Project instructions

We will progressively build a machine that outputs expected payoffs for a finite state Markov chain. Importantly, this is to be done \*symbolically\*m, using SymPy; NOT numerically.

Here's the overview of how we will proceed:

The machine should consist of several key components and routines:

Core Routines:

1. A routine that takes a game input and strategy profile, and outputs a transition matrix P

- This P matrix should handle both on-path and off-path states

- It should capture the two-cycle nature of the strategy profiles being analyzed

2. A deviation analysis routine that takes:

- Game input

- Strategy profile

- Player identity

- Deviation move

And outputs the deviation payoff for that player at the time of deviation

- This includes constructing a "little d" transition vector for the moment of deviation

- It should handle the transition probabilities from the game input for that specific deviation

3. A comprehensive deviation checker that:

- Takes a strategy profile

- Systematically considers all possible single-state deviations

- Computes deviation payoffs for each using the above routine

- Outputs all "don't want to deviate" inequalities

4. A final system that:

- Takes all the inequalities generated above

- Expresses them in terms of the underlying parameters

- Can check if they can be satisfied

- Should work symbolically (though could be evaluated numerically if needed)

Additional Requirements:

1. The machine should consider incentive conditions both on-path and off-path

2. It should be able to handle blunder probabilities from the game input

3. All calculations should be done from player one's perspective

4. The system should be able to handle both symbolic and numerical computations

The end goal is to have a machine that can automatically verify whether a conjectured strategy profile could be an equilibrium by checking all relevant incentive constraints.