Stevens Institute of Technology

SYS-611 Homework #1 Solutions

1.1 Systems Modeling and Simulation [5 points]

This question is adapted from A.M. Law and W.D. Kelton, *Simulation Modeling and Analysis*, 3rd Edition, McGraw-Hill, 2000, p. 99.

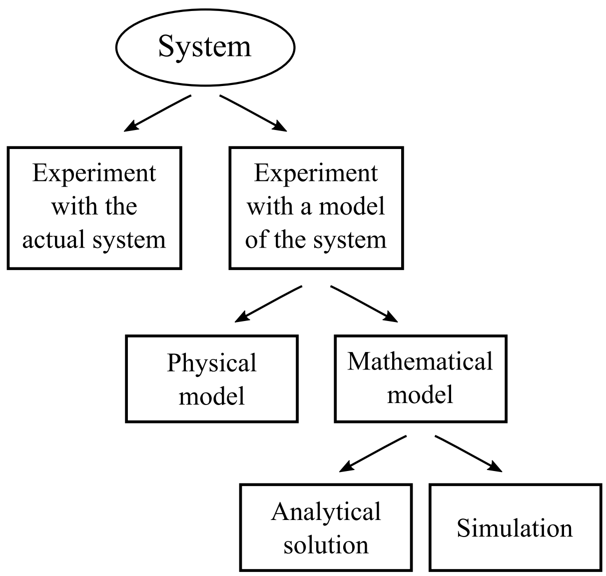


Figure 1.1: Ways to study a system (from Law and Kelton, 2000, p. 4).

(a) Figure 1.1 shows several approaches to study a system. For each of the following problems, suggest a potential solution and argue for the most effective way to study it.

(i) The kitchen line at Mamoun's Falafel restaurant cannot keep up with lunchtime orders and customers are experiencing long delays. It is most effective to experiment with the real system because it is inexpensive to try different process structures, unlikely to cause severe problems with ongoing business, and reversible. However, if one possible solution looks at hiring a new employee, it may be more cost effective to first perform a study using an analytical model (under idealized conditions) or a simulation model (under more realistic conditions)

(ii) The lobby at the Metropolitan Museum of Art often has over-crowding issues and must improve customer access to the ticket counters. It may be most effective to experiment with a physical model because it is inexpensive to create, yet effective to communicate the architecture of a new lobby layout. However, if more sophisticated analysis is required, a simulation model of customer motion and crowding behavior may be useful.

(iii) The NJ Transit bus line 126 between Hoboken and New York is experiencing excessive delays due to traffic at the Lincoln tunnel during evening rush hours. It is most effective to experiment with a simulation model because it is not possible to frequently change a bus schedule and there may be complex contextual factors which contribute to the delays.

(b) Choose **one** of the systems above and imagine a simulation study will be conducted. Should it be static or dynamic and deterministic or stochastic? Discuss why.

There are many answers, consider the following:

(i) A dynamic and stochastic simulation could be used to understand how long it takes to complete the sub-tasks for preparing a sandwich. Uncertainty captures variation in task completion time which may contribute to process bottlenecks.

(ii) A static simulation could be used to evaluate relevant factors related to the lobby appearance which do not generally change over time or are designed for a particular worst-case scenario (e.g. building capacity). For example, a simulation may allow someone to virtually walk through a proposed lobby scene. It is deterministic because there are no sources of uncertainty to model.

(iii) A dynamic and stochastic simulation could be used to understand the effect of passengers getting on and off the bus at different stops during the route or the effect of traffic conditions on maintaining schedule. Uncertainty incorporates factors causing the problem (e.g. crowding on a late bus which only becomes later).

1.2 Interacting with Model State [10 points]

This question builds on the Tic-Tac-Toe model (`ticTacToe.py`) developed in class.

(a) Describe how to check if a Tic-Tac-Toe model state (i.e. `state`) contains a winner. A Tic-Tac-Toe model state contains a winner if there are three "x" or "o" characters in a single row, column, or diagonal.

(b) Describe how to check if a Tic-Tac-Toe model state (i.e. `state`) contains a tie. A Tic-Tac-Toe model state contains a tie if there is no winner and no blank cells (there are other possible definitions).

(c) Implement the Tic-Tac-Toe `get_winner()` function: inspect the model state and return the string `"x"` or `"o"` if there is a winner, otherwise return the special keyword `None`.

A simple "brute-force" solution can be found below.

```
def get_winner():
  if (state[0][0]==state[0][1]==state[0][2]=="x" or
      state[1][0]==state[1][1]==state[1][2]=="x" or
      state[2][0]==state[2][1]==state[2][2]=="x" or
      state[0][0]==state[1][0]==state[2][0]=="x" or
      state[0][0]==state[1][0]==state[2][0]=="x" or
```

```
      state[0][1]==state[1][1]==state[2][1]=="x" or
      state[0][2]==state[1][2]==state[2][2]=="x" or
      state[0][0]==state[1][1]==state[2][2]=="x" or
      state[2][0]==state[1][1]==state[0][2]=="x")
    return "x"
  if (state[0][0]==state[0][1]==state[0][2]=="o" or
      state[1][0]==state[1][1]==state[1][2]=="o" or
      state[2][0]==state[2][1]==state[2][2]=="o" or
      state[0][0]==state[1][0]==state[2][0]=="o" or
      state[0][0]==state[1][0]==state[2][0]=="o" or
      state[0][1]==state[1][1]==state[2][1]=="o" or
      state[0][2]==state[1][2]==state[2][2]=="o" or
      state[0][0]==state[1][1]==state[2][2]=="o" or
      state[2][0]==state[1][1]==state[0][2]=="o")
    return "o"
  return None
```

(d) Implement the Tic-Tac-Toe `is_tie()` function: inspect the model state and return a boolean `True` or `False` depending on whether there is a tie.

A simple "brute-force" solution can be found below.

```
def is_tie():
  if is_winner():
    return False
  if (state[0][0]==" " or state[0][1]==" " or state[0][2]==" " or
      state[1][0]==" " or state[1][1]==" " or state[1][2]==" " or
      state[2][0]==" " or state[2][1]==" " or state[2][2]==" "):
    return False
  return True
```

(e) How could you verify the correct implementation of these two functions? The correct implementation of these two functions could be verified with a test suite to check several key cases. For example, to test the `get_winner()` function, a test suite would correctly identify a winner for each of the 8 winning combinations, correctly identify a winner in several other cases (partially-filled board, and completely-filled board, "x" winner, "o" winner), correctly identify no winner in several cases (empty board, partially-filled board, and completely-filled board). To test the `is_tie()` function, a test suite would correctly identify a tie in a completely-filled board and correctly identify no tie in several cases (empty board, partially-filled board, board with a winner, etc.).

1.3 Modeling a Game as a Simulation [10 points]

Connect Four is a two-player board game produced by Milton Bradley described as[1]:

> a two-player connection game in which the players first choose a color and then take turns dropping colored discs from the top into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the next available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs.

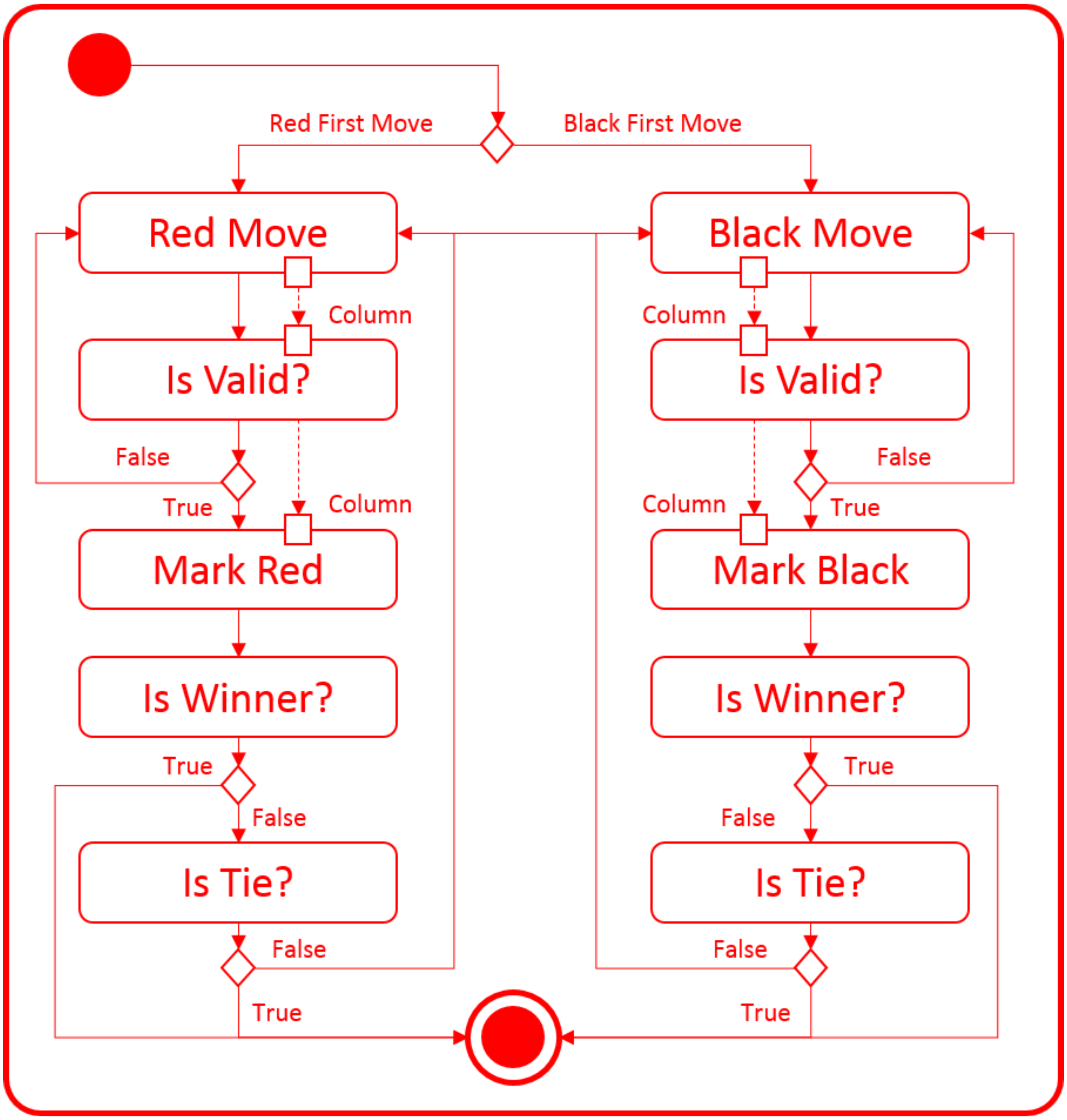
(a) What is the elementary *Connect Four* model state? Describe it in sufficient detail to allow someone to create a mathematical (symbolic) model. (*hint: think of what information is necessary to "save" a game and later recreate it*) The elementary state can be represented as a matrix or list of lists identifying the color of tokens in each cell. There are 42 cells (7 columns by 6 rows) and three possibilities each: red, black, or blank which could be represented as the characters `"x"`, `"o"`, and `" "`, for example. The other piece of elementary state is to know which color (`"x"` or `"o"`) goes first or, equivalently, which color has the next move.

(b) Identify at least two derived states which may be useful for playing *Connect Four* and describe how they could be calculated based on the elementary state in (a). Two pieces of important derived state include whether a game state has a winner (i.e. if there are four tokens of the same color in a row, column, or diagonal) and whether a game state is a tie (i.e. if there are no winners and the board has no blank spaces remaining). Other relevant derived states include how many of each token are on the board, how many turns have passed, and how many remaining turns there are in the game.

(c) What behaviors or state changes are possible in *Connect Four*? For each, describe any arguments or other inputs required and describe how it changes the model state. The primary behavior is to add a token to the game state. The two inputs are: 1) the color of the token and 2) the column it should be dropped. If it is a valid move, the token gets added to the last row in that column with a blank cell.

(d) Draw an activity diagram to show the natural progression of a game from start to end. See the figure on the next page.

[1] `https://en.wikipedia.org/wiki/Connect_Four`


Red First Move
Black First Move
Red Move
Black Move
Column
Column
Is Valid?
Is Valid?
False
False
True
True
Column
Column
Mark Red
Mark Black
Is Winner?
Is Winner?
True
True
False
False
Is Tie?
Is Tie?
False
False
True
True