BCSE Game Theory 06-01
Extended-Form Games and Backward Induction

Author

Nov. 11, 2025

Today's Agenda

Today's Goals

▶ Formalise the dynamic games studied so far—every decision node is perfectly observed—even though the explicit term "perfect information" will only be defined in Lecture 06-02.

▶ Practise reading a game tree: histories, available actions, and payoffs.

▶ Execute backward induction step by step and link it to subgame perfection.

▶ Connect the tree representation with contingent strategies for each player.

Where We Are

- ▶ Lectures 03–05 reviewed simultaneous-move (static) games, mixed strategies, core equilibrium concepts, and criteria for selecting among equilibria.

- ▶ From now on we analyse sequential decisions with observation along the path.

- ▶ Perfect information (our working assumption in 06-01) means every move is observed before the next decision, while complete information (knowledge of payoffs and types) remains in force.

- ▶ Finite horizon allows us to trace outcomes using backward induction.

Motivating Case: Chain Store Entry

Chain Store Entry as a Normal-Form Game

▶ Incumbent I chooses whether to Fight or Accommodate entry.

▶ Potential entrant E chooses to Enter or Stay Out.

▶ Payoffs measure long-run profit (higher is better for both players).

	Fight	Accommodate
Enter	$(-4, -6)$	$(3, 5)$
Stay out	$(0, 2)$	$(0, 2)$

▶ Multiple Nash equilibria exist: (Stay out, Fight) and (Enter, Accommodate).

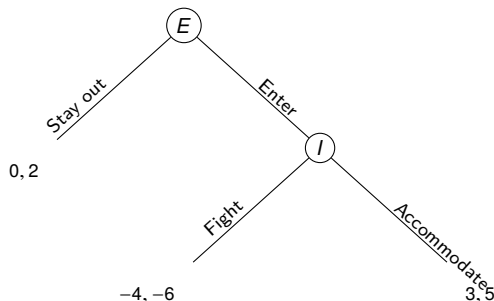▶ The matrix hides how a post-entry fight would unfold over time.

Why the Normal Form Is Unsatisfactory

▶ The matrix treats moves as simultaneous and cannot encode the fact that *I* reacts to entry.

▶ Threats such as "Fight if you enter" are not evaluated for credibility—both equilibria look equally valid.

▶ We cannot describe histories (Enter first, Fight second) or condition strategies on them.

▶ Dynamic issues—capacity commitments, observation, and sequential rationality—require the extended form we study today.

Extended-Form Building Blocks

Chain Store Tree Preview

▶ The entrant E moves first, deciding whether to Enter or Stay out.

▶ If entry occurs, the incumbent I reacts by Fighting or Accommodating, leading to the same payoffs as in the matrix.

▶ Visualising histories clarifies which threats depend on future nodes (game tree).

Why Game Trees?

▶ Sequential interactions embed timing, observability, and conditional choices.

▶ A game tree records the order of moves, the possible actions, and resulting payoffs.

▶ By expanding the tree we can identify credible threats and promises explicitly.

▶ Many applied models (entry deterrence, contract renegotiation) require this structure.

Anatomy of an Extended-Form Game

- Player $i \in N = \{1, \ldots, n\}$ participates in the sequential interaction.
- History $h \in H$ is a finite sequence of past actions that has not yet ended the game.
- Terminal history $z \in Z$ ends the game and determines payoffs.
- Player function $P(h)$ tells us who moves after history h.
- Action set $A(h)$ collects feasible moves at h.
- Payoff $u_i(z)$ is defined for each player i and each terminal history $z \in Z$.

Histories and Tree Structure

▶ The empty history $\emptyset$ is the root node.

▶ If action $a \in A(h)$ is chosen, the history becomes (h, a).

▶ The set $H \cup Z$ is prefix-closed: every prefix of a valid history is valid.

▶ Each history $h \in H$ corresponds to exactly one node in the game tree.

▶ In perfect-information games every node has a unique predecessor.

Action Sets and Feasible Moves

▶ Each $A(h)$ can vary with the history—investments, bids, or exit options.

▶ Finite action sets ensure that the game tree is finite.

▶ Zero-probability branches still matter because strategies must cover every contingency.

▶ Modelling tip: label actions with verbs (Enter, Exit) to keep interpretation clear.

Player Function and Information

▶ $P(h)$ tells us who controls the next move after history h.

▶ In perfect-information games player $i \in N$ knows exactly which node was reached.

▶ Therefore every information set is a singleton and beliefs need not be specified.

▶ Later we will relax this assumption to cover imperfect information.

Terminal Histories and Payoffs

▶ Each terminal history $z \in Z$ associates payoffs $u_i(z)$ to all players.

▶ Payoffs can represent profit, utility, or scores—choose consistent units.

▶ Label terminal nodes with ordered pairs to read the outcome at a glance.

▶ Keep numerical scales comparable when contrasting different branches.
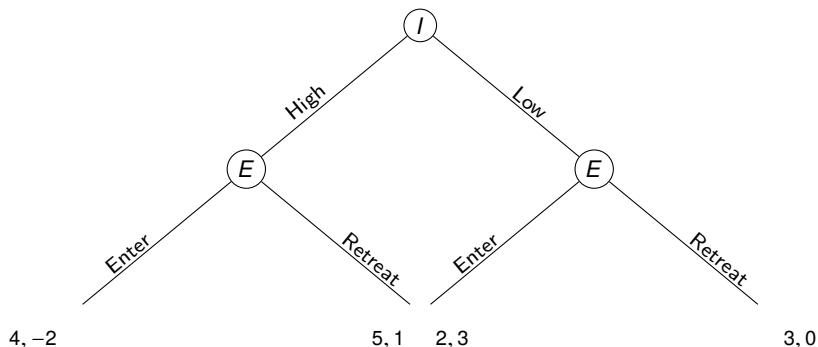
Running Example: Investment Timing

Scenario Overview

▶ Incumbent firm *I* chooses an investment level: High or Low.

▶ **High** means committing to an expensive expansion that boosts monopoly profits if the entrant stays out, while **Low** keeps costs down but leaves the market vulnerable.

▶ Entrant *E* observes the decision and decides to Enter or Retreat.

▶ Payoffs reflect profit outcomes measured in millions of dollars.

▶ We use this tree throughout to illustrate definitions and algorithms.

Payoff Table

	Enter	Retreat
High investment	$(4, -2)$	$(5, 1)$
Low investment	$(2, 3)$	$(3, 0)$

▶ Payoffs are written as (π_I, π_E).

▶ The entrant prefers High investment when staying out; the incumbent dislikes entry after investing.

Game Tree Representation

- ▶ Each complete branch represents a smaller decision problem that can be analysed on its own.
- ▶ We call any branch that starts at a single decision node and includes all descendants a subgame.

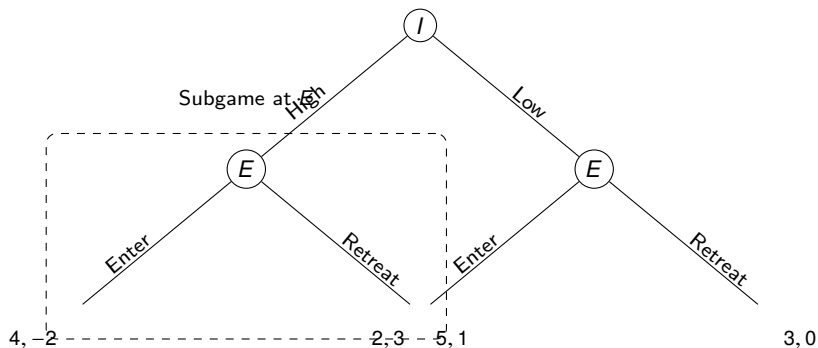# What Counts as a Subgame?

Definition: Subgame

A subgame is a portion of the tree that begins at a single decision node whose information set is a singleton and contains every successor node and terminal outcome below it.

▶ The starting node must include the full history leading up to it —otherwise players at that node would not know which branch they are in, so the resulting fragment would not be a well-defined game.

▶ Once a subgame is identified we can analyse it as a stand-alone perfect-information game with the same payoffs and feasible actions.

▶ Backward induction solves each subgame from the bottom up, ensuring that the strategy profile we construct is a Nash equilibrium in every subgame.
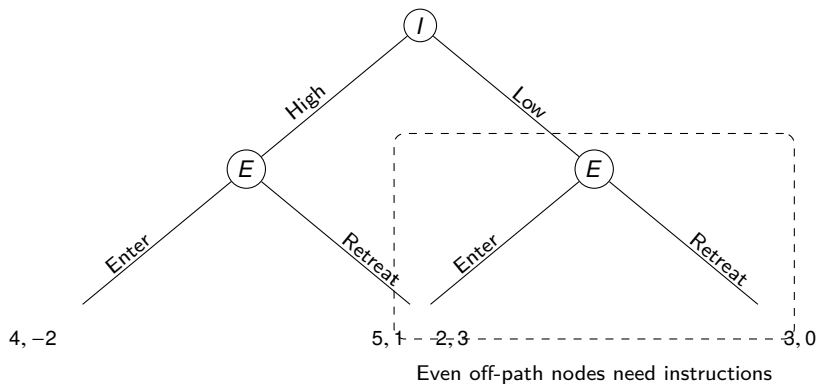
Identifying Subgames
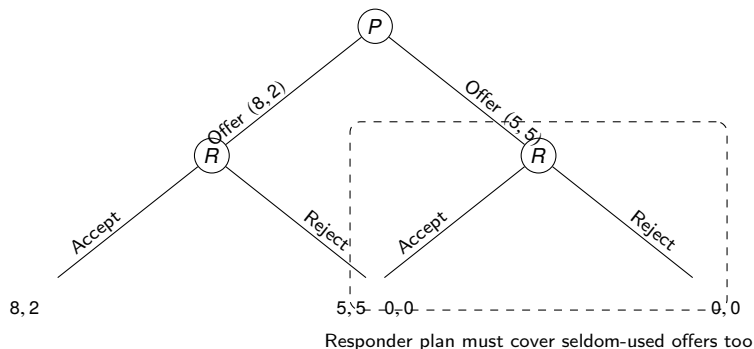
Strategies in the Example

▶ The incumbent's first move is a capacity investment.

▶ Incumbent I: strategies {High, Low}.

▶ Entrant E: strategies specify an action after High and after Low: {Enter/Enter, Enter/Retreat, Retreat/Enter, Retreat/Retreat}.

▶ A strategy profile is a complete contingent plan, even off the realised path.

▶ The term "information set" will be defined formally in Lecture 06-02; for now treat it as the player's decision node.

▶ Later we will map these strategies into a normal-form payoff matrix.

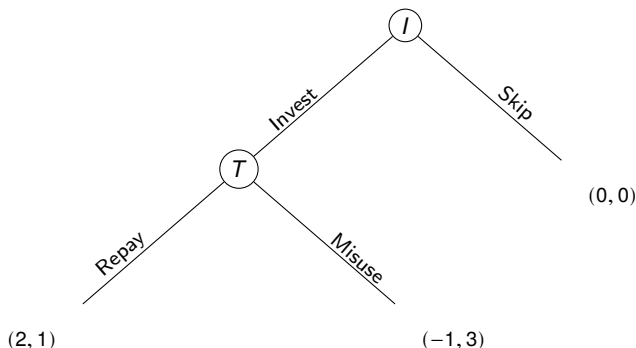# Contingent Plans Illustrated

Even off-path nodes need instructions

- ▶ Even if *I* intends to play High, *E* must state what to do after Low.
- ▶ Contingent plans guarantee that every information set is covered, enabling equilibrium refinements later.

Another Example: Ultimatum Game

Responder plan must cover seldom-used offers too

▶ Even if the proposer rarely chooses (5, 5), the responder must specify Accept or Reject on that branch.

▶ Contingent replies let us evaluate threats such as "Reject any offer below (5, 5)" once information sets are defined in Lecture 06-02.

Case Study: Trust Investment Snapshot

- ▶ One-shot trust games lack credible commitment for the trustee to repay.
- ▶ Backward induction predicts "Skip" unless repeated interaction or reputation changes incentives.

Backward Induction

Why Backward Induction Works

▶ Finite trees allow us to evaluate decisions starting from terminal nodes.

▶ Each player knows future responses because later players observe earlier moves.

▶ Selecting optimal actions at each node yields credible behaviour in every subgame.

▶ The resulting strategy profile is credible everywhere—we will later name this refinement formally.

Credibility and Non-Credible Threats

▶ **Credible threat**: an announced response that the player would still choose when that node is actually reached.

▶ **Non-credible threat**: a planned response that would make the threatening player worse off at its information set.

▶ Trees expose credibility because every information set lists feasible actions and payoffs.

▶ Backward induction discards non-credible threats automatically: if an action is not optimal at its node, it is never selected along the solution path.

▶ We can therefore explain equilibrium refinements purely in terms of observable choices without yet naming new equilibrium concepts.

Algorithm Outline

1. Start at terminal nodes and record payoffs for each player.
2. Move to the parent node: choose the action that maximises the current player's payoff.
3. Replace the subgame with its equilibrium payoff and continue upward.
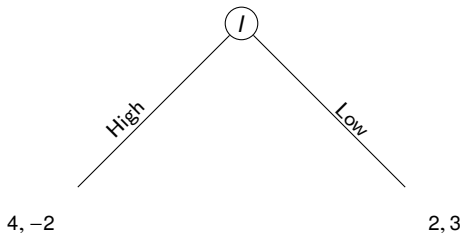4. The sequence of chosen actions defines the backward-induction path.

Verbal Recipe for Backward Induction

▶ Think of the game tree as a collection of smaller subgames that can be summarised one by one.

▶ Starting from any terminal node, record the payoff vector that would result if play reached that node.

▶ At a predecessor node h, compare the recorded payoffs of all actions $a \in A(h)$ and keep the action that maximises player $P(h)$'s payoff.

▶ Replace the entire subgame below h with the chosen payoff vector and continue moving upward until reaching the root.

▶ Because each node is visited once in a finite tree, this verbal procedure completes in time proportional to the number of nodes.

Backward Induction on the Investment Game

▶ Solver looks at E's nodes first: Enter yields 2 after High, 3 after Low.

▶ Retreat gives 1 after High, 0 after Low.

▶ Therefore E prefers Enter after both High and Low.

▶ Knowing this, I compares High with payoff 4 versus Low with payoff 2.

▶ Outcome: I chooses High, E chooses Enter, payoffs $(4, -2)$.

Reduced Game After Solving *E*'s Subgames

- ▶ After solving *E*'s decision nodes, each branch collapses to the payoff that would follow from the unique best reply.
- ▶ The remaining choice is entirely on *I*: High keeps profit 4 while Low falls to 2.
- ▶ Presenting the reduced tree helps students see how backward induction progressively eliminates subgames.

Backward Induction Ensures Credibility

- ▶ At every node we picked a best response for the player who moves there.
- ▶ Any subgame beginning at that node inherits the optimal continuation.
- ▶ Thus the entire profile is a Nash equilibrium in every subgame even though we have not yet given the refinement a formal label.
- ▶ In finite perfect-information games the credible outcome is unique if payoffs differ across actions.

When Backward Induction Fails

- ▶ Infinite horizon or continuous action sets require additional tools (fixed points, calculus).
- ▶ Imperfect information demands beliefs and sequential rationality—we will define these formally in later lectures before relying on them.
- ▶ Multiple optimal actions can produce several credibility-preserving outcomes; tie-breaking rules matter.
- ▶ Payoff indifferences sometimes yield equilibria supported by mixed strategies.

Practical Modelling Checks

▶ Are payoffs aligned with the story? double-check units and order.

▶ Does every node specify feasible actions and the correct moving player?

▶ Have we labelled terminal nodes clearly to avoid transcription errors?

▶ Are there off-path contingencies that warrant discussion or simplification?

Strategies and Representations

Strategies as Contingent Plans

▶ A pure strategy lists an action for every information set of the player.

▶ Even if a node is not reached in equilibrium, the plan must specify a response.

▶ This viewpoint is essential when we translate trees into normal-form matrices.

▶ Consistency between tree labels and strategy names keeps the conversion transparent.

Counting Strategies

▶ Suppose a player moves at histories $h_1, \ldots, h_k$ with action counts $|A(h_j)|$.

▶ The number of pure strategies is the product $\prod_{j=1}^{k} |A(h_j)|$.

▶ In the investment game, I has 2 strategies and E has 4.

▶ Large trees can produce enormous strategy spaces— motivation for computational tools.

From Tree to Normal Form (Preview)

▶ Enumerate strategies for each player as rows and columns of a matrix.

▶ For every strategy profile, follow the tree to the induced terminal history.

▶ Record payoffs $u_i(z)$ in the corresponding cell.

▶ We will automate this conversion in Lecture 06-02 to compare equilibria.

Wrap-Up

Today's Summary

▶ Extended-form games codify timing, information, and payoffs in a single structure.

▶ Backward induction exploits the finite tree to produce an outcome that is credible at every stage.

▶ Strategies are complete contingent plans, ready to be mapped into the normal form.

▶ Rigorous modelling keeps stories, data, and algorithms aligned.

Checklist for Self-Study

▶ Draw and label a new two-stage game from your domain and solve it by backward induction.

▶ Enumerate all strategies and verify that the backward-induction outcome survives a credibility check in every subgame.

▶ Modify one payoff entry in the investment example and predict how the backward-induction path changes.

▶ Prepare questions about imperfect information—our next topic builds on today.