CMSC 474, Game Theory

6a. Repeated Games

Dana Nau

University of Maryland

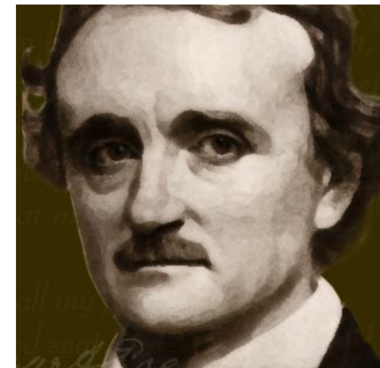
Repeated Games

- Repeatedly play the same game against the same opponent


Prisoner’s Dilemma

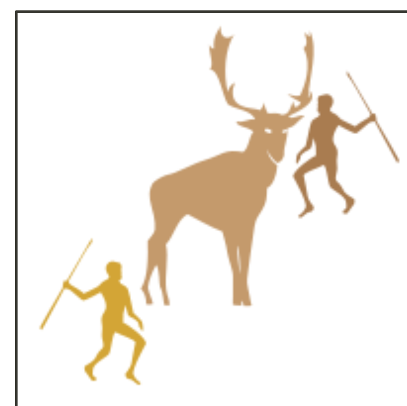

Battle of the Sexes


Rock, paper, scissors


Matching Pennies


Chicken Game


Ultimatum Game


Stag Hunt

Finitely Repeated Games

- Some game G is played multiple times by the same set of agents
  - G is called the **stage game**
    - Usually (but not always) a normal-form game
  - Each occurrence of G is called an **iteration**, **round**, or **stage**
- Usually each agent knows what all the agents did in the previous iterations, but not what they're doing in the current iteration
  - Thus, *imperfect information* with *perfect recall*
- Usually each agent's payoff function is additive

Prisoner's Dilemma:

	C	D
C	3, 3	0, 5
D	5, 0	1, 1

Iterated Prisoner's Dilemma, 2 iterations:

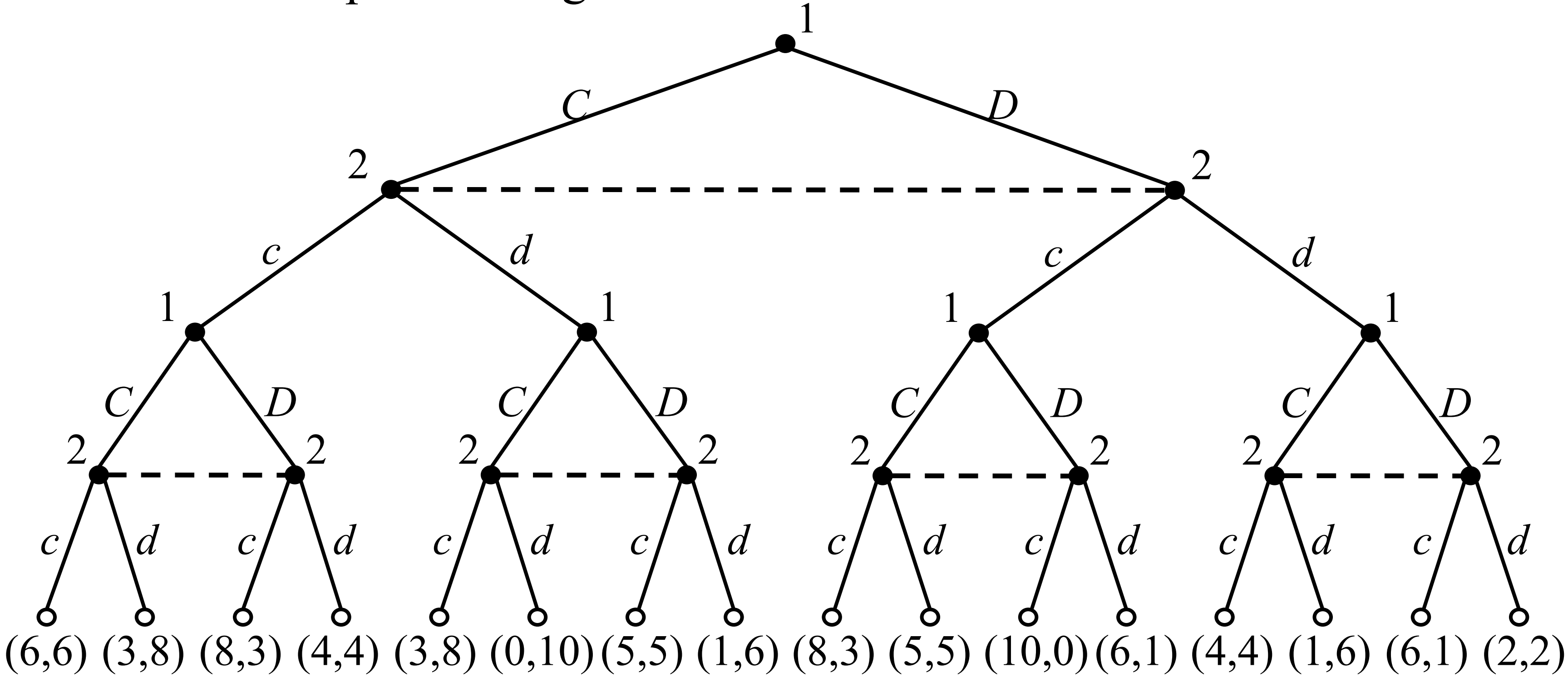

	Agent 1:	*Agent 2*:
Stage 1:	C	C
Stage 2:	D	C
Total payoff:	3+5 = 8	3+0 = 3

Strategies

	c	d
C	3, 3	0, 5
D	5, 0	1, 1

- Much bigger strategy space than the stage game
  - E.g., Iterated Prisoner's Dilemma (IPD)
  - 1 iteration → each player has 1 choice node, 2 pure strategies
  - 2 iterations → each has 1 + 4 choice nodes, 2^{1+4} pure strategies
  - n iterations → each has $1 + 4 + 4^2 + \ldots + 4^{n-1} = (4^n - 1)/3$ choice nodes,

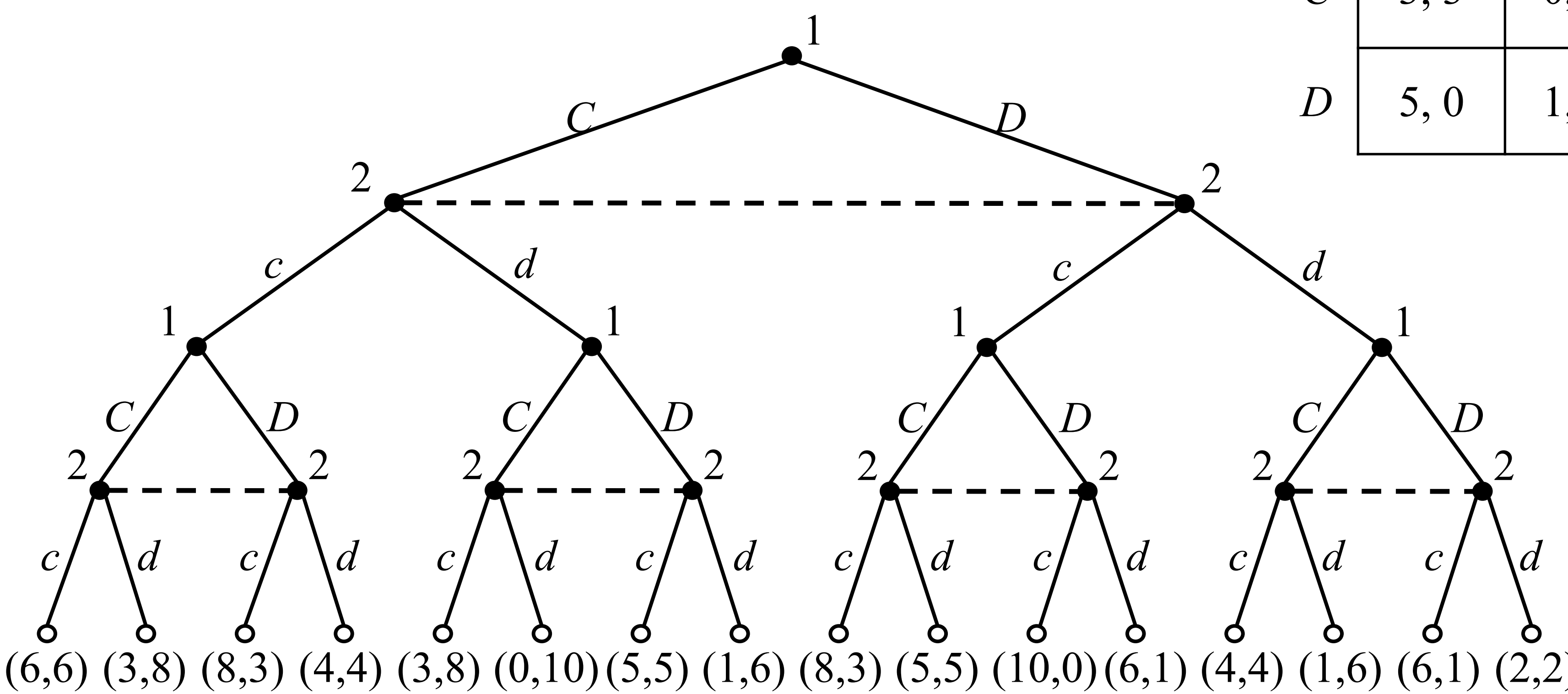
    $2^{(4^n - 1)/3}$ pure strategies



Simple Strategies

- **Stationary strategy**: use the same strategy in every stage game
  - In IPD, only 2 pure stationary strategies
- Slightly more complicated: non-stationary strategy that only depends on the last *k* iterations
  - There are many well-known strategies that use $k \leq 1$

	c	*d*
C	3, 3	0, 5
D	5, 0	1, 1



Examples

Some well-known IPD strategies:

- **AllC**: always cooperate
- **AllD**: always defect
- **Grim**: cooperate until the other agent defects, then defect forever
- **Tit-for-Tat (TFT)**: on 1st move, cooperate. On n^{th} move, repeat the other agent's $(n–1)^{th}$ move
- **Tit-for-Two-Tats (TFTT)**: like TFT, but only only retaliates if the other agent defects twice in a row
- **Tester**: D then C. If opponent retaliates, play C then TFT. Otherwise alternate D and C
- **Pavlov**: in 1st stage, cooperate. Thereafter,
  win => use same action on next stage;
  lose => switch to the other action
  ("win" means 3 or 5 points, "lose" means 0 or 1 point)

AllC, Grim, TFT, or Pavlov	*AllC, Grim, TFT, or Pavlov*
C	C
C	C
C	C
C	C
C	C
⋮	⋮

TFT	*Tester*
C	*D*
D	C
C	C
C	C
C	C
C	C
C	C
⋮	⋮

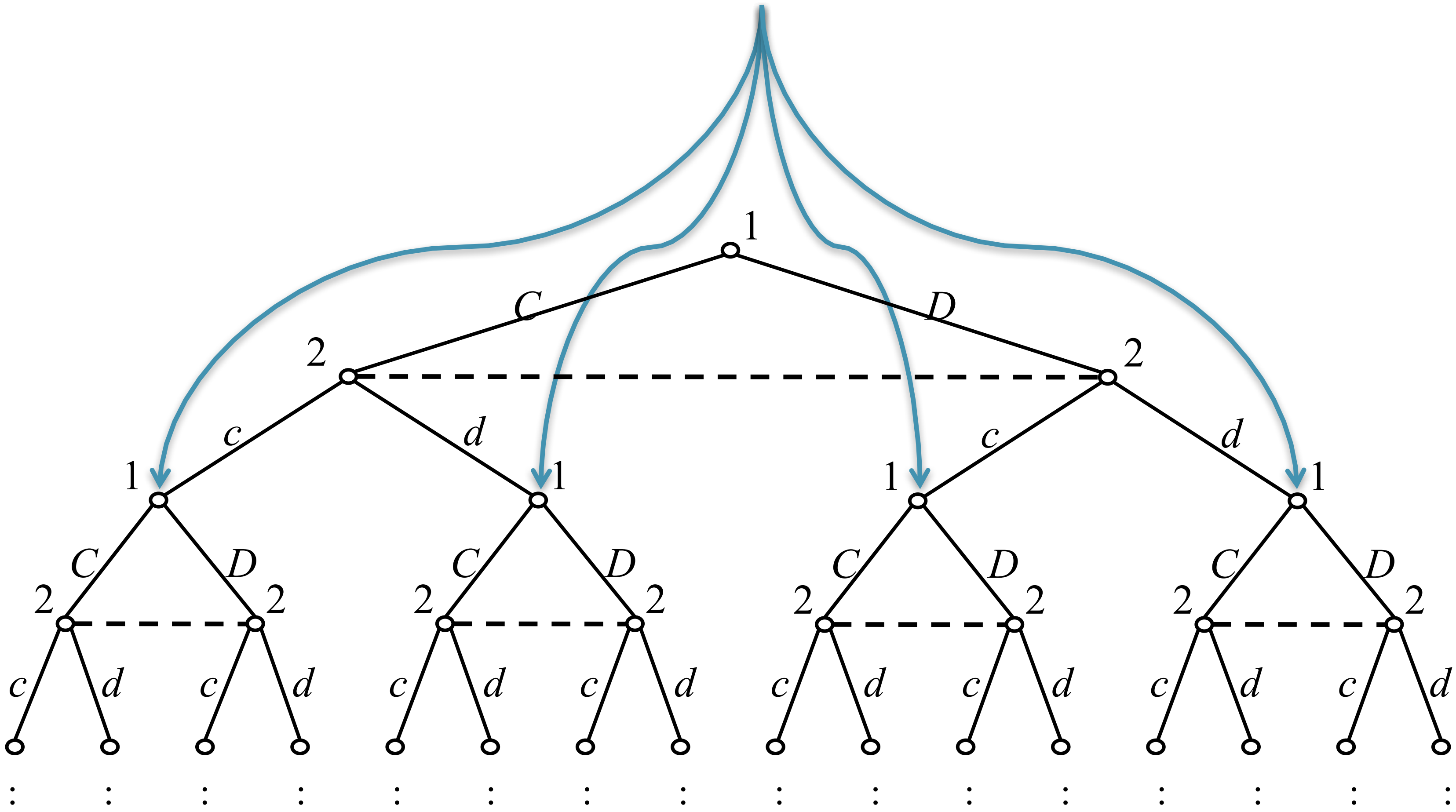
TFTT	*Tester*
C	*D*
C	C
C	*D*
C	C
C	*D*
C	C
C	*D*
⋮	⋮

TFT or Grim	*AllD*
C	*D*
D	*D*
D	*D*
D	*D*
D	*D*
D	*D*
D	*D*
⋮	⋮

Pavlov	*AllD*
C	*D*
D	*D*
C	*D*
D	*D*
C	*D*
D	*D*
C	*D*
⋮	⋮

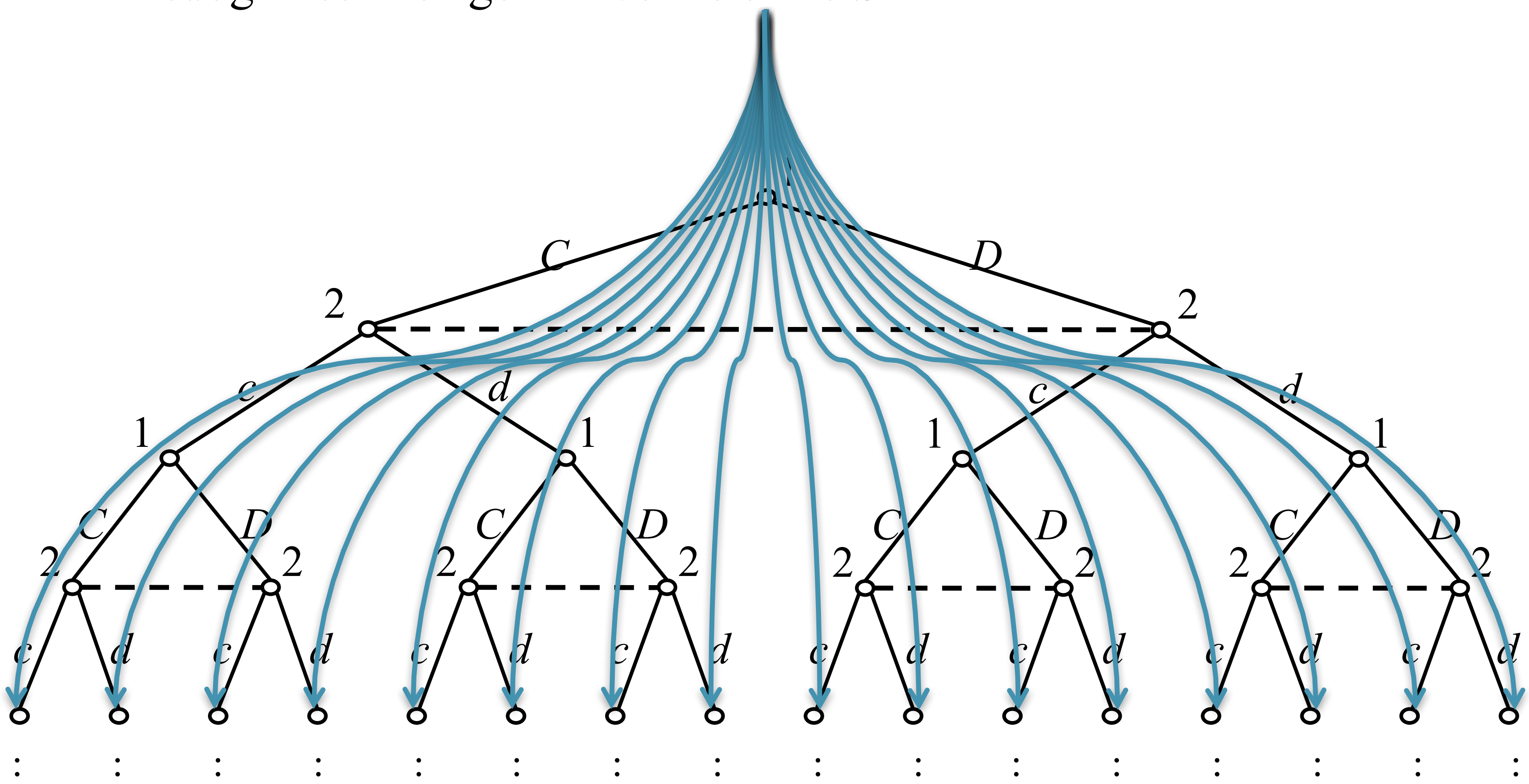
Backward Induction

- n iterations, all players know what n is, rationality is common knowledge
- Use backward induction to find a subgame-perfect equilibrium
- This time it's simpler than game-tree search
  - All subgames at stage 2 have the same SPE



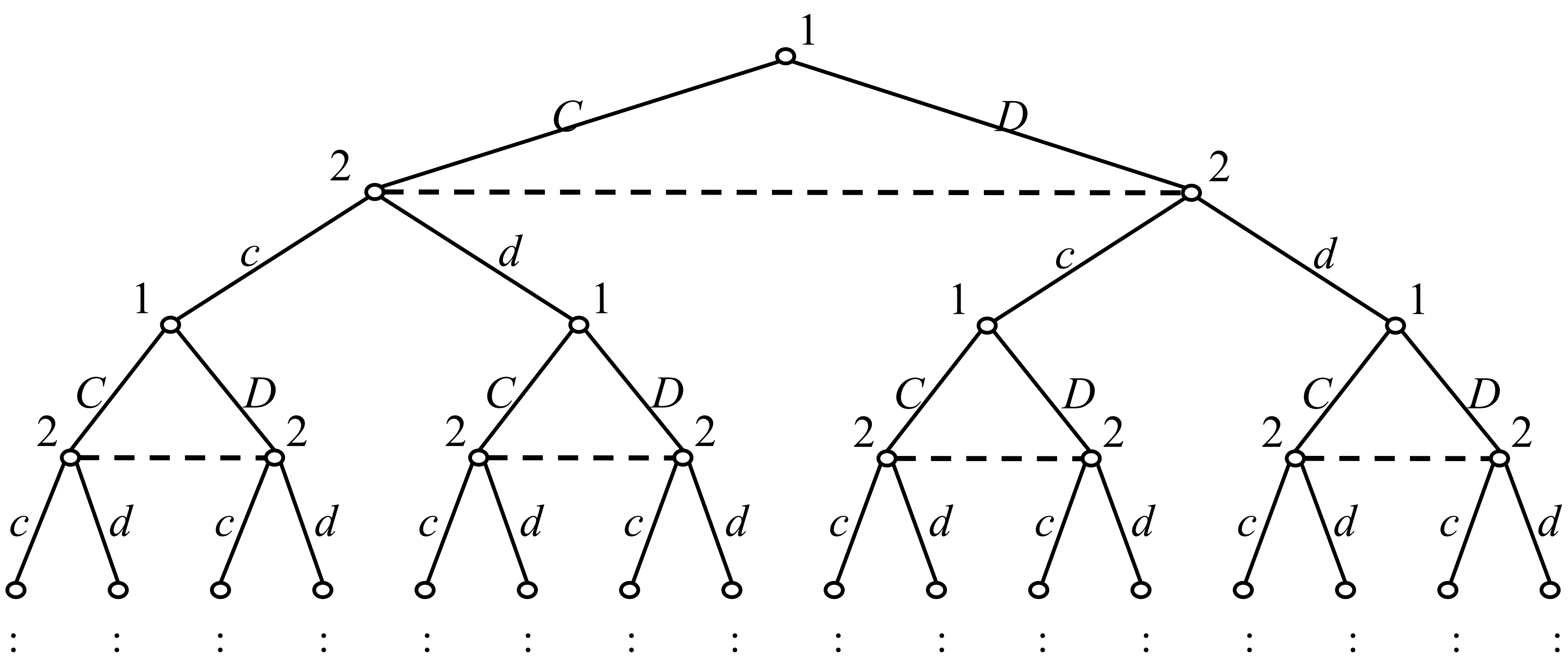
Backward Induction

- *n* iterations, all players know what *n* is, rationality is common knowledge
- Use backward induction to find a subgame-perfect equilibrium
- This time it's simpler than game-tree search
  - All subgames at stage 2 have the same SPE
  - All subgames at stage 3 have the same SPE



Backward Induction

- n iterations, all players know what n is, rationality is common knowledge
- Use backward induction to find a subgame-perfect equilibrium
- This time it's simpler than game-tree search
  - All subgames at stage 2 have the same SPE
  - All subgames at stage 3 have the same SPE
- For j = 1, …, n, all subgames at stage j have the same SPE



Backward Induction

- n iterations, all players know what n is, rationality is common knowledge
- Use backward induction to find a subgame-perfect equilibrium
- This time it's simpler than game-tree search
  - All subgames at stage 2 have the same SPE
  - All subgames at stage 3 have the same SPE
- For $j = 1, \ldots, n$, all subgames at stage j have the same SPE

- First calculate the SPE action profile for stage n (the last iteration)
- For stage $j = n–1, n–2, \ldots, 1$,
  - Common knowledge of rationality → everyone will play their SPE actions after stage j → can calculate each player's cumulative payoff
  - Create payoff matrix showing cumulative payoffs from stage j onward
  - From this, calculate SPE at stage j

Example

- Stage n (last stage): SPE profile is (D,D); each player gets 1
- Stage n–1:
  - Cumulative payoffs = (stage n–1 payoffs) + 1
    - SPE: (D,D) at stages n–1 and n
    - Each player's SPE payoff = 2
- Stage n–2:
  - Cumulative payoffs = (stage n–2 payoffs) + 2
  - SPE: (D,D) stages n–2, n–1, and n
  - Each player's SPE payoff = 3
  - . . .

- SPE: play (D,D) at every stage

n	C	D
C	3, 3	0, 5
D	5, 0	1, 1

n–1	C	D
C	4, 4	1, 6
D	6, 1	2, 2

n–2	C	D
C	5, 5	2, 7
D	7, 2	3, 3

Example

- Limitation
  - If the other players play something other than their SPE strategies, then your SPE strategy isn't your best response

	C	*D*
C	3, 3	0, 5
D	5, 0	1, 1

- **Poll**: Suppose you're playing the IPD with 4 iterations, and the other player's strategy is TFT. Which of the following is a best response?
  - C,C,C,C
  - C,C,C,D
  - C,C,D,D
  - D,C,C,C
  - D,D,D,D

Example

- Limitation
  - If the other players play something other than their SPE strategies, then your SPE strategy isn't your best response
- IPD:
  - Situation somewhat similar to the Centipede game
  - If both players cooperate until near the end, both do better

	C	*D*
C	3, 3	0, 5
D	5, 0	1, 1

Rock, Paper, Scissors

A_1 \ A_2	Rock	Paper	Scissors
Rock	0, 0	–1, 1	1, –1
Paper	1, –1	0, 0	–1, 1
Scissors	–1, 1	1, –1	0, 0

- Zero-sum game, nothing to be gained by cooperating
- Nash equilibrium for the stage game:
  - choose randomly, $P=1/3$ for each move
- SPE for the repeated game:
  - always choose randomly, $P=1/3$ for each move, expected payoff = 0
- Suppose the other player doesn't use the SPE strategy
  - If you can predict their actions well, you may be able to do much better
- One reason the other agents might not use the SPE strategy:
  - Because they may be trying to predict *your* actions too

Rock, Paper, Scissors

A_1 \ A_2	Rock	Paper	Scissors
Rock	0, 0	–1, 1	1, –1
Paper	1, –1	0, 0	–1, 1
Scissors	–1, 1	1, –1	0, 0

- 1999 international roshambo programming competition

  www.cs.ualberta.ca/~darse/rsbpc1.html

  - Round-robin tournament:
    - 55 programs, 1000 iterations for each pair of programs
    - Lowest possible score = –55000; highest possible score = 55000
  - Average over 25 tournaments:
    - Lowest score (*Cheesebot*): –36006
    - Highest score (*Iocaine Powder*): 13038
      - http://www.veoh.com/watch/e1077915X5GNatn

Infinitely Repeated Games

- An infinitely repeated game in extensive form would be an infinite tree
  - ➢ Payoffs can't be attached to any terminal nodes
- Let $r_i^{(1)}, r_i^{(2)}, \ldots$ be an infinite sequence of payoffs for agent i
  - ➢ the sum usually is infinite, so it can't be i's payoff
- Two common ways around this problem:

1. **Average reward**: average over the first k iterations; let $k \to \infty$

$$\lim_{k \to \infty} \sum_{j=1}^{k} r_i^{(j)} / k$$

2. **Future discounted reward**: $$\sum_{j=1}^{\infty} \beta^j r_i^{(j)}$$

   - $\beta \in [0,1)$ is a constant called the *discount factor*

➢ Two possible interpretations:

1. The agent cares more about the present than the future
2. At each stage, the game ends with probability $1 - \beta$

Nash Equilibria

- What are the Nash Equilibria in an infinitely repeated game?
  - Often many more equilibria than in the finitely repeated game
- Infinitely repeated prisoner's dilemma:
  - Infinitely many Nash equilibria

- There's a "folk theorem" that tells what the possible equilibrium **payoffs** are in repeated games, if we use average rewards

- First we need some definitions …

Feasible Payoff Profiles

- Stage game G, action profiles $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m$, reward profiles $\mathbf{u}(\mathbf{a}_1), \ldots, \mathbf{u}(\mathbf{a}_m)$
- Example: Prisoner's Dilemma

$$\mathbf{u}(C,C) = (3,3), \quad \mathbf{u}(C,D) = (0,5), \quad \mathbf{u}(D,C) = (5,0), \quad \mathbf{u}(D,D) = (1,1)$$

- In the repeated game, a payoff profile $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ is *feasible* if $\mathbf{r}$ is a convex rational combination of $\mathbf{u}(\mathbf{a}_1), \ldots, \mathbf{u}(\mathbf{a}_m)$
  - *Convex combination*: $\mathbf{r} = c_1\, \mathbf{u}(\mathbf{a}_1) + \ldots + c_j\, \mathbf{u}(\mathbf{a}_j) + \ldots + c_n\, \mathbf{u}(\mathbf{a}_n)$
    - $c_1, c_2, \ldots, c_m$ are nonnegative numbers that sum to 1
  - *Rational* combination: $c_1, c_2, \ldots, c_m$ are rational numbers
- Intuitive meaning:
  - $\mathbf{r}$ is feasible if there's a finite sequence of action profiles $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(n)}$ whose average reward profile is $\mathbf{r}$
  - Can achieve $\mathbf{r}$ if the players repeat the action profiles *ad infinitum*

Feasible Payoff Profiles

- Stage game G, action profiles $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m$, reward profiles $\mathbf{u}(\mathbf{a}_1), \ldots, \mathbf{u}(\mathbf{a}_m)$
- Example: Prisoner's Dilemma

$$\mathbf{u}(C,C) = (3,3), \quad \mathbf{u}(C,D) = (0,5), \quad \mathbf{u}(D,C) = (5,0), \quad \mathbf{u}(D,D) = (1,1)$$

- (2, 13/4) is feasible
  - Sequence of action profiles (C,C), (C,D), (C,D), (D,C)
    - $\frac{1}{4}(\mathbf{u}/(C,C) + \mathbf{u}(C,D) + \mathbf{u}(C,D) + \mathbf{u}(D,C))$

      $= \frac{1}{4}((3,3) + (0,5) + (0,5) + (5,0))$

      $= \frac{1}{4}(8,13)$

- (5,5) isn't feasible; no convex combination can produce it
  - If one agent's average payoff is 5, then the other's is 0
- (π/2, π/2) isn't feasible; no **rational** convex combination can produce it

Enforceable Payoff Profiles

- A payoff profile $\mathbf{r} = (r_1, \ldots, r_n)$ is **enforceable** if for each i,
  - $r_i \geq$ player i's minimax value in G
- Intuitive meaning:
  - If i deviates from the sequence of action profiles that produces $\mathbf{r}$, the other agents can punish i by playing their minimax strategy profile against i
    - reduces i's average reward to i's minimax value
- The other agents can do this by using **grim trigger** strategies:
  - Generalization of the Grim strategy
    - If any agent i deviates from the sequence of actions it is supposed to perform, then the other agents punish i forever by playing their minimax strategies against i

	Agent 1	*Agent 2*	
	C	C	
	C	*D*	
	C	*D*	
	D	C	
	C	C	
	C	*D*	
	C	*D*	
	D	C	
	C	$\boxed{D}$	deviate
	D	…	
	D	…	
punish	*D*	…	
	D	…	
	⋮	⋮	

The Theorem

Theorem: If G is infinitely repeated game with average rewards, then

- If there's a Nash equilibrium with payoff profile $\mathbf{r}$, then $\mathbf{r}$ is enforceable
- If $\mathbf{r}$ is both feasible and enforceable, then there's a Nash equilibrium with payoff profile $\mathbf{r}$

Summary of the proof:

- **Part 1**: Use the definitions of minimax and best-response to show that in every Nash equilibrium, each agent i's average payoff $\geq$ i's minimax value
- **Part 2:** Show how to construct a Nash equilibrium that gives each agent i an average payoff r_i
    - The agents are grim-trigger strategies that cycle in lock-step through a sequence of action profiles $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(n)}$ such that $\mathbf{r} = (\mathbf{u}(\mathbf{a}^{(1)}) + \mathbf{u}(\mathbf{a}^{(2)}) + \ldots + \mathbf{u}(\mathbf{a}^{(n)}))/n$
    - No agent can do better by deviating, because the others will punish it
      => Nash equilibrium

Iterated Prisoner's Dilemma

	C	*D*
C	3, 3	0, 5
D	5, 0	1, 1

- For a finitely iterated game with a large number of iterations, the practical effect can be roughly the same as if it were infinite
- E.g., the Iterated Prisoner's Dilemma
- Widely used to study the emergence of cooperative behavior among agents
  - e.g., Axelrod (1984), *The Evolution of Cooperation*
- Axelrod ran a famous set of tournaments
  - People contributed strategies encoded as computer programs
  - Axelrod played them against each other

TFT with Other Agents

- In Axelrod's tournaments, TFT usually did best
  - » It could establish and maintain cooperations with many other agents
  - » It could prevent malicious agents from taking advantage of it

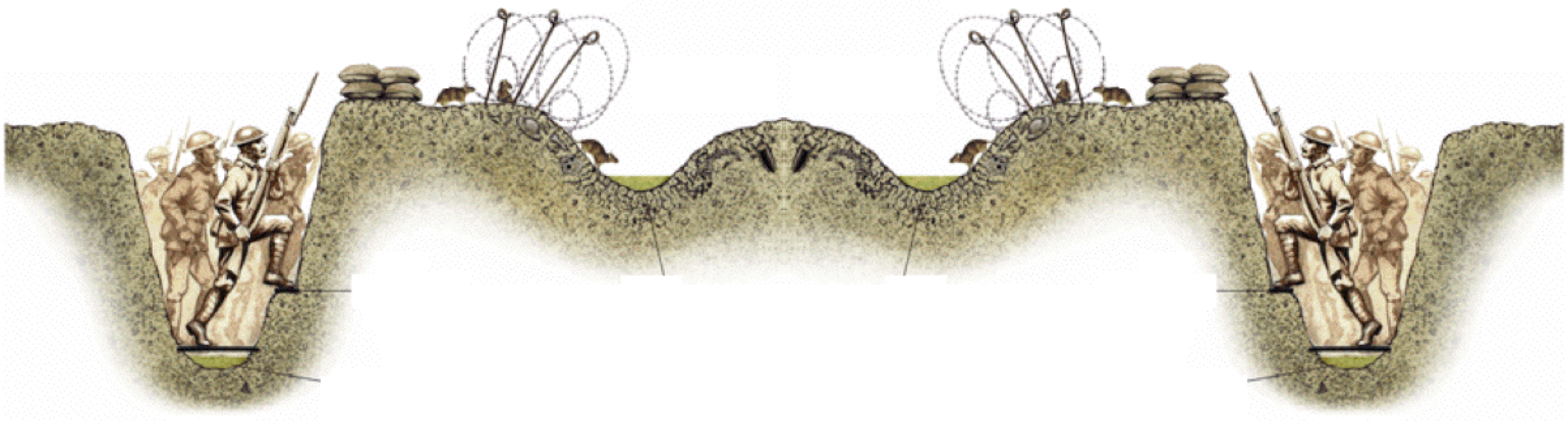
TFT	*AllC, TFT, TFTT, Grim, or Pavlov*
C	C
C	C
C	C
C	C
C	C
C	C
C	C
⋮	⋮

TFT	*AllD*
C	*D*
D	*D*
D	*D*
D	*D*
D	*D*
D	*D*
D	*D*
⋮	⋮

TFT	*Tester*
C	*D*
D	C
C	C
C	C
C	C
C	C
C	C
⋮	⋮

Example:

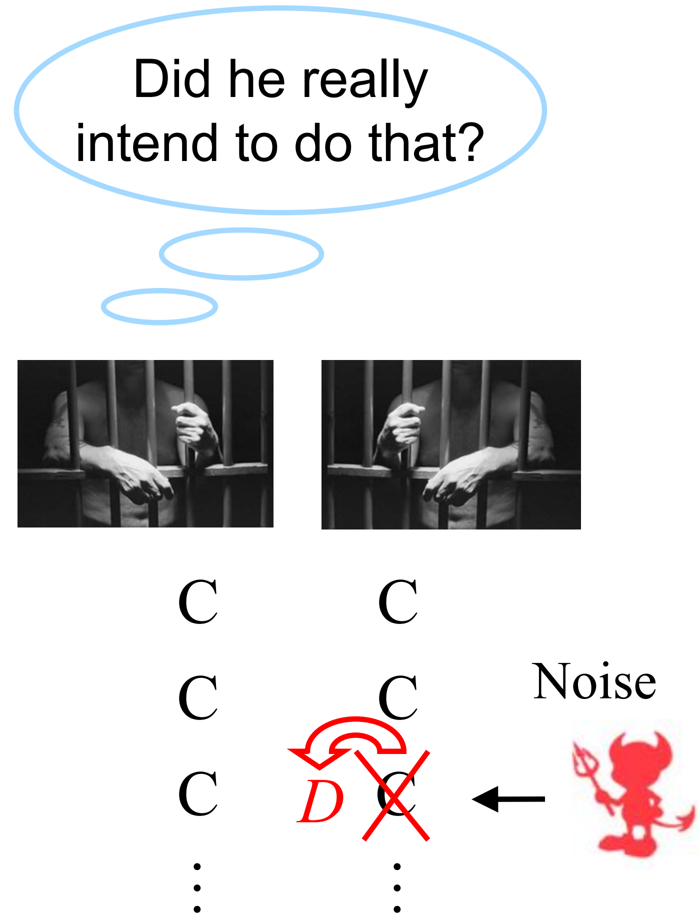
- A real-world example of the IPD, described in Axelrod's book:
  - World War I trench warfare



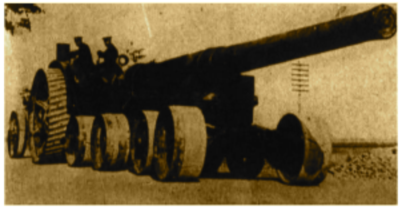
- Incentive to cooperate:
  - If I attack the other side, then they'll retaliate and I'll get hurt
  - If I don't attack, maybe they won't either
- Result: evolution of cooperation
  - Although the two infantries were supposed to be enemies, they avoided attacking each other

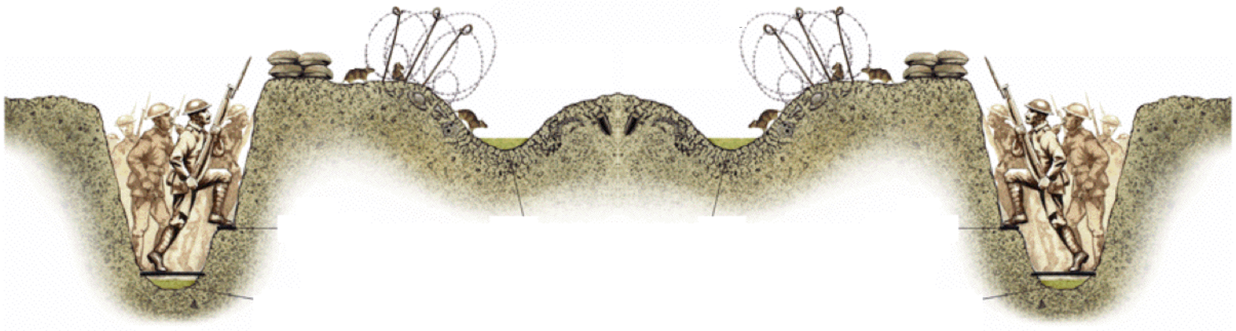
IPD with Noise

- In noisy environments,
  - There's a nonzero probability (e.g., 10%) that a "noise gremlin" will change some of the actions
    - *Cooperate* (C) becomes *Defect* (D), and vice versa
- Can use this to model accidents
  - Compute the score using the changed action
- Can also model misinterpretations
  - Compute the score using the original action



Example of Noise





- Story from a British army officer in World War I:
  - I was having tea with A Company when we heard a lot of shouting and went out to investigate. We found our men and the Germans standing on their respective parapets. **Suddenly a salvo arrived but did no damage**. Naturally both sides got down and our men started swearing at the Germans, when all at once **a brave German got onto his parapet and shouted out: "We are very sorry about that; we hope no one was hurt. It is not our fault. It is that damned Prussian artillery."**
- The salvo wasn't the German infantry's intention
  - They didn't expect it nor desire it

Noise Makes it Difficult to Maintain Cooperation

- Consider two agents who both use TFT
- One accident or misinterpretation can cause a long string of retaliations


	C	C	
	C	C	
	C	C	Noise
	C	D ~~C~~	
Retaliation	D	C	
	C	D	Retaliation
Retaliation	D	C	
	C	D	Retaliation
	⋮	⋮	

Some Strategies for the Noisy IPD

- **Principle**: be more forgiving in the face of defections

- Tit-For-Two-Tats (TFTT)
  - Retaliate only if the other agent defects twice in a row
    - Can tolerate isolated instances of defections, but susceptible to exploitation of its generosity
    - Beaten by the Tester strategy I described earlier
- Generous Tit-For-Tat (GTFT)
  - Forgive randomly: small probability of cooperation if the other agent defects
  - Better than TFTT at avoiding exploitation, but worse at maintaining cooperation

Discussion

- The British army officer's story:
  - a German shouted, "We are very sorry about that; we hope no one was hurt. It is not our fault. It is that damned Prussian artillery."
- The apology avoided a conflict
  - It was convincing because it was consistent with the German infantry's past behavior
  - The British had ample evidence that the German infantry wanted to keep the peace
- If you can tell which actions are *affected* by noise, you can avoid *reacting* to the noise
- IPD agents often behave deterministically
  - For others to cooperate with you it, helps if you're predictable
- This makes it feasible to build a model from observed behavior

The DBS Agent

- Work by Tsz-Chiu Au (one of my PhD graduates)
  - Now a professor elsewhere

- From the other agent's recent behavior, DBS builds a model of their strategy
- DBS use the model
  - to filter noise
  - to help plan its next action

Modeling the other agent

- A set of rules of the following form

  action profile at previous stage $\Rightarrow$

  Pr[the other agent will play *C* in the current stage]

  - Four rules: one for each of (C,C), (C,D), (D,C), and (D,D)
- e.g., TFT is

  (C, C) $\Rightarrow$ 1; (C, D) $\Rightarrow$ 1;

  (D, C) $\Rightarrow$ 0; (D, D) $\Rightarrow$ 0

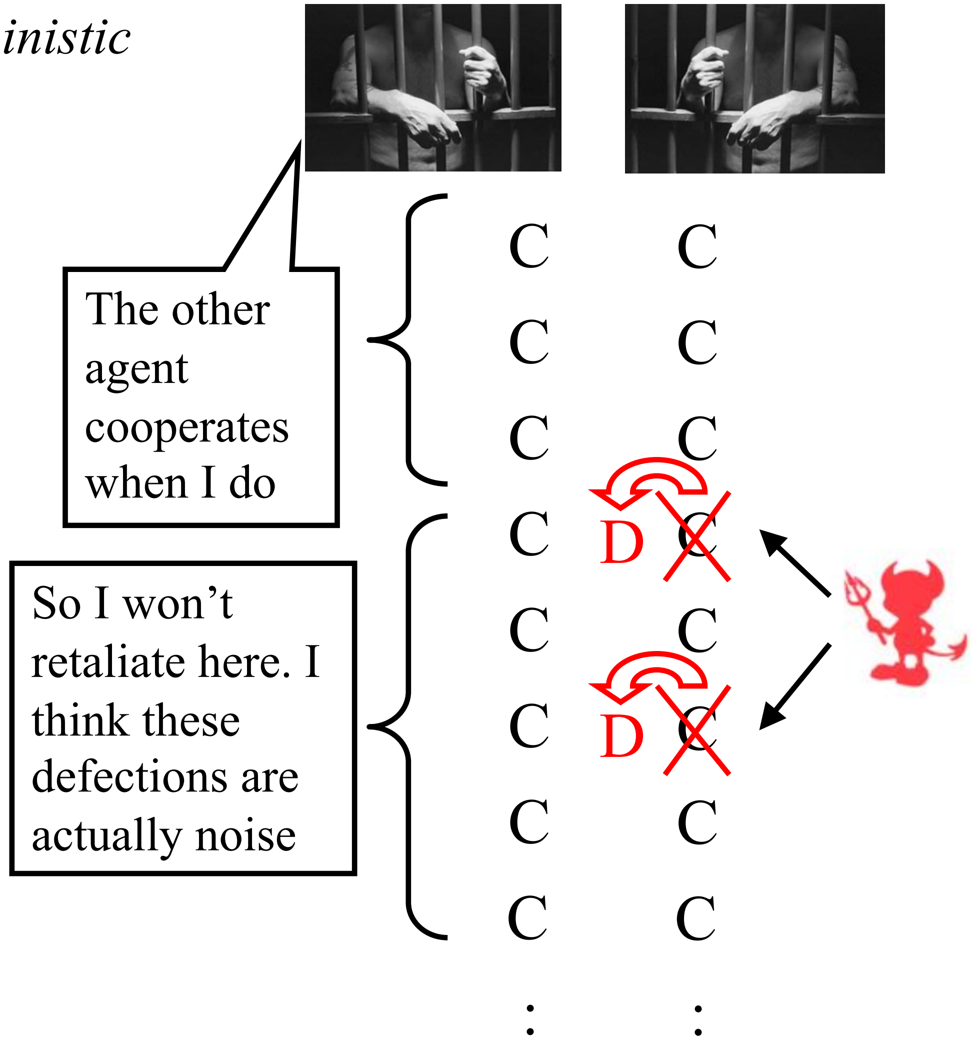
- How to get the probabilities?
  - One way: look at the agent's behavior in the recent past
- During the last k iterations,
  - What fraction of the time did the other agent cooperate at iteration j when the action profile was (x,y) at iteration $j-1$?

Modeling the other agent

- The rules can only model a very small set of strategies
- They don't even model *TFTT* correctly:
  - If *TFTT* defects, it's because the other player defected in the past *two* stages
- But we're not trying to model an agent's entire strategy.
  - Just want a simple model that can make reasonable predictions of an agent's next action
- If an agent's behavior changes, then the probabilities in π will change
  - e.g., after *Grim* defects a few times, the rules will give a very low probability of it cooperating again

Noise Filtering

- Suppose the applicable rule is *deterministic*
  - P[other agent will play C] = 0
    - or
  - P[other agent will play C] = 1
- Suppose DBS sees the other agent playing the opposite of what the rule predicts
  - Assume the observed action is noise
  - Behave as if the action were what the rule predicted



Change of Behavior

- Anomalies in observed behavior can be due either to noise or to a genuine change of behavior
- Changes of behavior occur because
  - the other agent can change their behavior anytime
  - E.g., suppose noise affects one of DBS's actions
    - other agent reacts to the noise rather than DBS's intended action
    - DBS doesn't know this happened
- How to distinguish noise from a real change of behavior?

I am *Grim*. If you ever defect, I will never forgive you.

C	C
C	C
~~C~~ D	C
C	D
C	D
C	D
:	D
:	D
:	:

These moves are *not* noise

Detection of a Change of Behavior

Temporary tolerance:

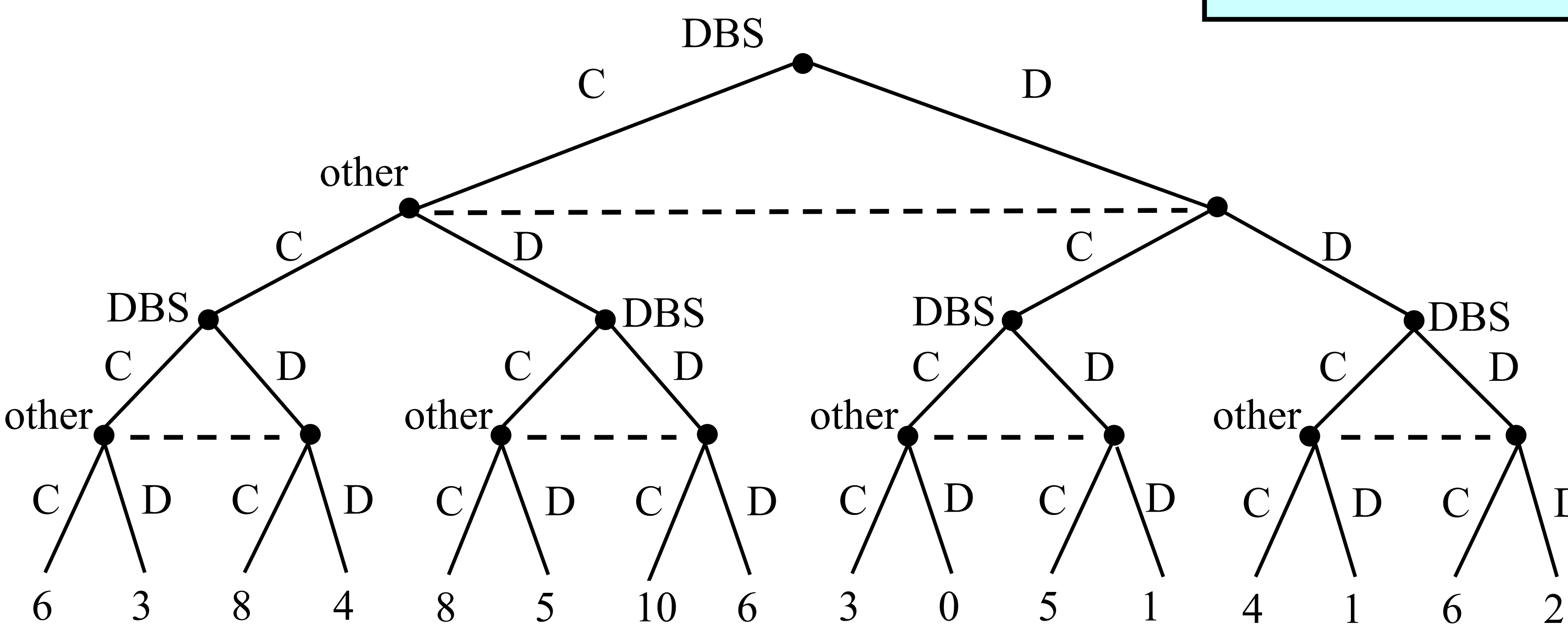
- When we observe unexpected behavior from the other agent
  - Don't immediately decide whether it's noise or a real change of behavior
  - Instead, defer judgment for a few iterations
- If the anomaly persists, then recompute the rules based on the other agent's recent behavior

The other agent cooperates when I do	C	C
	C	C
	C	C
The defections might be accidents, so I shouldn't lose my temper too soon	C	*D*
	C	*D*
	C	*D*
I think the other agent's has really changed, so I'll change mine too	*D*	*D*
	D	*D*
	:	:

Modified Version of Game-Tree Search

- At nodes where DBS moves, v = max of children's values
- At nodes where the other agent moves,
  - Use the rules to get probabilities that the agent will play C or D
  - Compute weighted average of children's values
- At leaf nodes, eval = DBS's total payoff so far

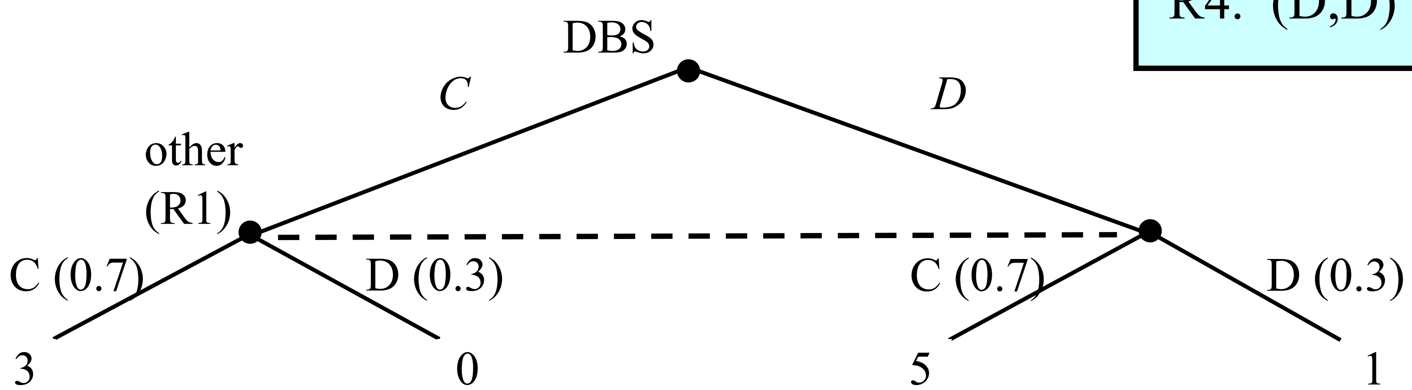
Suppose the rules are
R1. (C,C) → 0.7
R2. (C,D) → 0.4
R3. (D,C) → 0.1
R4. (D,D) → 0.1



Example

- Suppose previous action profile was (C,C)
- Search to depth 2
  - v(C) = 0.7*3 + 0.3*0 = 2.1 + 0 = 2.1
  - v(D) = 0.7*5 + 0.3*1 = 3.5 + 0.3 = 3.8
- So *D* looks better
- Is it really what DBS should choose?

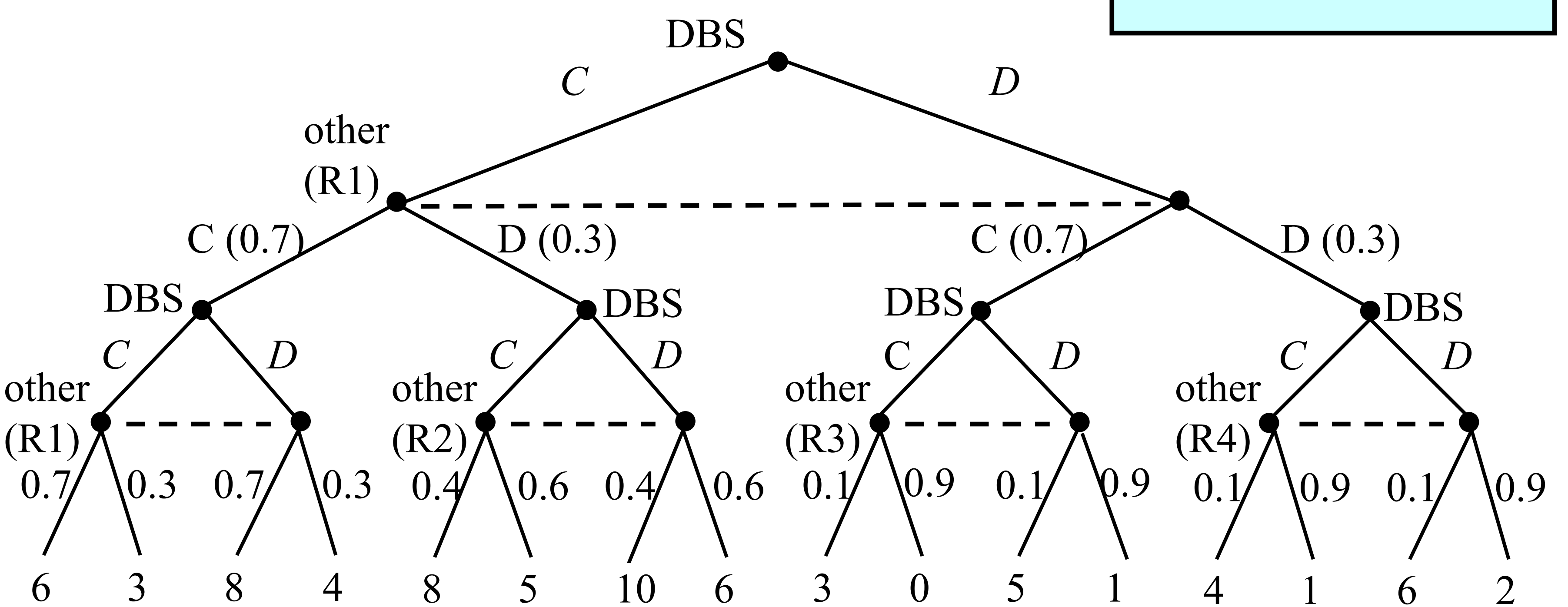
Suppose the rules are
R1. (C,C) → 0.7
R2. (C,D) → 0.4
R3. (D,C) → 0.1
R4. (D,D) → 0.1



Example

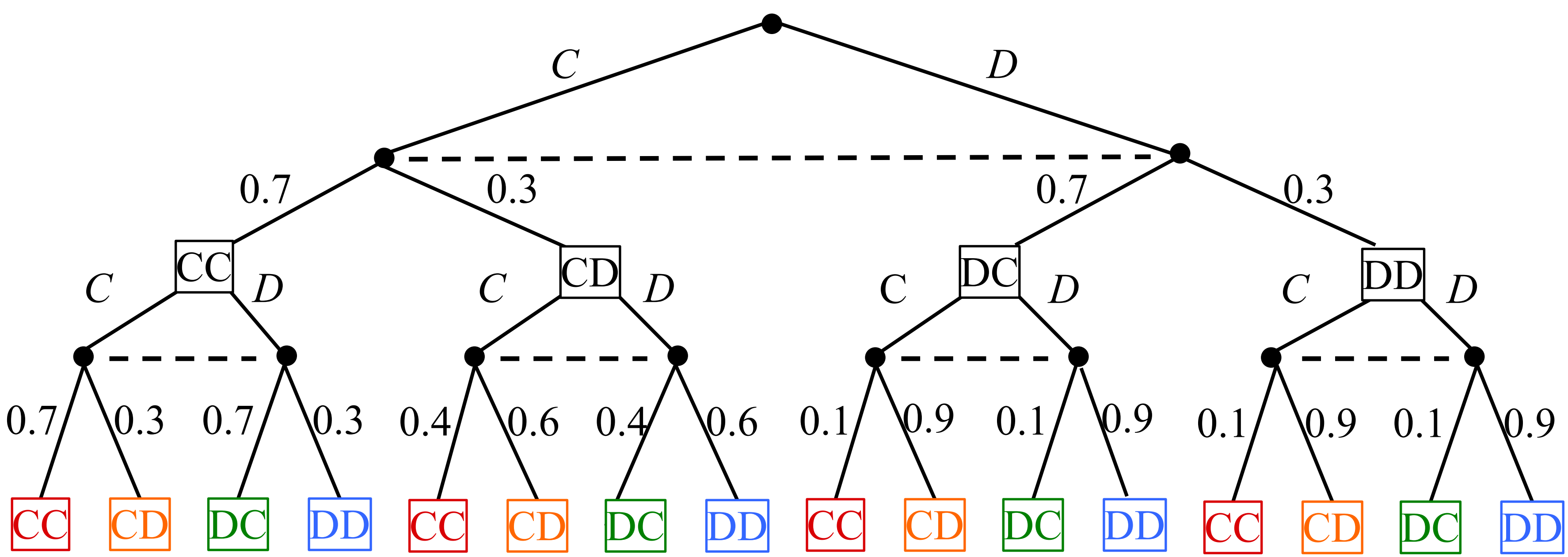
- If DBS plays *D* in stage 1, the other agent is very likely to retaliate with *D* in stage 2
- Depth-2 search won't see this, but depth 4 will
  - In general, it's best to use a large search depth
- Problem: game trees grow exponentially
  - How to search deeply?

Suppose the rules are
R1. (C,C) → 0.7
R2. (C,D) → 0.4
R3. (D,C) → 0.1
R4. (D,D) → 0.1



Search Algorithm

- Assumption: other agent's strategy won't change in the future
  - Current rules will accurately predict **all** their future behavior
  - The rules depend **only** on the previous iteration
- Collapse the tree into a graph
- At each level, just four subtrees
  - one for CC, one for CD, one for DC, one for DD
- Makes the search polynomial in the search depth
  - Can easily search to depth 60
- This generates pretty good moves



20th Anniversary IPD Competition

http://www.prisoners-dilemma.com

- Category 2: IPD with noise
  - 165 programs participated
- DBS dominated the top 10 places
- Two agents scored higher than DBS
  - They both used *master-and-slaves* strategies

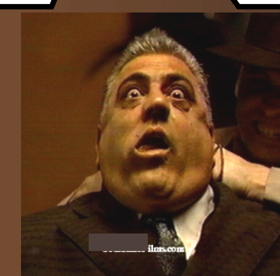
Rank	Program	Avg. score
1	BWIN	433.8
2	IMM01	414.1
3	DBSz	408.0
4	DBSy	408.0
5	DBSpl	407.5
6	DBSx	406.6
7	DBSf	402.0
8	DBStft	401.8
9	DBSd	400.9
10	lowESTFT_classic	397.2
11	TFTIm	397.0
12	Mod	396.9
13	TFTIz	395.5
14	TFTIc	393.7
15	DBSe	393.7
16	TTFT	393.4
17	TFTIa	393.3
18	TFTIb	393.1
19	TFTIx	393.0
20	mediumESTFT_classic	392.9

Master & Slaves Strategy

- Each participant could submit up to 20 programs
- Some submitted programs that could recognize each other
  - (by communicating pre-arranged sequences of Cs and Ds)
- The 20 programs worked as a team
  - 1 master, 19 slaves
  - When a slave plays with its master
    - Slave cooperates, master defects
    - => maximizes the master's payoff
  - When a slave plays with an agent not in its team
    - It defects
    - => minimizes the other agent's payoff

My goons give me all their money …

… and they beat up everyone else



Comparison

- Analysis
  - Each master-slaves team's average score was much lower than DBS's
  - If BWIN and IMM01 had each been restricted to ≤ 10 slaves, DBS would have placed 1st
  - Without any slaves, BWIN and IMM01 would have done badly
- In contrast, DBS had no slaves
  - DBS established cooperation with *many* other agents
  - DBS did this *despite* the noise, because it filtered out the noise

Summary

- Finitely repeated games – backward induction
- Infinitely repeated games
  - average reward, future discounted reward
  - equilibrium payoffs
- Non-equilibrium strategies
  - opponent modeling in rock-paper-scissors
  - iterated prisoner's dilemma with noise
    - opponent models based on observed behavior
    - detection and removal of noise
    - game-tree search against the opponent model
  - 20^{th} anniversary IPD competition