J.C. Helm, Kallen Marcavage
2020-03-10
CS 4710
Group Project #2

CS4710: Model-Driven Software Development, Spring 2020
Computer Science Department
Michigan Technological University
Group Project #2: Play By Your Own Rules
Due Date: March 6, 2020 at 9:30pm

Problem description:

A casino has invented a new game called Play By Your Own Rules (PBYOR). This game is different from any other game in that there is no single winner; instead, everybody has to win, or else everybody loses. At least 4 people should participate in this game. Let K be the number of participants, where $K > 3$. Here are more details about the game:

- The participants sit around a round table.
- Each person is handed a deck of 4 cards, where each card has a unique number from 0 to 3 written on it. That is, there are no duplicate values in each hand.
- This table has been designed in such a way that each person can only see the top cards in the hands of its left and right neighbors. In other words, each participant is allowed to show its top value to its left and right neighbors and can see their values as well.
- Initially, the casino owner randomly determines what the values of the top cards could be. The owner may even interrupt the game for a finite number of times and shuffle all cards. When he stops shuffling, the players can continue making moves in the game.
- Each participant reads the values of its immediate left and right neighbors and decides as to whether or not it should show a new top value between 0 and 3. If a participant decides to change its value, it does so atomically. That is, a participant can atomically read the left and right values and pick a new value as its top card.
- Note that everybody must use the same rules of decision making for selecting a new top value.

The constraints imposed by the casino owner are as follows:

- **Termination:** The rules of the game should be designed in such a way that the game will eventually reach a state where no one can make a move; i.e., termination. That is, the rule(s) each participant uses is(are) disabled.
- **Symmetry:** Everyone should follow the same rule(s) for determining what his/her next top value should be. Two players have symmetric rules/actions if the actions of one can be obtained from the other by a simple variable re-naming.
- **Winning conditions:** Everybody wins if the participants can design the rules in such a way that after a finite number of rounds of decision making by participants, the game terminates and everybody has the same value on their top card; otherwise, everybody loses.
- **<u>No collusion:</u>** The players cannot collude on a specific value before the game starts; nor can they collude on a value during the game. For example, they cannot design a rule where they show a specific value c $\in$ {0, 1, 2, 3} as their top card regardless of the top cards of their neighbors.

Extended Constraints, Via "Tips on Project 2":

- You cannot have a global flag that is read by players or the owner to detect the start/end of the game; players can read only their own top card and their neighbor?s.
- The owner cannot create, run or terminate players. The only thing it can do is to change the values of top cards randomly for a finite number of times. For example, player proctypes must not be started from within the owner proctype.
- Notice that all proctypes, the players and the owner, should run concurrently. There cannot be any implicit or explicit synchronization mechanism between the owner and players.

Objective:
Before the participants start to play, they should collectively design the rule(s) that each participant should follow so the game eventually terminates and everybody wins. Please help them by designing the rule(s) of the game.

Deliverables:

- Design the rule(s) that each participant should follow so the game eventually terminates and everyone wins no matter what the initial values of the cards are. Each rule should be executed atomically. (50 points)
- Create a Promela model for a version of the PBYOR, where K=5 and each proctype implements the rule(s) that you designed in Step 1. The proctypes should be symmetric because they implement the same rule(s). Two proctypes are symmetric if the code of one can be obtained from the other by a simple variable re-naming. (15 points)
- Design and implement an Owner proctype that implements the behavior of the owner. Recall that, the owner can randomly perturb the values of the top cards of all participants for a finite number of times. (15 points)
- Use SPIN to verify that no matter how the casino owner shuffles the cards, the participants eventually win this game played by the rule(s) you designed in Step 1. (20 points)

This project was originally due on March 6th at 9:30pm.
As of March 5th, the assignment was extended to March 10th at midnight.

Deliverables (and see attached pml file)

- The Rules:

    1. If both the cards to the left and the right are the same as your card, hold and do nothing.
    2. If both cards to your left and right are less than yours, swap to your next card as it's likely you're almost in a hold condition.
    3. If both cards to your left and right are greater than yours, swap to your next card as it's likely you're almost in a hold condition.
    4. If the card to your right is less than yours, but to your left is greater then swap to your next card.
    5. if your card to your right is greater than yours, but the card to left is less than yours, do nothing. This action is more of a stabilizer to ensure that there is not a rapid firing of actions, and there is more than one wait condition.
    6. If there have been (Number of cards + 1) occurrences of holding your cards and doing nothing due to the left and the right staying the same as yours, then that is stated as a win condition. As everyone should have been able to swap through all of their cards by that point in time.

- Symmetric Proctypes:

    1. All player processes are created from one proctype, with the only variable being the ID of the player.

- Owner Proctype

    1. The owner proctype performs a do loop that only ends when the owner has performed a finite number of shuffles, determined by the random value of a byte.
    2. In the do loop, the owner evaluates a random number between 1 and the ratio value set. Due to state space constraints this value is currently set at 2, and to refactor this would require more variables and minor modifications. Currently this evaluation is similar to a coin flip.
    3. If the owner wins the above coin flip, he enters an atomic state and shuffles every player's deck of cards.

- Verification with SPIN

1. Due to the amount of potential randomness that is available in a game such as this, verification of this model is nearly impossible with the resources we have available. Each shuffle, and the many iterations of potential shuffle outcomes, great and exponentially large amount of states that the program could be in. With this in mind we were unable to perform the verification as we were unable to have enough resources to check all states. The maximum resources attempted were 24 CPU cores and 256GB of memory of the guardian.it.mtu.edu server.

Other Notes:
Due to the features of Promela used, SPIN version 6.5 or higher is required. Running this model on the campus Linux machines results in an error as they are on SPIN version 6.4.8.