Decision Theory II: Open-Source Game Theory, Commitment Races, and Safe Pareto Improvements

Daniel C Satya Benson
Williams College

September 8, 2026

1 Afternoon lecture

Open-source game theory. When players can read each other's source code, cooperation becomes possible without repetition or prior trust. **FairBot** cooperates if and only if it can prove its opponent cooperates with it; two FairBots cooperate by Löb's theorem (from `□(□C -> C) -> □C`). It is unexploitable but brittle (it relies on a proof system and exact source). **Epsilon-grounded bots** (Oesterheld) replace proofs with simulation: cooperate unconditionally with small probability epsilon, else simulate the opponent and copy its move; this terminates almost surely, is a Nash equilibrium, and is robust. The unifying idea is *conditional commitment*: "I commit to X conditional on you committing to Y." Conditional makes it unexploitable; commitment makes it legible. FairBot is exactly this.

Commitment races. Among consequentialists who can commit, there is an incentive to commit *first*: the first mover can pick the point on the bargaining frontier best for itself and leave the responder to best-respond. So "the best response is not the best response", and agents race to commit before they even finish learning, risking incompatible lock-in or wasteful conflict (including threats and s-risks). Moving first "in logical time" is a genuine safety concern, which makes *avoiding* commitment races a safety goal. The deeper obstacle is *entanglement*: each agent optimizes against a distribution over the *counterfactual* programs the other might submit, so punishing a Pareto improvement means forgoing utility against the opponent's *actual* program; the incentive to make improvements is entangled with those counterfactuals, and this persists even with full conditional commitment.

Safe Pareto improvements (SPI). An SPI modifies the agents' default (conflict-prone) strategies so that *every* player is guaranteed at least as well off regardless of the opponent, a guaranteed weak Pareto improvement (Oesterheld-Conitzer; DiGiovanni-Clifton-Macé). In the open-source/program setting, each player individually prefers to use renegotiation-based SPIs, and the guaranteed floor is the *Pareto Meet Minimum* (your lowest efficient payoff), which is tight under a *participation-independence* assumption. That assumption can fail (agents may become hawkish when conflict is cheap), the "cheating" problem. *Entanglement-free SPI* (a research-frontier idea) restructures

the interaction into two stages (exchange renegotiation programs, update on the actual one, then submit defaults), which breaks the entanglement and drops participation independence, at the cost of the tight floor. Teach the standard SPI result in full; present entanglement-free SPI briefly as the current frontier.

2 The tournament

Full rules are in the tournament handout; the structure is summarized here so the day can be run from this document.

The game. The one-shot Prisoner's Dilemma, with the twist that your program can read the opponent's program before deciding (source is open). Payoffs (you, them): mutual cooperation `(C,C)` gives 3 each; mutual defection `(D,D)` gives 1 each; defecting against a cooperator gives 5 to the defector and 0 to the cooperated. This satisfies `T` $>$ `R` $>$ `P` $>$ `S` ($5 > 3 > 1 > 0$) and `2R` $>$ `T + S` ($6 > 5$).

Two leagues.

- *League A (one-shot, open source).* Bots see the opponent's source and decide once. This is the program-equilibrium / Löbian-cooperation setting.
- *League B (iterated, history only).* Bots see the move history over a hidden number of rounds (around 100-200) and cannot read source. This is the classic reciprocity setting (Tit-for-Tat and relatives).

Submitting a bot. A submission may be in any form (real Python, pseudocode, or a careful English description) plus a short design rationale explaining the decision-theoretic approach; an LLM compiles it into the canonical `Agent` class:

```
class Agent:
    def move_oneshot(self, opp) -> Move:                    # League A
    def move_iterated(self, my_hist, opp_hist, t) -> Move: # League B
```

In League A a bot may query: `opp.move_against(SELF)` (the opponent's move against you), `opp.move_against(DEFECT_BOT)`, `opp.move_against(COOP_BOT)`, and `opp.sour` (the opponent's canonical source). League B provides only the two history lists and the round index `t`.

How circular queries resolve. Two source-reading bots can each ask "what does the other do against me?", which is circular. Rather than simulate the recursion, the engine treats every "what does the opponent do against X?" query as a *provability* question and computes the fixed point directly, the same Löbian move that lets two FairBots cooperate. When no stable resolution exists, it applies a *no-wishful-thinking* rule: if cooperation cannot be established, default to D.

Reference bots. League A: CooperateBot, DefectBot, FairBot, PrudentBot, CliqueBot, RandomBot. League B: AllC, AllD, Random, TitForTat, GrimTrigger, GenerousTitForTat, Pavlov, TitForTwoTats. These seed the field and give students targets to beat or cooperate with.

Scoring. Round-robin: every bot plays every other bot, including a copy of itself, and ranking is by total points accumulated across all matchups (not head-to-head wins). League B matches add about 2% move-flip noise and a randomized hidden length, so brittle strategies are penalized.

What good looks like. A strong submission embodies a clear decision-theoretic concept (commitment, transparency, the limits of self-reference, functional decision theory) and explains it, rather than chasing the leaderboard. The standout lesson is usually that FairBot-style conditional cooperation beats both naive cooperation (exploited) and naive defection (misses mutual cooperation) in League A, while robustness to noise dominates in League B.

3 Learn more

Multi-agent and open-source game theory. The annotated program-equilibrium bibliography (Oesterheld); the epsilon-grounded FairBot paper; When would AGIs engage in conflict?; the safe-Pareto-improvement papers (Oesterheld-Conitzer; DiGiovanni-Clifton-Macé). Andrew Critch's work on cooperative and uncooperative institution design is a good, pedagogically simple source for further exercises.

Frontier. Logical updatelessness and logical time; UDT 2 and policy-selection fixes; bargaining and notions of fairness (the Nash bargaining solution and the "zoo" of coalitional structures); entanglement-free safe Pareto improvements; reflective oracles and program equilibrium as a single picture.