BCSE Game Theory 06-02

Dynamic Games and Normal-Form Conversion

Author

Nov. 12, 2025

Today's Agenda

Today's Goals

- Describe information sets in perfect-information dynamic games (while keeping complete-information payoffs) and motivate why we relax that assumption.
- Translate extended-form strategies into a normal-form payoff matrix.
- Compare Nash equilibria in the normal form with subgame perfection on the tree.
- Prepare computational tools for converting between representations.

Bridge from Lecture 06-01

- We now assume the tree representation from yesterday is available.
- Backward induction delivered one subgame perfect equilibrium.
- To check other equilibria we convert the tree into the normal form.
- The conversion also clarifies how strategies encode off-path behaviour.

Information Sets Refresher

Perfect-Information Dynamic Games

Definition: Perfect-Information Dynamic Game

Every decision node is contained in a singleton information set, so the player who moves knows the entire history of previous actions.

- All previous moves are observed before a player chooses, which makes backward induction feasible without specifying beliefs.
- Finite perfect-information games therefore admit a unique path under straightforward backward induction (once payoffs are generic).
- Lecture 06-01 operated entirely in this environment; today's material builds on that baseline.

Imperfect-Information Dynamic Games

Definition: Imperfect-Information Dynamic Game

At least one information set contains multiple decision nodes that are indistinguishable to the player who must act there.

- The mover knows whose turn it is but cannot tell which history within the information set has actually occurred, so a single contingent plan must fit every node in the set.
- Payoffs can still be complete-information (everyone knows utility functions), but delayed observability or simultaneous moves introduce uncertainty about past actions.
- Once information sets have more than one node we must specify beliefs, which motivates the conversion to the normal form developed later in this lecture.

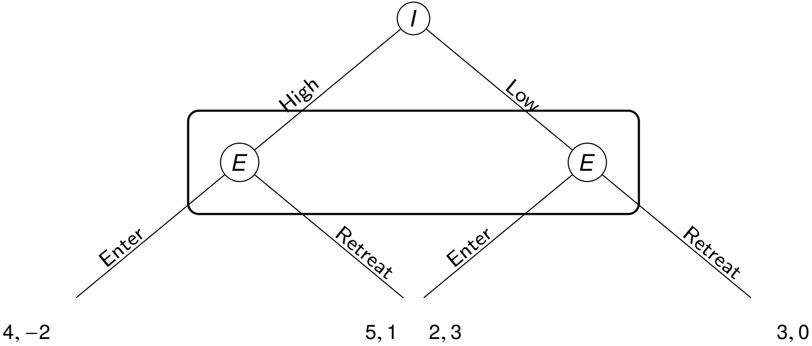
Observability in Complete Information (Recap of 06-01)

- In Lecture 06-01 we assumed every decision node is fully identified by preceding actions.
- Players knew who moved previously and what was chosen, so each information set was a singleton.
- We still recorded them to remind ourselves which contingencies exist, but beliefs were degenerate.
- Starting in 06-02 we allow information sets with multiple nodes, so beliefs and delayed observability matter.

Histories and Information Sets

- A history h is a sequence $(a_1, a_2, \ldots, a_k)$ of past actions.
- In Lectures 06-01 we always knew which branch had been taken, so the information set containing h coincided with $\{h\}$.
- From this lecture onward we consider situations in which players may have to act without observing the full history, so sets can contain multiple nodes.
- Off-path histories (e.g., Low investment followed by Retreat) must remain in the plan regardless of whether they are observed.

Information Set When History Is Hidden



Assume I's investment decision becomes public only after some delay, so the entrant must move before learning whether High or Low was chosen.

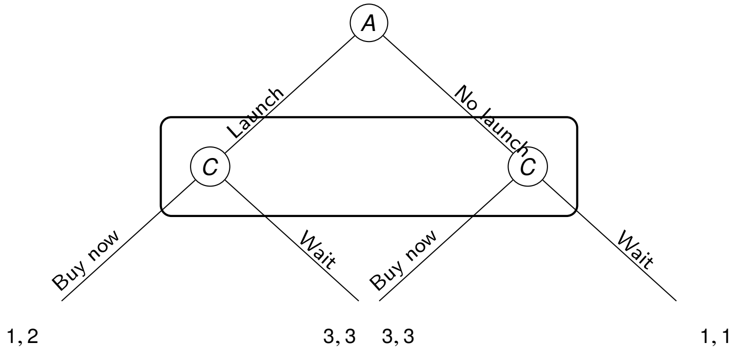
Information Set When History Is Hidden: Implications

- E assigns a belief $\mu = \Pr(\text{High} \mid I_E)$ and picks Enter or Retreat based on expected payoff.
- Retreat payoffs differ because the incumbent's monopoly profit and the entrant's outside option both depend on whether the costly capacity investment was made.
- This non-degenerate information set forces us to specify beliefs and is the foundation for sequential rationality.

Information Set Example: Apple Launch Rumor

- Apple decides whether to launch a new flagship model, but the official announcement reaches customers only after they must decide whether to buy the current model.
- Customers therefore act at an information set containing two histories and must prescribe Buy/Wait actions for each possible belief about the launch.
- Because the history is indistinguishable inside the information set, customers have to commit to the same contingency plan regardless of which node is actually true.

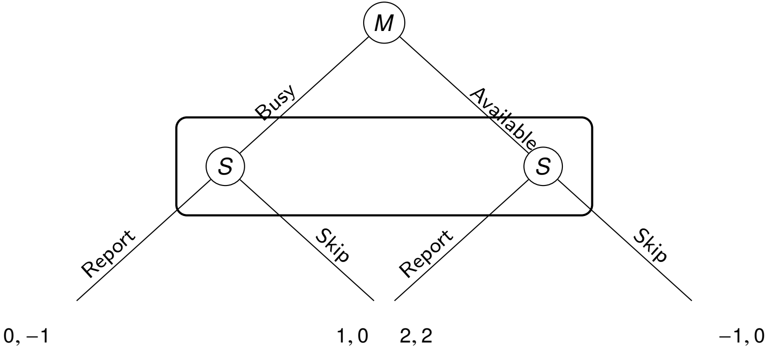
Apple Launch Rumor Tree



Information Set Example: Busy Manager Signal

- A manager M can be **Busy** or **Available**. The subordinate S must decide whether to **Report** before knowing which state the manager is in.
- Reporting to a busy manager wastes effort (negative payoff for S and no gain for M), whereas reporting to an available manager produces a performance boost for both.
- Because S's information set contains both histories, the same Report/Skip plan must apply regardless of the manager's hidden state, so M has incentives to signal availability.

Busy Manager Signal Tree



- Because the subordinate cannot identify the node, report/skip actions must coincide inside the information set, and if the manager always appears busy, information sharing across the organisation will stall.

Why Track Information Sets Anyway?

- They remind us which actions require contingency planning.
- Computational routines attach strategy choices to each information set.
- The notion generalises smoothly once we add imperfect information.
- Leaving them explicit avoids mistakes when we extend a model.

Normal-Form game and Extensive-form game

Simultaneous-Move Nodes (Preview)

- If two players move simultaneously we would connect their nodes by a shared information set.
- Complete information breaks because players cannot identify the branch.
- Today we abstain from such nodes, but keep the vocabulary ready.
- When we reach imperfect information we will reuse the same symbols.

Extensive-Form Game Definition

Definition: Extensive-Form Game

An extensive-form game specifies the player set N, the collection of histories $H \cup Z$, a player function $P(h)$ assigning the mover after each non-terminal history $h \in H$, feasible action sets $A(h)$, information sets partitioning nodes that are indistinguishable to the mover, and payoff functions $u_i(z)$ for every player i at each terminal history $z \in Z$.

- The structure records both the timing of moves and what each player knows when acting.
- Perfect-information trees are special cases where every information set is a singleton.
- Allowing larger information sets accommodates simultaneous or unobserved moves.

Strategies in the Extended Form

Strategy Definition

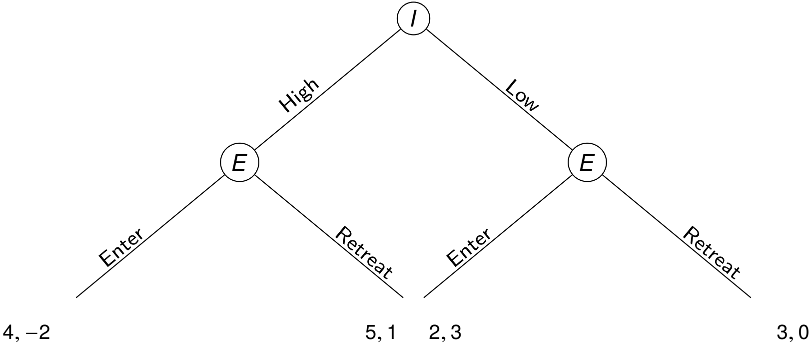
- A pure strategy for player i specifies an action in every information set belonging to i.
- Denote the strategy set by S_i; the profile space is $S = \prod_{i \in N} S_i$.
- Strategies can be described textually ("Enter after High") or coded as tuples.
- Enumerating these strategies is the first step toward the normal form.

Enumerating Strategies: Algorithm

1. List all histories where player i moves.
2. For each history, note the available action set $A(h)$.
3. Form the Cartesian product of these action sets.
4. Label each resulting combination with a descriptive name.

- Complexity grows quickly; we often rely on scripts to automate the enumeration.

Strategies in the Investment Game



- Incumbent I: $S_I = \{\text{High}, \text{Low}\}$.
- Entrant E: $S_E = \{\text{Enter/Enter}, \text{Enter/Retreat}, \text{Retreat/Enter}, \text{Retreat/Retreat}\}$.
- Notation shorthand: EE, ER, RE, RR to save space in matrices.
- This enumeration is the basis for the normal-form payoff table.

From Tree to Matrix: Workflow

1. Pick a strategy profile (s_I, s_E).
2. Follow the tree: start at the root, apply the action prescribed by s_I or s_E depending on who moves.
3. Record the terminal history reached.
4. Write down the payoffs u_I, u_E for that profile.

- Repeat for all profiles to fill the matrix.

Payoff Matrix for the Investment Game

	EE	ER	RE	RR
High	$(4, -2)$	$(4, -2)$	$(5, 1)$	$(5, 1)$
Low	$(2, 3)$	$(3, 0)$	$(2, 3)$	$(3, 0)$

- Duplicated columns reflect identical behaviour off path.
- Nevertheless we keep them all to maintain the definition of strategies.

Locating Nash Equilibria

- Best responses: I prefers High against EE/ER and prefers Low against RE/RR.
- E prefers Enter after High, Retreat after Low.
- The profile (High, ER) is a Nash equilibrium; so is (High, EE).
- Only (High, ER) survives the credibility test because it is consistent with backward induction. (Player E prefers Enter after Low.)

Comparing NE with Credibility

- Normal-form NE check: look for mutual best responses in the matrix.
- Credibility check: require best responses in every subgame—backward induction guarantees it.
- Extra NE may exist because off-path threats are not credible.
- Conversion helps us identify such extraneous equilibria explicitly.

Diagnosing Extraneous Equilibria

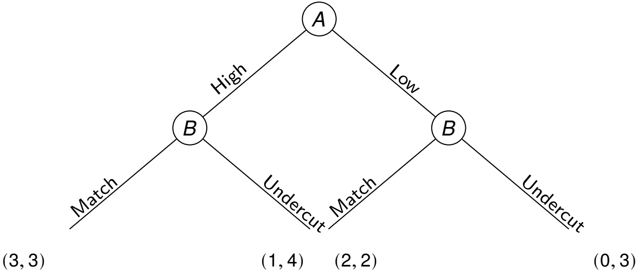
- (High, EE): entrant threatens Enter after Low but the threat is never tested.
- In the tree, if Low occurred the entrant would still Enter, harming both players.
- Because the threat is not credible, backward induction rejects the profile.
- Conclude that the credibility check refines NE by eliminating non-credible contingencies.

Additional Examples

Sequential Pricing Game

- Firm A posts a price first; Firm B observes and decides whether to match or undercut.
- Matching keeps prices high; undercutting steals demand but reduces industry profit.
- Analysing this tree with the normal form exposes which pricing threats are credible and which equilibria backward induction rules out.

Sequential Pricing Game Tree



Example: Pricing Game in Normal Form

	M/M	M/U	U/M	U/U
High	$(3,3)$	$(3,3)$	$(1,4)$	$(1,4)$
Low	$(2,2)$	$(0,3)$	$(2,2)$	$(0,3)$

- B's columns specify responses after (High, Low). Only M/U differs off the realised path when A chooses High.
- Matrix analysis recovers two Nash equilibria: $(\text{High}, \text{M}/\text{U})$ and $(\text{Low}, \text{U}/\text{U})$.
- Backward induction eliminates $(\text{Low}, \text{U}/\text{U})$ because B would still undercut after Low, harming A.

Conversion Toolkit

Algorithm Checklist

- Confirm that every information set has its actions listed before forming products.
- Use consistent labels when exporting strategies to code or spreadsheets.
- Validate the payoff table by cross-checking a few profiles manually.
- Document the order of strategies to avoid mismatches when sharing data.

Wrap-Up

Today's Summary

- Strategies are full contingent plans tied to information sets.
- Converting to the normal form exposes all Nash equilibria, including non-credible ones.
- Subgame perfection filters the matrix equilibria down to those supported on the tree.
- Worked pricing examples highlight how the conversion clarifies credible threats.

Checklist for Self-Study

- Convert one of your project games from a tree to a payoff matrix.
- Identify all Nash equilibria and verify which are subgame perfect.
- Document the strategy labels you used; consistent naming aids collaboration.
- Bring questions on imperfect information—the next lecture extends these tools.