# Project Summary

***Preface:*** *In the project summary section and propositions section, the labeling of the board’s squares is different to the labeling in the constraints section, model exploration section, and run.py file. This decision was made to make generalizing our model easier for the future however both are valid ways to represent the game. Refer to figure 1 when reading the summary and proposition sections and refer to figure 2 for the rest of the report and the code.*

*Given a pre-set snakes-and-ladders board (Figure 1), given a 3-sided dice with numbers 1, 2 and 3, and given 3 rolls, can a player, who starts at position X2,1, reach position X1,1 and win the game. The player, who starts at position X2,1, moves to the right from the starting position to X2,4, then climbs up to position X1,4 and then continues shifting left towards position X1,1. The arrow pointing downwards is a snake. If a player lands on the tail of an arrow, then the player automatically moves to the head of the arrow. The ladder moves a player in the opposite direction. If a player lands on the bottom of a ladder, the player automatically moves to the top of the ladder.*

*The player wins only if he/she lands exactly on position X1,1 within a given number of rolls. For example, if a player is situated at position X1,3, rolls a 2 and has not used all their potential rolls, then the player would land directly on position X1,1 and the player would win the game. Note that the player does not need to use all available rolls to reach the goal. If, however, the player rolls a 3 at position X1,3, and has not used all their potential rolls, then the player would not land directly on position X1,1 and would lose the game.*

*A player could also lose the game by not reaching position X1,1 within the allotted number of rolls. For example, if the limit on the number of rolls was 3, the player has already rolled 2 times, is at position X1,3 and then rolls a 1, the player ends up at position X2,2 and has exhausted all his/her rolls. Since the player has not reached position X1,1 and has no more rolls, the player loses.*

Diagram

Description automatically generated

Figure : Snakes-and-Ladders pre-set board with old square labeling

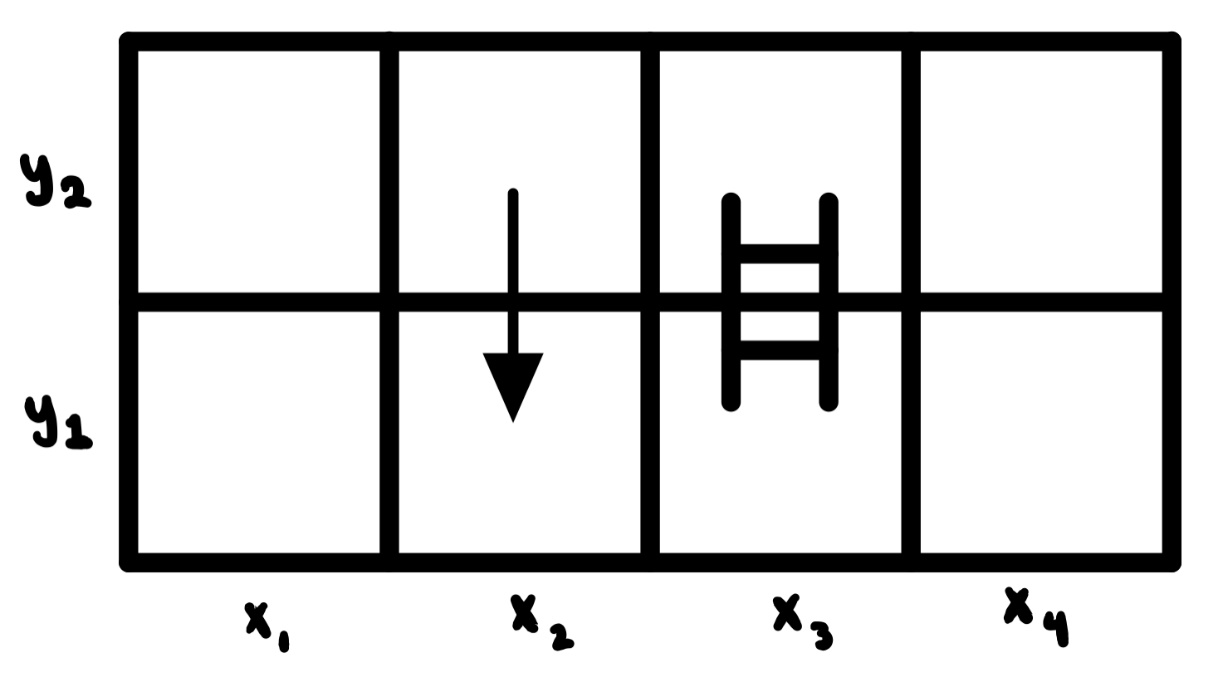


Figure 2: Snakes-and-Ladders pre-set board with updated square labeling

# Propositions

* Xi,j,k: True if the player is at position (i,j) at roll k where k = 1 represents the first roll
* Ry: True when y is rolled
* W: True if the player has won (i.e. landed perfectly on position X1,1 within the allotted number of rolls
  + X1,1,k W: If a player lands on position X1,1 at turn k such that k is not greater than 3, then the player has won
* L: True if the player has lost (i.e. did not land perfectly at position X1,1 in the allotted number of rolls
  + (X1,3,2 R3) (X1,3,1 R3) (X1,4,3) (X1,3,3) (X2,2,3)L: If the player rolls a 3 after 1 or 2 turns at position X1,3, or the player lands at position X1,4 or X1,3 or X2,2 after 3 rolls, then the player has lost
* X1,2,k  X2,2,k: True when the player lands on position X1,2 (the tail of the snake) at roll k. The player is automatically moved to position X2,2 if the player lands on position X1,2
* X2,3,k X1,3,k: True when the player lands on position X2,3 (the bottom of the ladder) at roll k. The player is automatically moved to position X1,3 if the player lands on position X2,3
* Xi,j,k Ry X(i)(j+(-1)i\*y)(k+1) except X2,2 R3  X1,4,(k+1) and X2,4 Ry  X(i-1), (j-y), (k+1) : If the player rolls y at any position on the board with the exception of positions X2,2 and X2,4, use X(i)(j+(-1)i\*y) to determine the players landing position and increment the number of rolls, k. This formula shifts the player to the right when i = 2 and shifts the player to the left when i = 1. For the case where the player is at position X2,2 and rolls a 3, the player lands at position X1,4 and the number of rolls, k, is incremented. For the case where the player is at position X2,4 and rolls y, use X(i-1,j-y) to determine the players landing position and the number of rolls, k, is incremented. This formula considers the vertical shift and the leftward shift when i goes from i = 2 to i = 1

# Constraints

* Model describes a single winning position where k is less than or equal to 3, the maximum number of rolls
  + X1,1,k W
* Model limits the number of rolls to 3
  + Xi,j,k where k cannot be greater than 3
* Model constrains the dice to be 3-sided, with number 1, 2 and 3 on one of the 3 sides
  + Ry for y = 1, 2, 3
* Player starts at position X2,1
  + X2,1 always starts off true
* Player can only occupy one position at a time
  + Every time a player leaves a position, that position becomes false

# Model Exploration

*List all the ways that you have explored your model – not only the final version, but intermediate versions as well. See (C3) in the project description for ideas.*

Jape Sequents

1. Rolling a 2 will never result in a loss. (R2 implies not a loss). To simplify, only tiles on the second row (y = 2) are being considered as a roll of 2 will not be enough to win or lose from any tile on the first row. Also, tiles (x=1, y =2) and (x=2, y=2) are impossible as they represent the winning tile and the beginning of a snake, respectively; therefore, only two tiles need to be considered and this is represented in the propositions by .

**Propositions:**

(being on x3 and rolling a 2 implies a win)

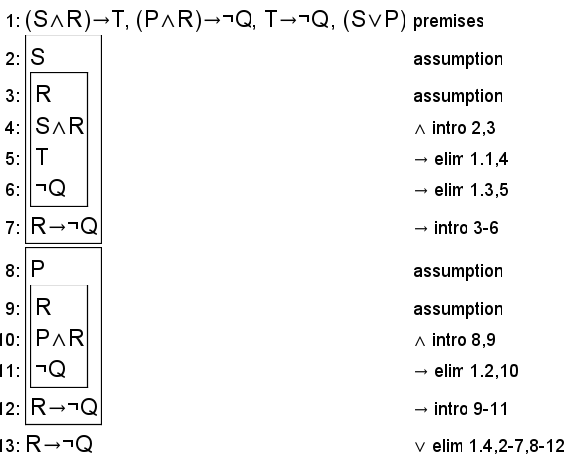
(being on x4 and rolling a 2 implies you haven’t lost)

(winning implies you have not lost)

(current position is either x3 or x4)

**Conclusion:**

(rolling a 2 means you have not lost)



**S represents x3**

**P represents x4**

**R represents r2**

**T represents W**

**Q represents L**

1. Rolling a 1 will never result in a win or a loss. The same simplifications have been made as the first proof. *(Note – this is very similar to the first one and one of the two will likely be changed for final submission)*

**Propositions:**

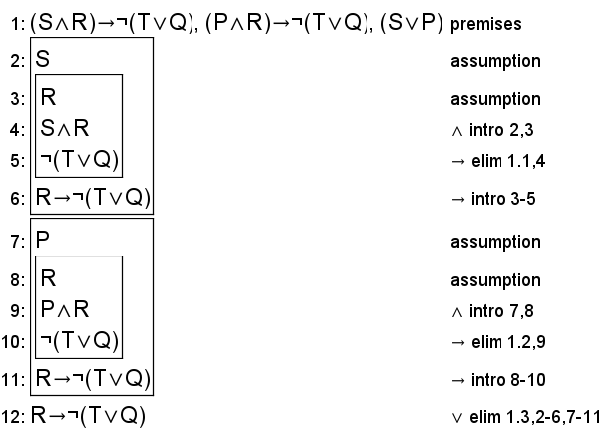
(being on x3 and rolling a 1 implies you haven’t won or lost)

(being on x4 and rolling a 1 implies you haven’t won or lost)

(current position is either x3 or x4)

**Conclusion:**

(rolling a 1 means you have not won or lost)



**S represents x3**

**P represents x4**

**R represents r1**

**T represents W**

**Q represents L**

1. (Incomplete) You can never return to the starting square. (When it is not turn 0, player is not on position (1, 1))

**Propositions:** (When it is not turn 0, it is either turn 1, 2, or 3)

(Three possibilities for locations after turn 1)

(6 possibilities)

(6 possibilities)

(Can only be on one x value at a time, not sure how to XOR Jape-friendly with 4 variables)

(Jape friendly XOR relationship for 2 variables split into two propositions)

**Conclusion:**

(When it is not turn 0 (at least one dice roll has occured), player is not on position (1, 1))

# First-Order Extension

*Describe how you might extend your model to a predicate logic setting, including how both the propositions and constraints would be updated.* ***There is no need to implement this extension!***

# Requested Feedback

1. Is there a better way to simplify/represent the propositions for the third conjecture that will not require 10 distinct variables (x1, x2, x4, x4, y1, y2, T0, T1, T2, T3) as Jape can’t handle this many? Also, what is the best way to represent an XOR relationship in Jape?
2. Our current solution is not satisfiable, do you see any logical errors in the constraints within our code?
3. General insights into how to run the operations of the game, meaning how do we track changes in the game state over time in a meaningful way (do we need separate variables for each time step or should we use a more complex data structure?).