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_r is true at position r depending on the round number (e.g. X_2 is true for r2)
Z_s is true if current mission is rejected s times (e.g. Z_3 means the 3rd consecutive rejection)
Note that r and s both range from 1 to 5, but X_r and Z_s don't necessarily have the same truth value when $r = s$ due to how the game functions
Y_k is true if mission k is approved, false if not (e.g. $\neg Y_4$ means mission 4 not approved)
P_i is true if player i has voted to approve the current mission and false otherwise, where i ranges from 1 to 6
Q_j is true if player j has played a success token in the currently approved mission, false if player j has played a fail token (e.g. $\neg Q_1$ means player 1 has played a fail token)
Note that i ranges from either 1 to 3 or 1 to 4 depending on the number of players per round
S_r is true if the current mission is a success for team Resistance and false otherwise, and vice-versa for team Spies
V represents the overall game victory condition: true for the Resistance winning the game, false for the Spies emerging victorious

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. Moose'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:

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

We proceed through an example game with specific votes & tokens to illustrate how the constraints are expressed in propositional logic.

Round 1:

$X_1 \wedge (\neg P_1 \wedge \neg P_2 \wedge P_3 \wedge \neg P_4 \wedge \neg P_5 \wedge P_6) \rightarrow \neg Y_1 \wedge Z_1$

Here 4/6 players voted to reject the mission so Y_1, Z_1 are updated.

$X_1 \land (P_1 \land \neg P_2 \land \neg P_3 \land \neg P_4 \land \neg P_5 \land \neg P_6) \to \neg Y_1 \land Z_1 \land Z_2$

Here 5/6 players voted to reject the mission, with Z_2 added accordingly.
...
Voting rounds 3 and 4 also result in rejections.
...

$X_1 \land (\neg P_1 \land \neg P_2 \land \neg P_3 \land \neg P_4 \land \neg P_5 \land \neg P_6) \to \neg Y_1 \land Z_1 \land Z_2 \land Z_3 \land Z_4 \land Z_5$
$Z_5 \to \neg V$

After 5 rejections, the Spies automatically win the game.

Alternatively, let's say the 1st mission is approved:
$X_1 \land (P_1 \land \neg P_2 \land P_3 \land P_4 \land \neg P_5 \land P_6) \to Y_1$

Only 2/6 players voted against so now the mission can go ahead.

$Y_1 \land (Q_1 \land Q_2 \land Q_3) \to S_1$

In this scenario round 1 counts as a success for the Resistance as none of the players put forward a fail token.

Round 2:
$X_2 \land (P_1 \land P_2 \land P_3 \land \neg P_4 \land P_5 \land P_6) \to Y_2$

$Y_2 \land (\neg Q_1 \land Q_2 \land Q_3 \land \neg Q_4) \to \neg S_1$

Here the mission is approved right away, and Spies win the round by playing a single failure token.

Round 3:
$X_3 \land (\neg P_1 \land \neg P_2 \land P_3 \land \neg P_4 \land P_5 \land \neg P_6) \to \neg Y_3 \land Z_1$

$X_3 \land (P_1 \land P_2 \land P_3 \land \neg P_4 \land P_5 \land P_6) \to Y_3 \land Z_1 \land \neg Z_2$

Here the mission is approved on the 2nd voting round; notice how the Z_s counter resets to 1 at every round.

$Y_3 \land (\neg Q_1 \land Q_2 \land Q_3) \to S_3$

The Q_j's and P_i's are also different per round. Here the Resistance wins mission 3.

...
Round 4 is a win for team Spies
...

Round 5:

$X_5 \wedge (P_1 \wedge P_2 \wedge P_3 \wedge P_4 \wedge P_5 \wedge P_6) \rightarrow Y_5$

$Y_5 \wedge (Q_1 \wedge Q_2 \wedge Q_3 \wedge Q_4) \rightarrow \neg S_5$

The final tally is then:

$S_1 \wedge \neg S_2 \wedge S_3 \wedge \neg S_4 \wedge S_5 \rightarrow V$

RESISTANCE WINS - YAHOO!

Alternatively, let's assume that the Spies win in rounds 1, 2, and 4:

$\neg S_1 \wedge \neg S_2 \wedge S_3 \wedge \neg S_4 \rightarrow \neg V$

In this case the Spies automatically win after round 4 finishes as they have the required 3/5 missions. It's also possible for the Resistance to win after having played only 3 rounds, etc.

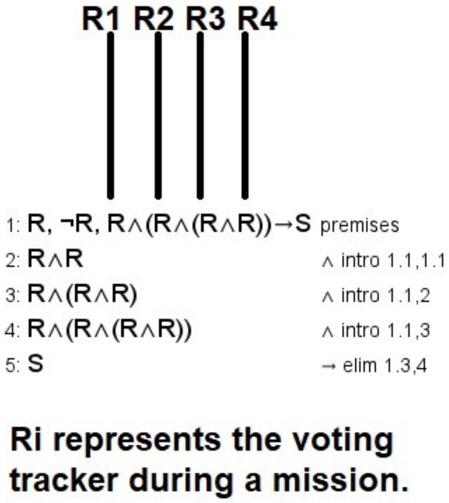
It's also possible to chain together a massive series of disjunctions for all the winning combinations of S_r's for the Resistance:

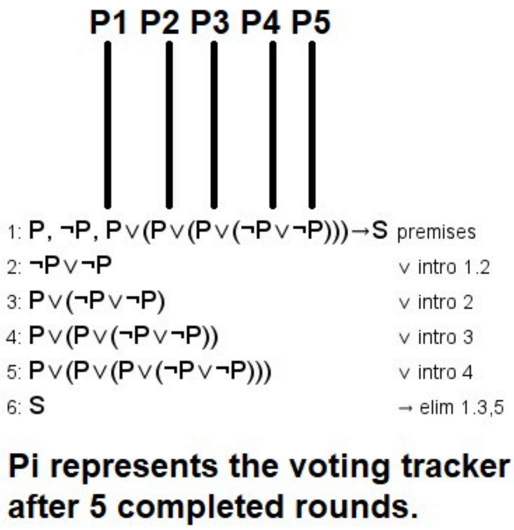
$$(S_1 \wedge S_2 \wedge S_3 \wedge \neg S_4 \wedge S_5) \vee (\neg S_1 \wedge S_2 \wedge S_3 \wedge S_4 \wedge S_5) \vee ... \vee (S_1 \wedge S_2 \wedge \neg S_3 \wedge \neg S_4 \wedge S_5) \rightarrow V \tag{0.2}$$

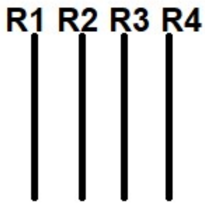
In the same way we can list all the possibilities that lead to a victory for the Spies. Bada-bing!

Jape Proofs

Please see the attached file proofs.jp.
We had trouble assigning subscripts, so the following images are for clarification purposes:

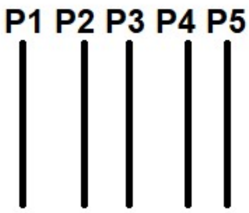






1:	R, ¬R, R∧(R∧(R∧¬R))→¬S	premises
2:	R∧¬R	∧ intro 1.1,1.2
3:	R∧(R∧¬R)	∧ intro 1.1,2
4:	R∧(R∧(R∧¬R))	∧ intro 1.1,3
5:	¬S	→ elim 1.3,4

Ri represents the voting tracker during a mission.



1:	P, ¬P, ¬P∨(¬P∨(P∨(¬P∨P)))→¬S	premises
2:	¬P∨P	∨ intro 1.1
3:	P∨(¬P∨P)	∨ intro 2
4:	¬P∨(P∨(¬P∨P))	∨ intro 3
5:	¬P∨(¬P∨(P∨(¬P∨P)))	∨ intro 4
6:	¬S	→ elim 1.3,5

Pi represents the voting tracker after 5 completed rounds.

1:	Q	premise
2:	¬S	assumption
3:	Q	hyp 1
4:	¬S→Q	→ intro 2-3

Q represents a victory. ~S implies a victory if position P5 on the tracker is not a Spy victory.

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

Further details involving bug fixes and such to be provided

From C3:

1. Example solution analysis
2. Constraint testing
3. Partial assignment testing
4. Other?

First Order Extension

Given the enumerative nature of our constraints in each voting round and mission, expanding the propositions to predicates is straightforward.

For a rejection to occur the conjunctions involving P_j are replaced with $\forall x : P(x)$, which is equivalent to a single vote against represented by $\exists x : \neg P(x)$.

e.g. in round 1:
$X_1 \land \exists x : \neg P(x) \rightarrow \neg Y_1$

The 5 rejections in a row victory condition for the Spies directly follows:
$\forall y : Z(y) \rightarrow \neg V$

After mission approval single $\exists z : \neg Q(z)$ causes the current mission to fail:
$Y_2 \land \exists z : \neg Q(z) \rightarrow \neg S_2$

The final tally for the Resistance win condition is easily represented with existential quantifiers:
$\exists a : \exists b : \exists c : S(a, b, c) \rightarrow V$

$\square$

README!?