Abstract

We are exploring the popular board game 'The Resistance' through a formal logic lens, where the turn-based format with 2 teams (Spies, Resistance) and hidden player identities allow the establishment of propositional models that are solvable by computer. The 7 players go on a series of 'missions' which are voted to go ahead or not by a subset of all players selected by a leader in each of the 5 rounds. A mission succeeds for the Resistance team and gains them a point if all participants submit a 'success' card in an anonymous choice procedure, which is the key game mechanic as a given player doesn't directly know the team membership of other players and is forced to infer this information as the rounds progress. Spies can submit a 'fail' card and cause the mission to fail which gives them one point, and either team wins the game with 3 points.

Propositions

x_i is true at position i depending on the round number (e.g. x_2 is true for r2)
Z_j is true if the vote tracker is at position j
Y_k is true if mission k is approved
S is true if the current mission is a success for team Resistance
W_l is true depending on the position of the round l won for team R
V represents the overall game victory condition

Constraints

Detailed description of the game rules:

7 players total $\rightarrow$ 4 R, 3 S

5 total rounds ('missions')$\rightarrow$ a team wins if they succeed in 3/5 in any order

Round structure:
r1: 3 players
r2: 4 players
r3: 3 players
r4: 4 players
r5: 4 players

Before the game starts (i.e. r0) each player is randomly assigned a team which none of the other players know. This assignment lasts until the game ends. At the start of each round, one player is assigned the role of 'mission leader' & the remaining 6 are given 1 accept and 1 reject token for voting purposes. The leader then gets to choose the corresponding number of players per round from the pool to send on the mission as above (leader not included).

Mission start:

Selected players vote to either accept or reject the current mission.
If a majority reject the mission, then the next counterclockwise player is assigned leader. NOTE: If 5 rejections occur in a row then the S team

wins automatically. The vote tracker resets every round (e.g. r2 can have 1 rejection, r3 0, r4 3, etc.)
If a majority accept the mission, then each active player is given 1 success card and 1 failure card. NOTE: R players have to play success every time, whereas S players can play either success or failure. Once all players turn in their cards then the total number is counted. If there's $\geq$1 fail cards then the current mission fails $\rightarrow$ 1 point for team S. Otherwise, 1 point for team R

If there's a tie of accept/reject tokens (possible on rounds w/ an even number of players) then flip a coin to see whether the mission happens or not (H yes, T no).

Mission end

At the end of any round, if a team gets 3 points then they are declared the winner, unless the above win condition for team S is satisfied.

As per Prof. Muise's suggestion, we've expanded the scope of the constraints by imagining that at the time of player selection for each mission the leader will have access to a computer with the Python scripts for this project, and they will attempt to calculate an estimate of winning based on the probability that the other players are on team R or team S. This is done by simply counting the number of valid models (propositional formulas evaluating to true under the selected conditions) and dividing by the total number of models:

$$P(win) = \frac{n_{valid}}{n_{valid} + n_{invalid}} \tag{0.1}$$

Model Exploration

The PDB debugger that comes included by default python3 installation (Ubuntu 20.04) was used to explore the run.py script as it ran, with breakpoints set at function calls. This confirmed that the basic version works and allowed for a better understanding of how the lists containing the models are modified depending on certain constraints being met or not. Importantly, use of the random package is projected to make generating various win/loss scenarios simpler. Please see the uploaded screenshots in the /draft folder for a detailed look

Requested Feedback

1. Do the listed propositions & constraints properly match the game rules? If not, what should be changed?
2. How well do the Python scripts simulate the mechanics of the game?
3. Are the Jape proofs complete from a logical perspective? Why or why not?

Thank you for taking the time to do this!