Jamila Magamedova

w1785175  University of Westminster

snakes and ladders:

b – specification

structure diagram



A screenshot of a computer screen

Description automatically generated

**Sets and Constants**

Define the fundamental elements and rules of the game, such as the board layout, snake and ladder positions, and valid dice values.

* **STATUS**: This set includes all possible game statuses (NewGameStarted, DescendedSnake, AscendedLadder, OverRoll, Victory, NormalMove).
* **BOARD**: Represents the game board squares, ranging from 1 to 100.
* **SNAKE\_POSITIONS**: Maps the heads of snakes to their tails on the board.
* **LADDER\_POSITIONS**: Maps the bottoms of ladders to their tops on the board.
* **FIRST\_SQUARE**: The starting square of the game, which is 1.
* **LAST\_SQUARE**: The final square of the game, which is 100.
* **DICE\_VALUES**: Represents possible dice values, ranging from 1 to 6.

**Variables**

Track the state of the game, including the player's position, dice rolls, turn count, and the sequence of squares visited.

* **current\_dice**: Holds the current value of the dice roll.
* **previous\_dice**: Stores the value of the previous dice roll.
* **player\_position**: Indicates the current position of the player on the board.
* **turn\_count**: Tracks the number of turns taken in the game.
* **snake\_count**: Counts how many times the player has landed on a snake.
* **ladder\_count**: Counts how many times the player has landed on a ladder.
* **path\_traversed**: Keeps a sequence of all squares the player has visited.

**Invariants**

Ensure that the game state remains valid at all times by enforcing constraints on the variables.

* **current\_dice : DICE\_VALUES**: Ensures that the current dice value is within the valid range of dice values.
* **previous\_dice : DICE\_VALUES / {0}**: Ensures that the previous dice value is within the valid range of dice values or is 0 (indicating no dice roll has occurred yet).
* **player\_position : BOARD**: Ensures that the player’s position is always within the valid range of board squares (1 to 100).
* **turn\_count : NAT**: Ensures that the number of turns taken is a natural number (0 or greater).
* **snake\_count : NAT**: Ensures that the number of times the player has encountered a snake is a natural number (0 or greater).
* **ladder\_count : NAT**: Ensures that the number of times the player has encountered a ladder is a natural number (0 or greater).
* **path\_traversed : seq(BOARD)**: Ensures that the sequence of visited squares only contains valid board positions.

**Initialisation**

The initialisation clause sets up the initial state of the game:

* **current\_dice := 1**: Sets the current dice value to 1.
* **previous\_dice := 0**: Indicates no previous dice roll has occurred.
* **player\_position := FIRST\_SQUARE**: Starts the player at the first square.
* **turn\_count := 0**: Initializes the turn count to 0.
* **snake\_count := 0**: Initializes the snake encounter count to 0.
* **ladder\_count := 0**: Initializes the ladder encounter count to 0.
* **path\_traversed := []**: Initializes the path traversed as an empty sequence.

**Operations**

Define the actions that can be taken during the game, such as rolling the dice, getting game statistics, viewing the path traversed, and resetting the game.

* **RollDice(rolled\_value)**:
  + **Preconditions**: Ensures the dice value is valid and the player hasn’t won yet.
  + **Updates**: Sets the previous dice value, increments the turn count, and updates the player’s position based on the dice roll.
  + **Scenarios**: Handles normal moves, encounters with snakes or ladders, winning the game, and over rolls.
* **Statistics**:
  + Outputs the current position, number of snakes encountered, number of ladders encountered, and number of turns taken.
* **Path**:
  + Outputs the sequence of visited squares.
* **ResetGame**:
  + Resets all game variables to their initial values and outputs a message indicating a new game has started.
* Define the actions that can be taken during the game, such as rolling the dice, getting game statistics, viewing the path traversed, and resetting the game.