRID)`
	playerGrids uses to map players to their own grids and this keeps track of each player's board setup. Each player must have a grid of valid coordinates. This ensures proper grid assignment.

`Fleet`
`fleet : PLAYER --> POW(GRID)`
`!(player).(player:PLAYER => card(fleet(player)) <= shipCount)`
	Stores ships' positions of each player for ship placement validation and hit detection.
	Each player's fleet must consist of valid coordinates within the grid.
	A player's fleet cannot exceed the allowed ship count.

`Shots`
`shots : PLAYER --> POW(GRID)`
	Tracks all shots fired by each player for prevent duplicate shots and helps determine game progress.
	All the shots must be valid grid coordinates.

`Hits`
`hits : PLAYER --> NAT`
	Tracks successful hits per player. This requires for determine when a player wins the game.
	Since negative hits are logically impossible and there can be an event that a player don't have any successful hit. Hence this tracks natural numbers including 0.

`Turn`
`turn : PLAYER`
	The game is alternates between player's turn, hence the turn variable tracks the current player who's attacking at the moment.
	The current turn must belong to a valid player and it must be always assigned.

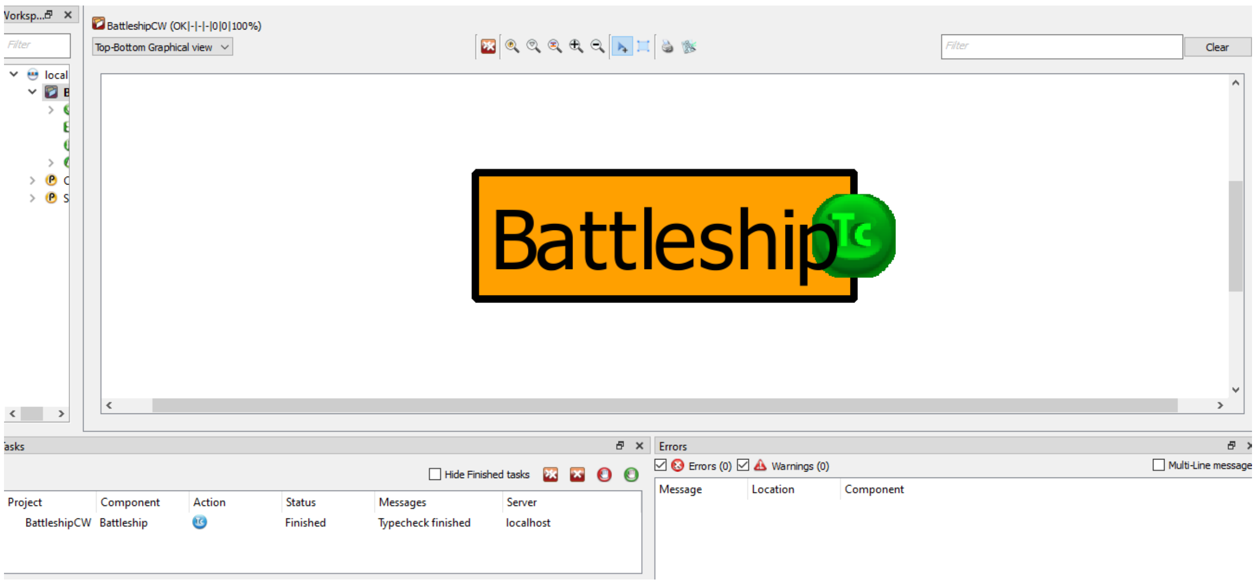
`opponent`
`opponent : PLAYER`
`turn /= opponent`
	This is the opposing player during the current turn. This requires for determine whose ships are targeted.

	Opponent also should be a valid player and since players cannot play against themselves, opponent cannot be the turn at the same time.

`gameState`
`gameState : STATUS`
	This Represents the current state of the game. This is requires to manage game flow from deployment to conclusion.
	gameState should be always a valid STATUS.

3 Machine Testing results:

3.1 Type Check Result



Project	Component	Action	Status	Messages	Server
BattleshipCW	Battleship		Finished	Typecheck finished	localhost

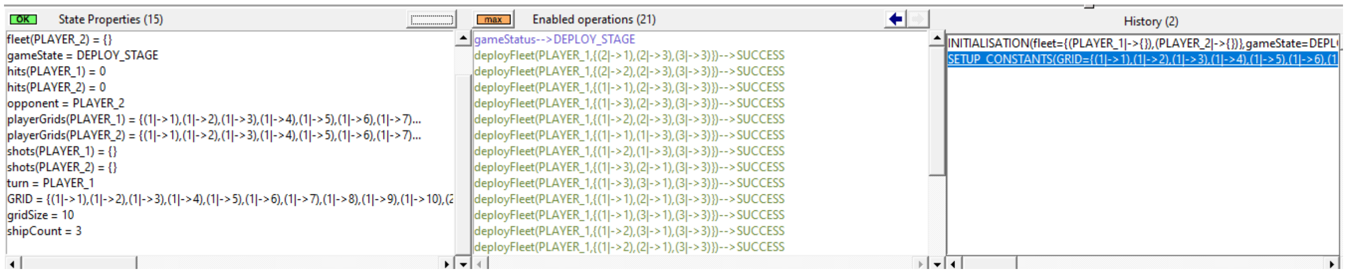
Message	Location	Component

Figure 3: Type Check result

3.2 Machine Testing

Test Case ID	Description	Input	Expected Output	Pass/Fail
TC-01	Verify Initial State	-	`gameState = DEPLOY_STAGE, turn = PLAYER_1`	Pass

Actual Output



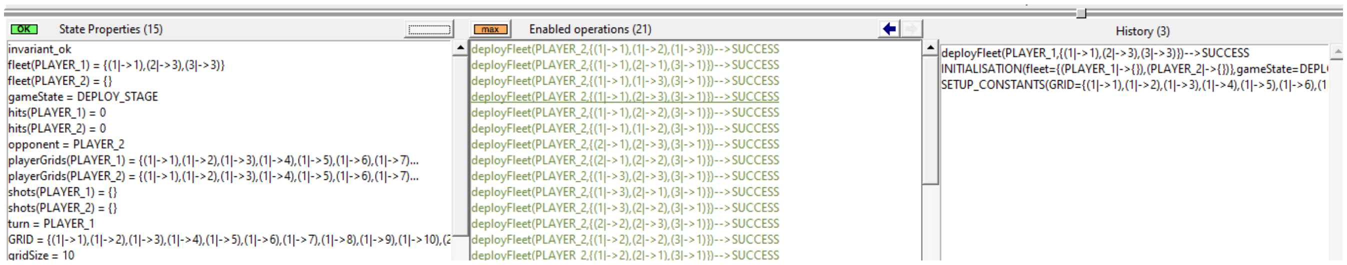
```
OK   State Properties (15)
fleet(PLAYER_2) = {}
gameState = DEPLOY_STAGE
hits(PLAYER_1) = 0
hits(PLAYER_2) = 0
opponent = PLAYER_2
playerGrids(PLAYER_1) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
playerGrids(PLAYER_2) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
shots(PLAYER_1) = {}
shots(PLAYER_2) = {}
turn = PLAYER_1
GRID = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7),(1|->8),(1|->9),(1|->10),(2
gridSize = 10
shipCount = 3
```

```
max   Enabled operations (21)
gameStatus-->DEPLOY_STAGE
deployFleet(PLAYER_1,{(2|->1),(2|->3),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(2|->2),(2|->3),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(2|->3),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->3),(2|->3),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->2),(2|->3),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->3),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->2),(1|->3),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->3),(2|->1),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->3),(3|->1),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(2|->1),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(3|->1),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->2),(3|->1),(3|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->2),(2|->1),(3|->3)})-->SUCCESS
```

```
History (2)
INITIALISATION(fleet={(PLAYER_1|->{}),(PLAYER_2|->{})},gameState=DEPL
SETUP_CONSTANTS(GRID={(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1
```

TC-02	Valid Fleet Deployment	`deployFleet(PLAYER_1, {(1,1), (2,3), (3,3)})`	`report = SUCCESS`	Pass

Actual Output



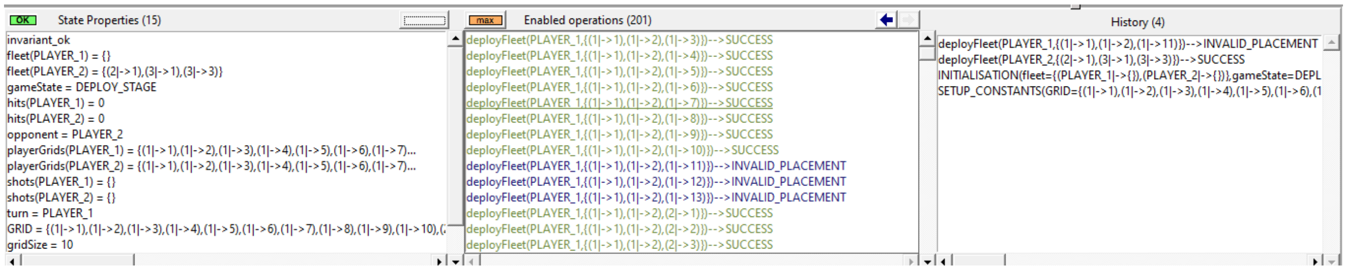
```
OK   State Properties (15)
invariant_ok
fleet(PLAYER_1) = {(1|->1),(2|->3),(3|->3)}
fleet(PLAYER_2) = {}
gameState = DEPLOY_STAGE
hits(PLAYER_1) = 0
hits(PLAYER_2) = 0
opponent = PLAYER_2
playerGrids(PLAYER_1) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
playerGrids(PLAYER_2) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
shots(PLAYER_1) = {}
shots(PLAYER_2) = {}
turn = PLAYER_1
GRID = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7),(1|->8),(1|->9),(1|->10),(2
gridSize = 10
```

```
max   Enabled operations (21)
deployFleet(PLAYER_2,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
deployFleet(PLAYER_2,{(1|->1),(2|->1),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(1|->1),(1|->3),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(1|->1),(2|->3),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(1|->1),(2|->2),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(1|->1),(1|->2),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(2|->1),(2|->3),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(2|->1),(2|->2),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(1|->3),(2|->3),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(1|->3),(2|->1),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(1|->3),(2|->2),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(2|->2),(2|->3),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(1|->2),(2|->2),(3|->1)})-->SUCCESS
deployFleet(PLAYER_2,{(1|->2),(2|->1),(3|->1)})-->SUCCESS
```

```
History (3)
deployFleet(PLAYER_1,{(1|->1),(2|->3),(3|->3)})-->SUCCESS
INITIALISATION(fleet={(PLAYER_1|->{}),(PLAYER_2|->{})},gameState=DEPL
SETUP_CONSTANTS(GRID={(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1
```

TC-03	Invalid Fleet Deployment	`deployFleet(PLAYER_1, {(1,1), (1,2), (1,11)})`	`report = INVALID_PLACEMENT`	Pass

Actual Output



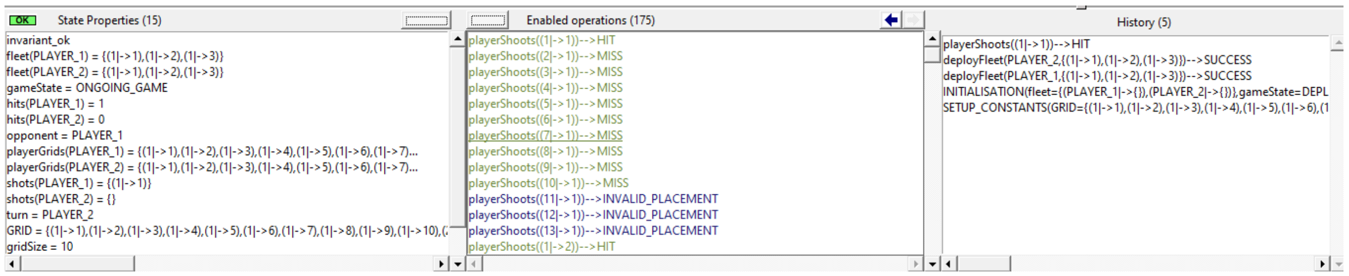
```
OK   State Properties (15)
invariant_ok
fleet(PLAYER_1) = {}
fleet(PLAYER_2) = {(2|->1),(3|->1),(3|->3)}
gameState = DEPLOY_STAGE
hits(PLAYER_1) = 0
hits(PLAYER_2) = 0
opponent = PLAYER_2
playerGrids(PLAYER_1) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
playerGrids(PLAYER_2) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
shots(PLAYER_1) = {}
shots(PLAYER_2) = {}
turn = PLAYER_1
GRID = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7),(1|->8),(1|->9),(1|->10),(2
gridSize = 10
```

```
max   Enabled operations (201)
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->4)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->5)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->6)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->7)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->8)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->9)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->10)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->11)})-->INVALID_PLACEMENT
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->12)})-->INVALID_PLACEMENT
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->13)})-->INVALID_PLACEMENT
deployFleet(PLAYER_1,{(1|->1),(1|->2),(2|->1)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(2|->2)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(2|->3)})-->SUCCESS
```

```
History (4)
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->11)})-->INVALID_PLACEMENT
deployFleet(PLAYER_2,{(2|->1),(3|->1),(3|->3)})-->SUCCESS
INITIALISATION(fleet={(PLAYER_1|->{}),(PLAYER_2|->{})},gameState=DEPL
SETUP_CONSTANTS(GRID={(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1
```

TC-04	Valid Shooting (Hit)	`playerShoots((1,1))` (if `(1,1)` in opponent fleet)	`report = HIT`	Pass

Actual Output



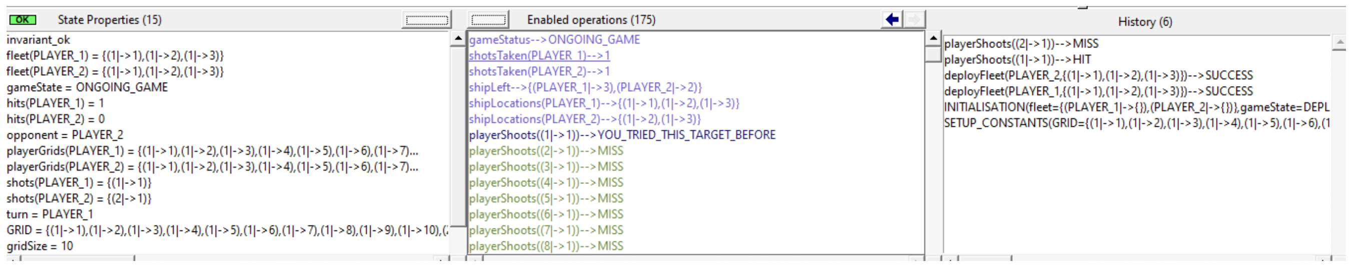
```
OK  State Properties (15)
invariant_ok
fleet(PLAYER_1) = {(1|->1),(1|->2),(1|->3)}
fleet(PLAYER_2) = {(1|->1),(1|->2),(1|->3)}
gameState = ONGOING_GAME
hits(PLAYER_1) = 1
hits(PLAYER_2) = 0
opponent = PLAYER_1
playerGrids(PLAYER_1) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
playerGrids(PLAYER_2) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
shots(PLAYER_1) = {(1|->1)}
shots(PLAYER_2) = {}
turn = PLAYER_2
GRID = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7),(1|->8),(1|->9),(1|->10),(
gridSize = 10
```

```
Enabled operations (175)
playerShoots((1|->1))-->HIT
playerShoots((2|->1))-->MISS
playerShoots((3|->1))-->MISS
playerShoots((4|->1))-->MISS
playerShoots((5|->1))-->MISS
playerShoots((6|->1))-->MISS
playerShoots((7|->1))-->MISS
playerShoots((8|->1))-->MISS
playerShoots((9|->1))-->MISS
playerShoots((10|->1))-->MISS
playerShoots((11|->1))-->INVALID_PLACEMENT
playerShoots((12|->1))-->INVALID_PLACEMENT
playerShoots((13|->1))-->INVALID_PLACEMENT
playerShoots((1|->2))-->HIT
```

```
History (5)
playerShoots((1|->1))-->HIT
deployFleet(PLAYER_2,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
INITIALISATION(fleet={(PLAYER_1|->{}),(PLAYER_2|->{})},gameState=DEPL
SETUP_CONSTANTS(GRID={(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1
```

TC-05	Valid Shooting (Miss)	`playerShoots((2,1))` (if `(2,1)` not in opponent fleet)	`report = MISS`	Pass

Actual Output



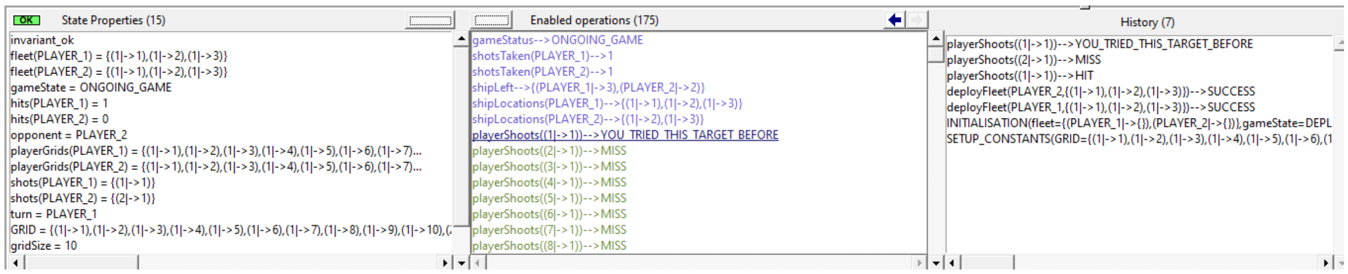
```
OK  State Properties (15)
invariant_ok
fleet(PLAYER_1) = {(1|->1),(1|->2),(1|->3)}
fleet(PLAYER_2) = {(1|->1),(1|->2),(1|->3)}
gameState = ONGOING_GAME
hits(PLAYER_1) = 1
hits(PLAYER_2) = 0
opponent = PLAYER_2
playerGrids(PLAYER_1) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
playerGrids(PLAYER_2) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
shots(PLAYER_1) = {(1|->1)}
shots(PLAYER_2) = {(2|->1)}
turn = PLAYER_1
GRID = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7),(1|->8),(1|->9),(1|->10),(
gridSize = 10
```

```
Enabled operations (175)
gameStatus-->ONGOING_GAME
shotsTaken(PLAYER_1)-->1
shotsTaken(PLAYER_2)-->1
shipLeft-->{(PLAYER_1|->3),(PLAYER_2|->2)}
shipLocations(PLAYER_1)-->{(1|->1),(1|->2),(1|->3)}
shipLocations(PLAYER_2)-->{(1|->2),(1|->3)}
playerShoots((1|->1))-->YOU_TRIED_THIS_TARGET_BEFORE
playerShoots((2|->1))-->MISS
playerShoots((3|->1))-->MISS
playerShoots((4|->1))-->MISS
playerShoots((5|->1))-->MISS
playerShoots((6|->1))-->MISS
playerShoots((7|->1))-->MISS
playerShoots((8|->1))-->MISS
```

```
History (6)
playerShoots((2|->1))-->MISS
playerShoots((1|->1))-->HIT
deployFleet(PLAYER_2,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
INITIALISATION(fleet={(PLAYER_1|->{}),(PLAYER_2|->{})},gameState=DEPL
SETUP_CONSTANTS(GRID={(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1
```

TC-06	Repeated Targeting	`playerShoots((1,1))` (already targeted)	`report = YOU_TRIED_THIS_TARGET_BEFORE`	Pass

Actual Output



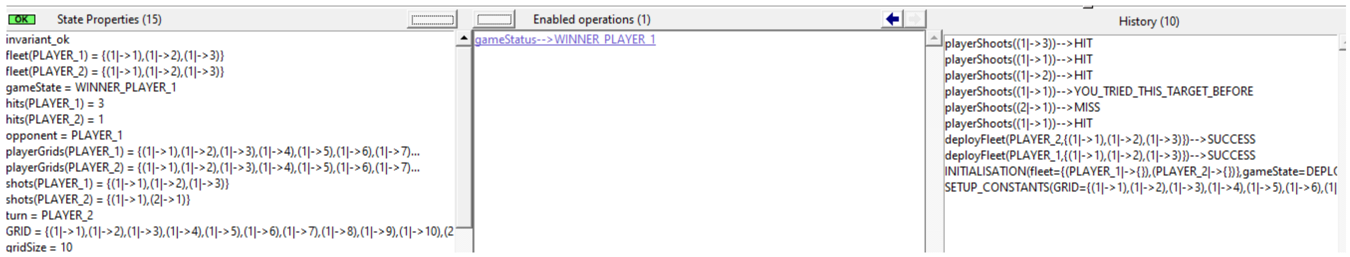
```
OK  State Properties (15)
invariant_ok
fleet(PLAYER_1) = {(1|->1),(1|->2),(1|->3)}
fleet(PLAYER_2) = {(1|->1),(1|->2),(1|->3)}
gameState = ONGOING_GAME
hits(PLAYER_1) = 1
hits(PLAYER_2) = 0
opponent = PLAYER_2
playerGrids(PLAYER_1) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
playerGrids(PLAYER_2) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
shots(PLAYER_1) = {(1|->1)}
shots(PLAYER_2) = {(2|->1)}
turn = PLAYER_1
GRID = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7),(1|->8),(1|->9),(1|->10),(
gridSize = 10
```

```
Enabled operations (175)
gameStatus-->ONGOING_GAME
shotsTaken(PLAYER_1)-->1
shotsTaken(PLAYER_2)-->1
shipLeft-->{(PLAYER_1|->3),(PLAYER_2|->2)}
shipLocations(PLAYER_1)-->{(1|->1),(1|->2),(1|->3)}
shipLocations(PLAYER_2)-->{(1|->2),(1|->3)}
playerShoots((1|->1))-->YOU_TRIED_THIS_TARGET_BEFORE
playerShoots((2|->1))-->MISS
playerShoots((3|->1))-->MISS
playerShoots((4|->1))-->MISS
playerShoots((5|->1))-->MISS
playerShoots((6|->1))-->MISS
playerShoots((7|->1))-->MISS
playerShoots((8|->1))-->MISS
```

```
History (7)
playerShoots((1|->1))-->YOU_TRIED_THIS_TARGET_BEFORE
playerShoots((2|->1))-->MISS
playerShoots((1|->1))-->HIT
deployFleet(PLAYER_2,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
INITIALISATION(fleet={(PLAYER_1|->{}),(PLAYER_2|->{})},gameState=DEPL
SETUP_CONSTANTS(GRID={(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1
```

TC-07	Winning Condition Check	Sink all PLAYER_2's ships	`gameState = WINNER_PLAYER_1`	Pass

Actual Output



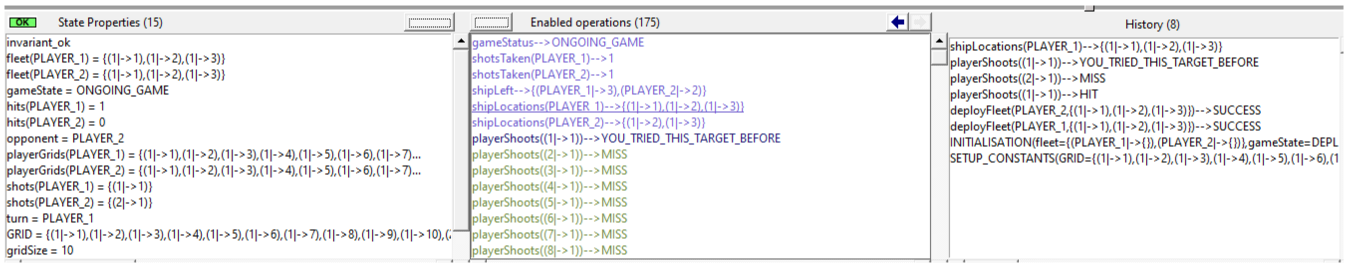
```
OK  State Properties (15)
invariant_ok
fleet(PLAYER_1) = {(1|->1),(1|->2),(1|->3)}
fleet(PLAYER_2) = {(1|->1),(1|->2),(1|->3)}
gameState = WINNER_PLAYER_1
hits(PLAYER_1) = 3
hits(PLAYER_2) = 1
opponent = PLAYER_1
playerGrids(PLAYER_1) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
playerGrids(PLAYER_2) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
shots(PLAYER_1) = {(1|->1),(1|->2),(1|->3)}
shots(PLAYER_2) = {(1|->1),(2|->1)}
turn = PLAYER_2
GRID = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7),(1|->8),(1|->9),(1|->10),(2
gridSize = 10
```

```
Enabled operations (1)
gameStatus-->WINNER_PLAYER_1
```

```
History (10)
playerShoots((1|->3))-->HIT
playerShoots((1|->1))-->HIT
playerShoots((1|->2))-->HIT
playerShoots((1|->1))-->YOU_TRIED_THIS_TARGET_BEFORE
playerShoots((2|->1))-->MISS
playerShoots((1|->1))-->HIT
deployFleet(PLAYER_2,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
INITIALISATION(fleet={(PLAYER_1|->{}),(PLAYER_2|->{})},gameState=DEPL(
SETUP_CONSTANTS(GRID={(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|
```

TC-07	Ship Location Query	`shipLocations(PLAYER_1)`	Remaining ship positions	Pass

Actual Output



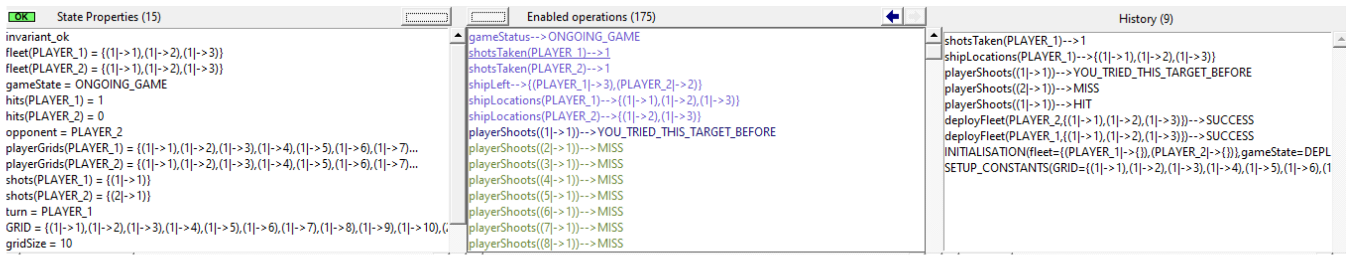
```
OK   State Properties (15)
invariant_ok
fleet(PLAYER_1) = {(1|->1),(1|->2),(1|->3)}
fleet(PLAYER_2) = {(1|->1),(1|->2),(1|->3)}
gameState = ONGOING_GAME
hits(PLAYER_1) = 1
hits(PLAYER_2) = 0
opponent = PLAYER_2
playerGrids(PLAYER_1) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
playerGrids(PLAYER_2) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
shots(PLAYER_1) = {(1|->1)}
shots(PLAYER_2) = {(2|->1)}
turn = PLAYER_1
GRID = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7),(1|->8),(1|->9),(1|->10),(
gridSize = 10
```

```
Enabled operations (175)
gameStatus-->ONGOING_GAME
shotsTaken(PLAYER_1)-->1
shotsTaken(PLAYER_2)-->1
shipLeft-->{(PLAYER_1|->3),(PLAYER_2|->2)}
shipLocations(PLAYER_1)-->{(1|->1),(1|->2),(1|->3)}
shipLocations(PLAYER_2)-->{(1|->2),(1|->3)}
playerShoots((1|->1))-->YOU_TRIED_THIS_TARGET_BEFORE
playerShoots((2|->1))-->MISS
playerShoots((3|->1))-->MISS
playerShoots((4|->1))-->MISS
playerShoots((5|->1))-->MISS
playerShoots((6|->1))-->MISS
playerShoots((7|->1))-->MISS
playerShoots((8|->1))-->MISS
```

```
History (8)
shipLocations(PLAYER_1)-->{(1|->1),(1|->2),(1|->3)}
playerShoots((1|->1))-->YOU_TRIED_THIS_TARGET_BEFORE
playerShoots((2|->1))-->MISS
playerShoots((1|->1))-->HIT
deployFleet(PLAYER_2,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
INITIALISATION(fleet={(PLAYER_1|->{}),(PLAYER_2|->{})},gameState=DEPL
SETUP_CONSTANTS(GRID={(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1
```

TC-09	Shot Count Query	`shotsTaken(PLAYER_1)`	Correct shot count	Pass

Actual Output



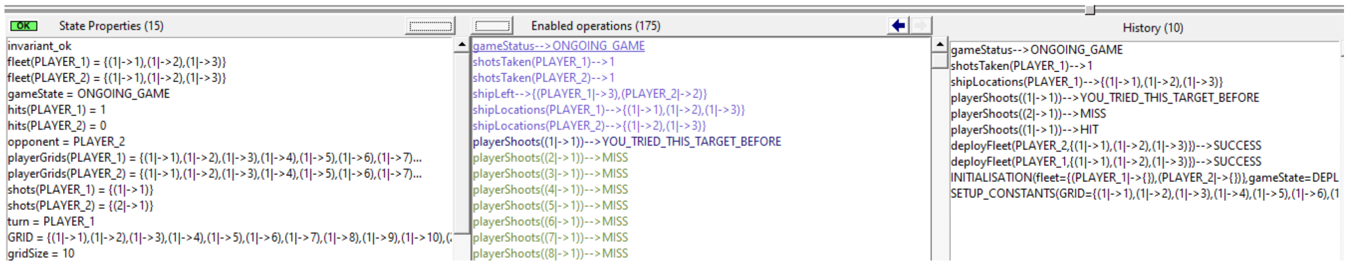
```
OK   State Properties (15)
invariant_ok
fleet(PLAYER_1) = {(1|->1),(1|->2),(1|->3)}
fleet(PLAYER_2) = {(1|->1),(1|->2),(1|->3)}
gameState = ONGOING_GAME
hits(PLAYER_1) = 1
hits(PLAYER_2) = 0
opponent = PLAYER_2
playerGrids(PLAYER_1) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
playerGrids(PLAYER_2) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
shots(PLAYER_1) = {(1|->1)}
shots(PLAYER_2) = {(2|->1)}
turn = PLAYER_1
GRID = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7),(1|->8),(1|->9),(1|->10),(
gridSize = 10
```

```
Enabled operations (175)
gameStatus-->ONGOING_GAME
shotsTaken(PLAYER_1)-->1
shotsTaken(PLAYER_2)-->1
shipLeft-->{(PLAYER_1|->3),(PLAYER_2|->2)}
shipLocations(PLAYER_1)-->{(1|->1),(1|->2),(1|->3)}
shipLocations(PLAYER_2)-->{(1|->2),(1|->3)}
playerShoots((1|->1))-->YOU_TRIED_THIS_TARGET_BEFORE
playerShoots((2|->1))-->MISS
playerShoots((3|->1))-->MISS
playerShoots((4|->1))-->MISS
playerShoots((5|->1))-->MISS
playerShoots((6|->1))-->MISS
playerShoots((7|->1))-->MISS
playerShoots((8|->1))-->MISS
```

```
History (9)
shotsTaken(PLAYER_1)-->1
shipLocations(PLAYER_1)-->{(1|->1),(1|->2),(1|->3)}
playerShoots((1|->1))-->YOU_TRIED_THIS_TARGET_BEFORE
playerShoots((2|->1))-->MISS
playerShoots((1|->1))-->HIT
deployFleet(PLAYER_2,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
INITIALISATION(fleet={(PLAYER_1|->{}),(PLAYER_2|->{})},gameState=DEPL
SETUP_CONSTANTS(GRID={(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1
```

TC-10	Game Status Query	`gameStatus()`	Current game state	Pass

Actual Output



```
OK   State Properties (15)
invariant_ok
fleet(PLAYER_1) = {(1|->1),(1|->2),(1|->3)}
fleet(PLAYER_2) = {(1|->1),(1|->2),(1|->3)}
gameState = ONGOING_GAME
hits(PLAYER_1) = 1
hits(PLAYER_2) = 0
opponent = PLAYER_2
playerGrids(PLAYER_1) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
playerGrids(PLAYER_2) = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7)...
shots(PLAYER_1) = {(1|->1)}
shots(PLAYER_2) = {(2|->1)}
turn = PLAYER_1
GRID = {(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1|->7),(1|->8),(1|->9),(1|->10),(
gridSize = 10
```

```
Enabled operations (175)
gameStatus-->ONGOING_GAME
shotsTaken(PLAYER_1)-->1
shotsTaken(PLAYER_2)-->1
shipLeft-->{(PLAYER_1|->3),(PLAYER_2|->2)}
shipLocations(PLAYER_1)-->{(1|->1),(1|->2),(1|->3)}
shipLocations(PLAYER_2)-->{(1|->2),(1|->3)}
playerShoots((1|->1))-->YOU_TRIED_THIS_TARGET_BEFORE
playerShoots((2|->1))-->MISS
playerShoots((3|->1))-->MISS
playerShoots((4|->1))-->MISS
playerShoots((5|->1))-->MISS
playerShoots((6|->1))-->MISS
playerShoots((7|->1))-->MISS
playerShoots((8|->1))-->MISS
```

```
History (10)
gameStatus-->ONGOING_GAME
shotsTaken(PLAYER_1)-->1
shipLocations(PLAYER_1)-->{(1|->1),(1|->2),(1|->3)}
playerShoots((1|->1))-->YOU_TRIED_THIS_TARGET_BEFORE
playerShoots((2|->1))-->MISS
playerShoots((1|->1))-->HIT
deployFleet(PLAYER_2,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
deployFleet(PLAYER_1,{(1|->1),(1|->2),(1|->3)})-->SUCCESS
INITIALISATION(fleet={(PLAYER_1|->{}),(PLAYER_2|->{})},gameState=DEPL
SETUP_CONSTANTS(GRID={(1|->1),(1|->2),(1|->3),(1|->4),(1|->5),(1|->6),(1
```

4 Graphical Visualization

Regarding the Battleship game B specification, Graphical visualization also implemented using an Animation Function. Following are the main Indicators and various states of the graphical visualization of the Battleship B specification.

4.1 Indicators

P1 Opposing Player (who takes the attack)

P2 Current Player (who attacks)

Missed shot

Unharmed Ship

Successful Hit (sinking Ship)

4.2 States of the Game

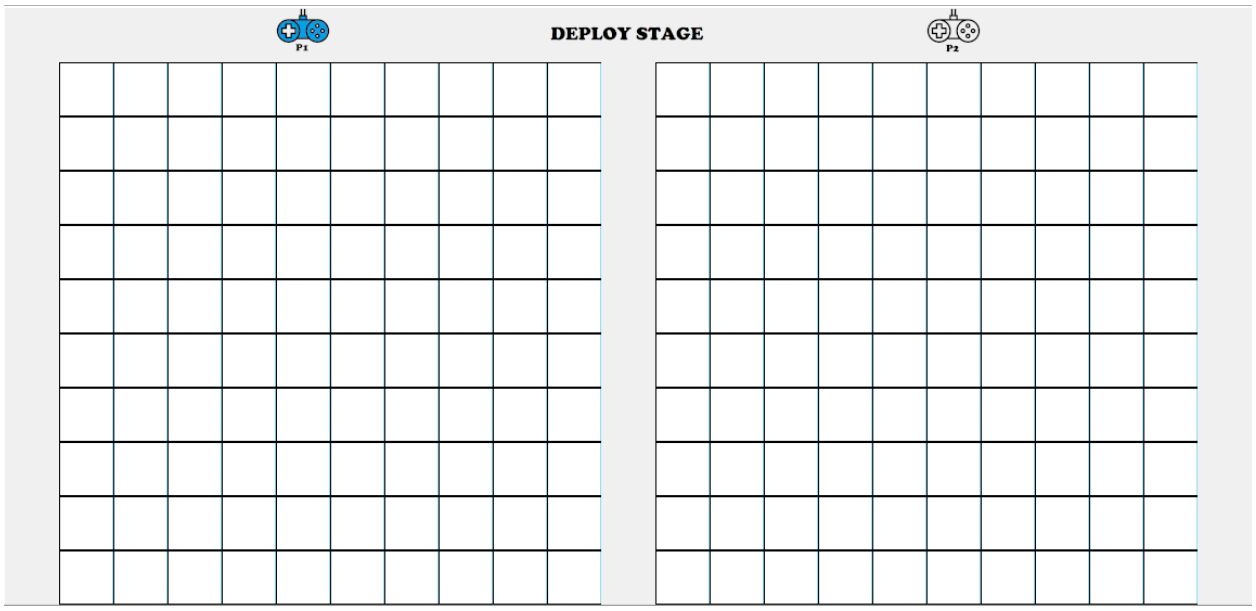


Figure 4: Deploy Game Stage (Initial Stage)

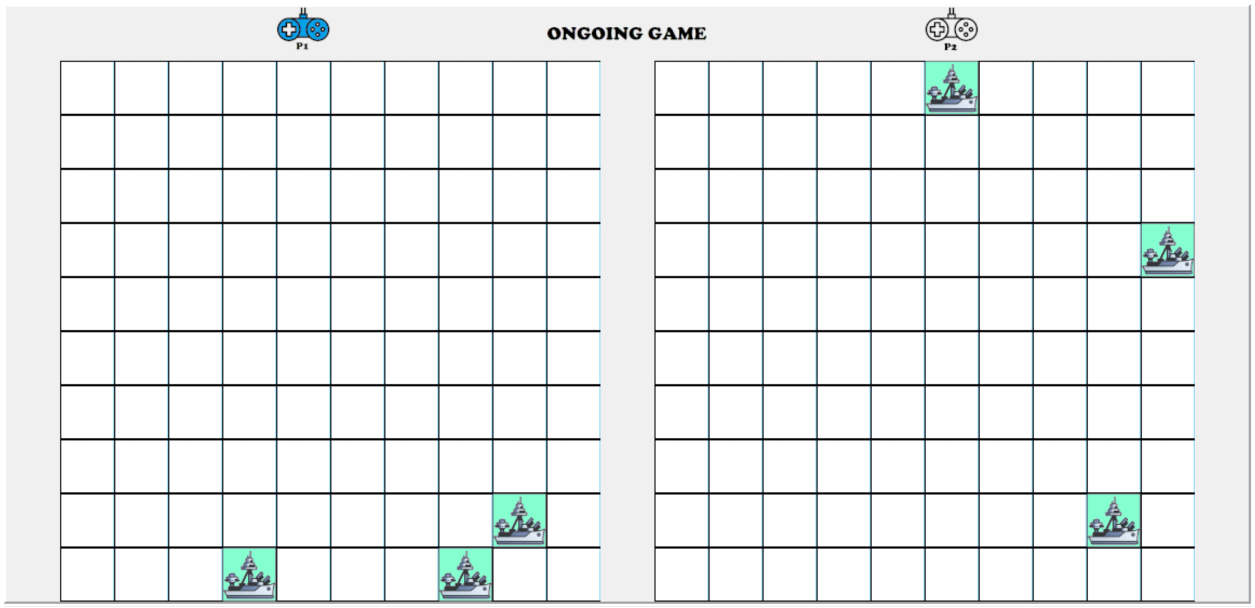


Figure 5: After the both the players placed their own fleets.

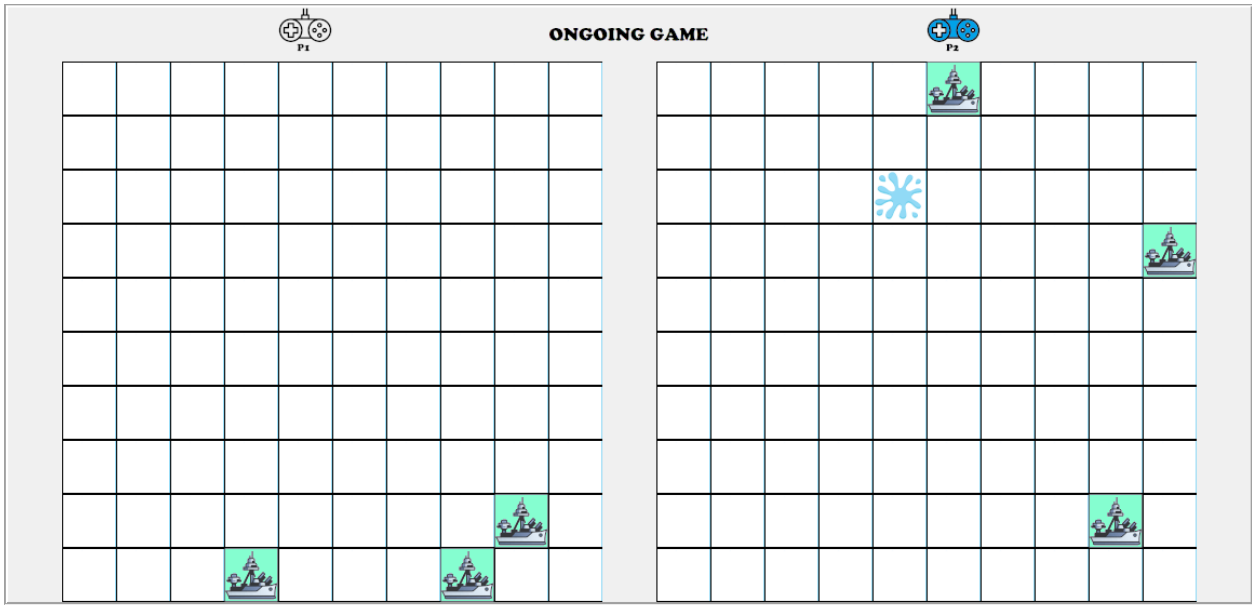


Figure 6: Player 1 fires a shot and Misses.

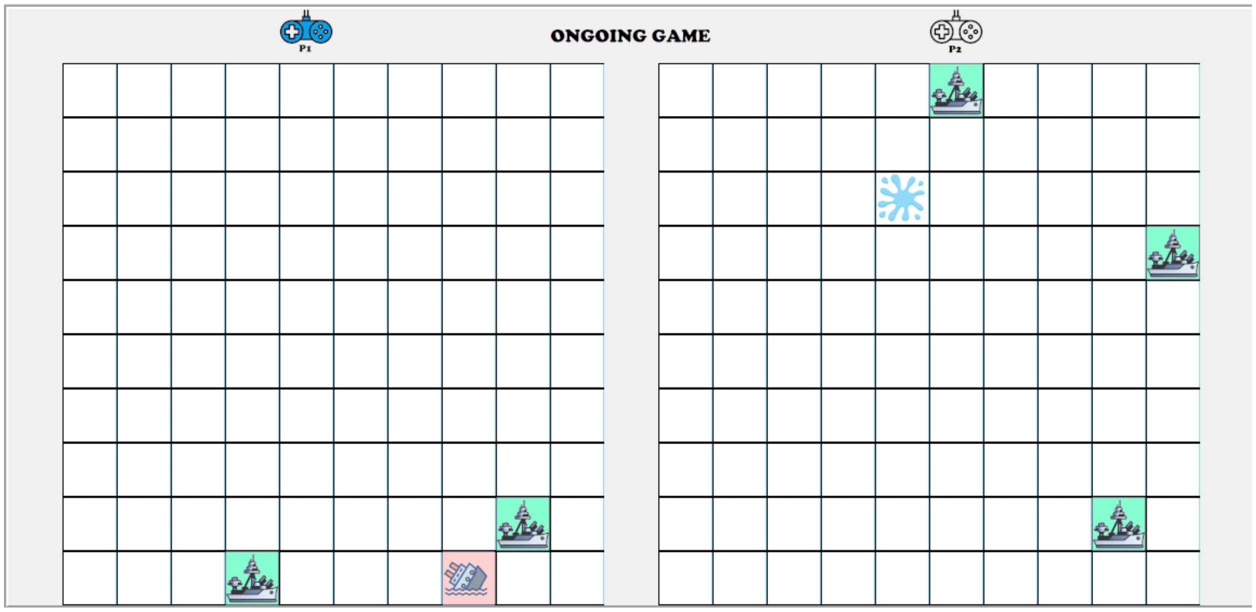


Figure 7: Player 2 Fires a shot and makes a successful hit.

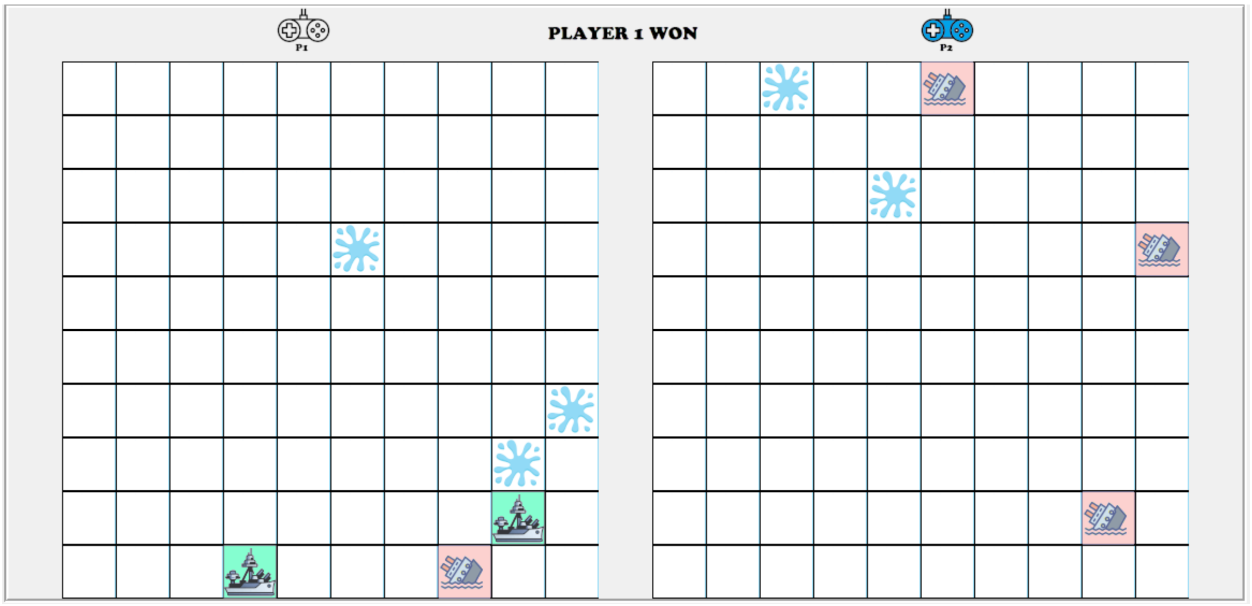


Figure 8: Player 1 sink all the ships of Player 2's Board and Win.

5 State Graph Visualization.

Since the State Graph visualization of the battleship B specification is too large to display inside this report it is attached as a separate PDF down below. This state graph visualization is captured during ONGOING_GAME state after deploy fleets and fire some shots.

Battleship.pdf