Comprehensive Study Guide: Agent and Multiagent Systems

This study guide summarizes the key concepts, properties, and applications of agent and multiagent systems, based on the provided lecture materials. It aims to be a single, comprehensive document for review.

1. Properties of Agent Systems

1.1 What is an Agent?

- **Definition**: A computer system capable of autonomous action in some environment in order to meet its design objectives.
- **Key point**: Autonomy is crucial; agents act independently and control their internal state.
- **Interaction**: Agents are in a closed-couple continual interaction with its environment: sense – decide – act – sense – decide – act.
    - **Diagram Description**: A diagram shows an AGENT interacting with an ENVIRONMENT. Actions flow from AGENT to ENVIRONMENT, and feedback flows from ENVIRONMENT to AGENT. Another diagram details this interaction: ENVIRONMENT -> sensors -> AGENT -> effectors/actuators -> ENVIRONMENT. Perception leads to decision, which leads to actions.

1.2 Simple (Uninteresting) Agents

- **Thermostat**:
    - Goal: Maintain room temperature.
    - Actions: Heat on / heat off.
    - Sensor: Temperature sensor.
- **Screen Saver**:
    - Goal: Avoid screen burning and prevent others from viewing desktop contents.
    - Actions: Blank the screen / show desktop.
- **Why uninteresting?**: Trivial from a decision-making perspective.

1.3 Intelligent Agents and Agent Properties

Intelligent agents typically exhibit the following types of behavior:

- **Reactivity**:

    - Definition: A system that maintains an ongoing interaction with its environment and responds to changes in time for the response to be useful.
    - Importance: Real-world environments are dynamic; software must account for changes and incomplete information.
- **Pro-activeness**:
    - Definition: Generating and attempting to achieve goals; not solely driven by environmental events.
    - Involves: Taking initiative, recognizing opportunities.
- **Balancing Reactive and Goal-Oriented Behavior**: A key challenge in agent design is to balance timely responses to changing conditions with systematic work towards long-term goals.
- **Social Ability**:
    - Definition: The ability to interact with other agents (and possibly humans) through coordination, cooperation, negotiation, and modeling others.
    - **Coordination**: Managing interdependencies between activities (e.g., sharing a non-sharable resource).
    - **Cooperation**: Working together as a team to achieve a shared goal, especially when no single agent can achieve it alone or when cooperation yields better results.
    - **Negotiation**: The ability to reach agreements on matters of common interest (e.g., sharing a TV).

1.4 Other Agent Properties

- **Autonomy**: Agent's ability to act independently and determine how to achieve its delegated goals/tasks.
- **Adaptivity**: Agent's ability to learn from experience to better interact with the particular environment.
- **Rationality**: Agent's ability to act in a way that maximizes some utility function.
- **Curiosity**: Agent's ability to engage creative, imaginative, or inquisitive reasoning.
- **Believability**: Agent's ability to create a suspension of disbelief, temporarily leading a user to accept the agent as alive or a real character (e.g., game characters).
- **Mobility**: Agent's ability to change its location in the environment, whether physical (robots) or virtual (virtual agents, Internet network).

1.5 Environment Properties

Agent systems are further characterized by properties of their surrounding environment.

- **Sensors**: Define the perceptors for the agent to perceive the world (e.g., cameras, speed sensor, GPS, sonar).
- **Effectors**: Define the actuators for the agent to perform in the world (e.g., start engine, accelerate, brake, turn, start light, change gears).
- **Interaction Diagram Description**: A diagram shows an Agent with Sensors (Input/perceptors) and Effectors (Output/actuators) interacting with the Environment.

Environment Properties:

- **Accessibility**: Accessible vs. Inaccessible.
    - Accessible: Agent can obtain complete, accurate, up-to-date data about the environment's state.
    - Most complex environments are inaccessible.
- **Determinism**: Deterministic vs. Non-deterministic.
    - Deterministic: Action has a single guaranteed effect.

    - Physical world is non-deterministic for humans.
- **Dynamism**: Static vs. Dynamic.
    - Static: World does not change while the agent is deliberating.
- **Continuity**: Discrete vs. Continuous.
    - Discrete: Fixed, finite number of possible actions and percepts.
- **Memory**: Episodic vs. Non-episodic.
    - Episodic: World can be divided into independent intervals (episodes); what happens in one episode has no influence on others.

1.6 Agent Applications

Agent systems can exist in:

- **Physical space**: Robots (e.g., NASA's Mars 2021 Perseverance rover, Unitree Dancing Robots).
- **Cyberspace**: Software and interface agents (e.g., Trading Agents - Automated/Algorithmic Trading, Business Process Automation - Robotic Process Automation).
- **Simulated physical space**: Traffic simulators.
- **Hybrid**: Virtual agents interacting with humans (e.g., LLM/LWM ChatBots: ChatGPT, Gemini, DeepSeek).
- **Games**: Agents in games (e.g., AlphaGO).

2. Introduction to Multiagent Systems

2.1 Motivation: Five Ongoing Trends in Computing

- **Ubiquity**: Processing power in many places and devices.
- **Interconnection**: Computer systems are networked into large distributed systems.
- **Intelligence**: Complex systems performing previously unthinkable tasks.
- **Delegation**: More control given to computers (devices, robots, cars).
- **Human-orientation**: Use of more human-oriented abstractions instead of low-level code.

These trends lead to:

- Need for computer systems that can act effectively on our behalf (implying independence and acting in our best interests).
- Interconnection and Distribution, coupled with the need for systems to represent our best interests, imply systems that can cooperate and reach agreements (or even compete).
- All these trends have led to the emergence of **multiagent systems**.

2.2 Agent and Multiagent Systems: Definitions

- **Agent**: A computer system that is capable of independent (autonomous) action on behalf of its user or owner. Autonomy is key; an agent figures out what to do rather than constantly being told.
- **Multiagent System**: Consists of a set of agents that interact with one another. Agents may have different goals and motivations, requiring cooperation, coordination, and negotiation.

2.3 The Two Key Problems in Multiagent Systems

- **Agent Design**: How to build agents that are independent, autonomous, and able to carry out delegated tasks. Decision-making is a key aspect.
- **Society Design**: How to build agents capable of interacting (cooperating, coordinating, negotiating) with other agents. These are the micro and macro perspectives.

2.4 Challenges in Multiagent Systems

Multiagent Systems address questions such as:

- How can agents interact with each other?
- How can cooperation emerge in societies of self-interested agents?
- What kinds of languages can agents use to communicate?
- How can self-interested agents recognize conflict and reach agreement?
- How can autonomous agents coordinate their activities to cooperatively achieve goals?

What makes the multiagent systems field unique is its emphasis on agents as **computational, information processing entities**.

2.5 Relevance of Agent Systems

Many different views on what multiagent systems are:

- **Agents as a paradigm for software engineering**:
    - Engineering complex software with dynamically interacting components.
    - Emphasizes decentralized approach, unforeseen situations, fault-tolerance, adaptive/flexible behavior.
    - “interaction is probably the most important single characteristic of complex software”
- **Agents as a tool to understand societies**: New tool for simulating society (multiagent based simulations) to understand dynamics/behavior.
- **Agents as a way to search for theoretical foundations**: Derive formal properties from single/multi agent behavior, theorem proofs.
- **Role of agents in other sciences**: Multi-agents is interdisciplinary, influenced by and influences many other fields:
    - **Economics/Game Theory**: Interactions among self-interested agents / economic entities, rational agents able to mimic humans / organizations.
    - **Social Sciences**: Interested in MAS to model/simulated social behavior, model of emotions and their impact, inspiration from agent traits.
    - Other fields: ecology, ethology, philosophy, logic, psychology, sociology, cognitive science, anthropology.
    - Strength: Can use well-founded methodologies.
    - Weakness: Many different views as to what an agent is about.

2.6 Frequently Asked Questions

- **Agents and Artificial Intelligence**: AI focuses on components of intelligence (learn, plan, act), but classical AI often ignores social aspects (communicate, coordinate, cooperate, reach agreements).
- **Agents and Distributed Systems**: Distributed/Concurrent Systems provide much to learn, but agents are assumed to be autonomous and can be self-interested.
- **Agents and Economics/Game Theory**: These fields offer valuable insights, but many concepts (e.g., Nash equilibrium) were developed without a view to computation, and some assumptions (e.g., rational agent) may not be valid for building artificial agent societies.

2.7 History and Prospect

- **1980**: First conference Workshop on Distributed Artificial Intelligence.
- **1980 (Europe)**: MAAMAW (after launch in European Conference on AI).
- **1995**: First international meeting ICMAS.
- **1994**: Workshop Agent Theories, Arch. and Languages (ATAL) launched in ECAI.
- **1997-99 (US), 2000 (Europe)**: Autonomous Agents Conference.
- **2002**: ICMAS and AA merged to launch the largest conference on agents: AAMAS.

3. Agent Architectures

3.1 Introduction to Agent Architectures

- **What is an agent architecture?** Principles describing agent behavior with a formal/abstract view of agents and a map of their internals (control-flow). Goal is to guide agent design and engineering.
- **What is an abstract architecture?** Common principles describing agent behavior independently of their specificities; a template for agent construction.

3.2 Abstract Architectures for Agents

- **Environment States (E)**: A finite set of discrete states, e.g., `E = {e0, e1, ...}`.
- **Actions (Ac)**: A set of actions that transform the environment's state, e.g., `Ac = {α0, α1, ...}`.
- **Run (r)**: A finite sequence of interleaved states and actions: `e0 --α0--> e1 --α1--> e2 ... --α(n-1)--> en`.
- **R**: Set of all possible runs.
- **R_Ac**: Subset of runs ending with an action.
- **R_E**: Subset of runs ending with an environment state.
- **State Transformer (τ)**: `τ: R_Ac -> P(E)`. Maps a run (ending in an action) to a set of possible environment states.
    - **Important points**: History dependent, non-determinism. If `τ(r)` is empty, the system has ended its run.
- **Environment (Env)**: Defined as a triple `<E, e0, τ>`, where `E` is the set of environment states, `e0` is the initial state, and `τ` is the state transformer function.
- **Agent (Ag)**: A function that maps runs to actions: `Ag: R_E -> Ac`. An agent decides what action to perform based on the history it has witnessed.
- **AG**: Set of all agents.

- **R_Ag, Env**: Set of runs of agent `Ag` in Environment `Env`.

3.3 Deductive Reasoning Agents

- **Oldest agent architecture (1956-1985)**: Symbolic reasoning agents, mostly deductive reasoning, with explicit logical reasoning.
- **Problems (1985-present)**: Led to reactive agents movement.
- **Hybrid architectures (1990-present)**: Combine deliberative reasoning and reactiveness.
- **Classical approach**: Agent as a knowledge-based system (Symbolic AI).
- **Two key problems**:
    1. **Transduction problem**: Translating real-world environment into an accurate, adequate symbolic description (Fields: computer vision, speech understanding, learning).
    2. **Representation/reasoning problem**: Symbolically representing information and reasoning with it (Fields: knowledge representation, automated reasoning, planning).
- **Characteristics**: Contains an explicit symbolic representation of the world (internal state as formulae, e.g., `isopen(valve221)`), analogous to human beliefs (can be incorrect/outdated), makes decisions via symbolic reasoning (proving theorems).
- **Agent Decision with State**:
    - `D`: Internal state (set of formulae or database).
    - `see`: `S -> Per` (observe environment).
    - `next`: `D x Per -> D` (update internal state).
    - `action`: `D -> Ac` (decision making with deduction rules).
    - **Diagram Description**: Agent with `see` (perception) and `action` (action) interacting with Environment. `next` updates the internal `state`.
- **Action Selection Function**: `action(DB:D) returns an action Ac`.
    - Attempts to prove `Do(a)` from its database `DB` using deduction rules `ρ`.
    - If `DB |-ρ Do(a)` then return `a`.
    - If no action is explicitly forbidden (`DB |-ρ ¬Do(a) = false`), then return `a`.
- **Vacuum World Example**:
    - **Environment State (S)**: Grid coordinates and dirt status.
    - **Percepts (Per)**: `dirt` or `null`.
    - **Actions (Ac)**: `forward`, `suck`, `turn` (right 90 degrees).
    - **Internal State (DB)**: `In(x, y)` (agent at x,y), `Dirt(x, y)` (dirt at x,y), `Facing(d)` (agent facing direction d).
    - **Deduction Rules (Agent's behavior)**: Examples like `In(x, y) ∧ Dirt(x, y) ⟶ Do(suck).`
- **Final remarks on deductive reasoning agents**:
    - Decision making strategy encoded as logical theory; action reduces to a problem of proof.
    - Elegant with clean logical semantics.
    - **Disadvantages**: High computational complexity of theorem proving, ineffective in time-constrained environments, environment cannot change during decision-making, difficult to represent/reason about complex and dynamic environments.

3.4 Agents as Intentional Systems

- **Intentional Stance**: Develop agent behaviors in terms of mental states (beliefs, desires, wishes, hopes, etc.).
- **Examples**:

"Michael took his umbrella because he believed it was going to rain.", "John worked hard because he wanted to obtain a PhD."

- **Legitimacy**: Useful to attribute beliefs, desires, etc., to artificial agents for modeling complex systems or systems with incompletely known structures.
- **Implementation**: Intentional Systems are the base for deliberative agents, following the intentional stance through practical reasoning.
- **Origin**: Philosophical work of Bratman (e.g., "Intention, Plans and Practical Reason", 1987).
- **Practical Reasoning**: "A matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes." [Bratman, 1990]
    - **Practical Reasoning = Deliberation + Means-Ends Reasoning**
    - **Deliberation**: Deciding what state of affairs an agent wants to achieve from (possibly conflicting) desires.
    - **Means-Ends Reasoning**: Deciding how an agent wants to achieve these states of affairs.
- **The B.D.I. Model**: Inspired by Bratman, based on mental attitudes of:
    - **Beliefs**: Information about the environment, other agents, and itself.
    - **Desires/Goals**: State of affairs to achieve.
    - **Intentions**: Commitments to achieving particular goals.
- **Intentions in Practical Reasoning**:
    - Stronger than mere desires; once intended, the matter is settled.
    - Drive means-ends reasoning; lead to action and attempts to achieve them.
    - **Property: Intentions persist**: Do not give up without good reason.
    - **Property: Intentions constrain future deliberation**: Will not entertain inconsistent options (filter of admissibility).
    - **Property: Intentions influence beliefs**: Closely related to beliefs about the future.
- **Modeling Deliberation (Belief Revision)**:
    - **Belief revision function (brf)**: `brf: 2^Bel x Per -> 2^Bel` (Update beliefs with sensory input and previous belief).
    - **Option generation function (options)**: `options: 2^Bel x 2^Int -> 2^Des` (Use beliefs and existing intentions to generate options/desires).
    - **Filtering function (filter)**: `filter: 2^Bel x 2^Des x 2^Int -> 2^Int` (Choose between competing alternatives and commit).
- **Means-Ends Reasoning**: An agent's means-ends function `plan: 2^Bel x 2^Int x 2^Ac -> Plan` (planner uses environment state/beliefs, goal/task, and possible actions to create a plan).
- **Implementing a Practical Reasoning Agent (Decision-making loop)**:
    1. Observe the world and update beliefs.
    2. Deliberate to decide the intention(s) (determine options, filter).
    3. Use means-ends reasoning to find a plan for the intention(s).
    4. Execute the plan.
    5. Return to step 1.

- **Commitments**: How committed an agent should be to its intention? How long should an intention persist? (Implies temporal persistence).
- **Commitment Strategies**:
    - **Blind commitment**: Maintains an intention until it believes the intention has been achieved (aka fanatical commitment).
    - **Single-minded commitment**: Maintains an intention until it believes that either the intention has been achieved or is no longer possible to achieve.
    - **Open-minded commitment**: Maintains an intention as long as it has not been achieved and it is still desired.

3.5 Reactive Agents

- **Problems with symbolic/logical approaches**: Inherent computational complexity of theorem proving, cannot operate effectively in time-constrained environments, environment cannot change while agent is making a decision, not easy to represent/reason about complex and dynamic environments.
- **Alternatives (mid to late 1980s)**:
    - Rejection of symbolic representation and syntactic manipulation.
    - Idea that intelligent behavior is linked to the environment.
    - Intelligent behavior can emerge from the interaction of various simpler behaviors.
- **What are reactive agents?**: Agents with simple processing units that perceive and quickly react to changes in the environment. Do not use complex symbolic reasoning.
- **Intelligence in reactive systems**: Not a property of a single component; distributed and emerges from interaction among agent components and the environment.
- **Purely reactive agents**: Make no reference to their history (no internal state); decision making entirely on the present.
    - Formally: `Ag: E -> Ac`.
    - `function decide(perception):current_state <- INTERPRET-INPUT(perception),rule <- RULE-MATCH(current_state,rules),action <- RULE-ACTION(rule),return action` (Russell and Norvig, 1994).
- **Reactive architectures inspiration**: Intelligent behavior of animals in the world; complex behaviors come from combining simple individual behaviors.
- **Brooks: Subsumption Architecture**: (Further details would be here if the content was not truncated).

3.6 Hybrid Architectures

(Further details would be here if the content was not truncated).

4. Multiagent Decision Making and Games in Normal Form

4.1 Multiagent Decision Making with Game Theory

- **Context**: Study of multiagent decision making where a group of agents coexist within an environment and make simultaneous decisions.
- **Game Theory**: Mathematical study of interactions among independent, self-interested agents.
    - **Applications**: Economics, political science, biology, psychology, linguistics, computer science (multiagent decision making).
    - **Assumptions**: Agents are self-interested (have preferred outcomes, modeled by utility theory), rational, and reason strategically (consider other agents' decisions).
    - **Key Idea**: The success of an agent depends on the decisions of all agents.
    - **Solution**: A prediction of the outcome based on the assumption that all agents are rational and strategic.

4.2 Normal-Form Games

- Also known as static games or strategic games.
- **Definition**: A (finite, n-person) normal-form game is a tuple (N, A, u) where:
    - N: A finite set of n players, indexed by i.
    - $A = A1 \times ... \times An$: Where Ai is a finite set of actions available to player/agent i. Each vector $a = (a1, ..., an) \in A$ is called an action profile (or joint action).
    - $u = (u1, ..., un)$: Where $ui: A \longmapsto \mathbb{R}$ is a real-valued utility (or payoff) function for player i.
- **Summary**: Each agent chooses a single action and receives a payoff depending on the joint action. Payoffs are common knowledge, but agents don't know others' actions. Agents predict others' actions.
- **Notation in Economics**:
    - n-player game.
    - $si \in Si$: Strategy for player i, where Si is the strategy space.
    - $(s1, ..., sn)$: Combination of strategies.
    - $ui(s1, ..., sn)$: Player i's payoff function.
    - Game denoted by $G = \{S1, ..., Sn, u1, ..., un\}$.

4.3 Example – Prisoner’s Dilemma

- One of the oldest and most studied models in game theory.
- **Scenario**: Two suspects arrested, police lack evidence unless one confesses. Held separately.
    - If neither confesses: 1 month jail each.
    - If both confess: 6 months jail each.
    - If one confesses, other doesn't: Confessor released, other gets 9 months jail.
- **Elements of this normal-form game**:
    - $N = \{Prisoner\ 1, Prisoner\ 2\}$.
    - $A1 = A2 = \{not\ confess, confess\}$.
    - $a = (a1, a2)$: Joint action (e.g., $(confess, not\ confess)$).
    - $u = (u1(a1, a2), u2(a1, a2))$: Payoff function for each agent.
- **Payoff Matrix (Example)**: | Prisoner 2 | Not confess | Confess | |---|---|---| | **Prisoner 1 Not confess** | -1, -1 | -9, 0 | | **Prisoner 1 Confess** | 0, -9 | -6, -6 |
- **General Form**: Any normal-form game with the following payoff structure, where $c > a > d > b$: | Agent 2 | Cooperate | Defect | |---|---|---| | **Agent 1 Cooperate** | a, a | b, c | | **Agent 1 Defect** | c, b | d, d |

4.4 Strictly Dominated Action

- **Assumption**: A rational agent will never choose a suboptimal action (or play a strictly dominated action/strategy).
- **Definition**: An action `ai` of agent `i` is strictly dominated by another action `ai'` of agent `i` if `ui(ai', a-i) > ui(ai, a-i)` for all actions `a-i` of the other agents.
- **Prisoner's Dilemma Example**: For Prisoner 1, `not confess` is strictly dominated by `confess`.
    - `u1(confess, not confess) > u1(not confess, not confess)` (0 > -1)
    - `u1(confess, confess) > u1(not confess, confess)` (-6 > -9)
    - Similarly for Prisoner 2.

4.5 Solution Concepts

- Methods to reason about games and predict outcomes/actions.

4.6 Iterated Elimination of Strictly Dominated Actions

- **First solution concept**: Based on the assumption that a rational agent will never choose a suboptimal action.
- **Technique**: Iteratively eliminates strictly dominated actions from all agents until no more actions are strictly dominated.
- **Assumptions**:
    1. A rational agent would never take a strictly dominated action.
    2. It is common knowledge that all agents are rational.
- **Example (Game 2)**: | Agent 2 | Left | Middle | Right | |---|---|---|---| | **Agent 1 Up** | 1, 0 | 1, 2 | 0, 1 | | **Agent 1 Down** | 0, 3 | 0, 1 | 2, 0 |
    - **Process**: Right is strictly dominated by Middle for Agent 2. Eliminating Right, Down becomes strictly dominated by Up for Agent 1. Eliminating Down, Left becomes strictly dominated by Middle for Agent 2. Solution: (Up, Middle).
- **Prisoner's Dilemma Example**: Iterated elimination leads to (Confess, Confess).
- **Order of elimination**: Does not matter for strictly dominated actions; the outcome will be the same.
- **Drawbacks**:
    1. Each step requires further assumptions about common knowledge of rationality.
    2. Often produces imprecise predictions (e.g., if no actions are eliminated, anything could happen).
    3. Some games may not have any strictly dominated actions, making the technique unhelpful.

4.7 Nash Equilibrium

- A stronger solution concept than iterated elimination of strictly dominated actions, producing more accurate predictions.

4.8 Exercises (Normal-Form Games)

This section outlines exercises related to Normal-Form Games, focusing on the Iterated Elimination of Strictly Dominated Actions (IESDA) and Nash Equilibrium (NE).

Exercise 1 (IESDA):

- **Scenario**: A game with two agents (Agent 1 and Agent 2) and their respective actions and payoffs.
- **Task**: Predict the outcome using the iterated elimination of strictly dominated actions.
- **Payoff Matrix**: | Agent 2 | Left | Right | Stay | |---|---|---|---| | **Agent 1 Up** | 1, 0 | 1, 2 | 0, 1 | | **Agent 1 Down** | 0, 3 | 0, 1 | 1, 2 | | **Agent 1 Stay** | 2, 4 | 2, 1 | 2, 3 |

Exercise 2 (Advertising Scenario - IESDA):

- **Scenario**: Two companies share a market, each making $5,000,000. Advertising costs $2,000,000 and captures $3,000,000 from the competitor if the competitor doesn't advertise.
- **Tasks**: Identify the number of agents, action sets, payoffs, and predict the outcome using IESDA.

Exercise 1 (Nash Equilibrium):

- **Scenario**: A game with two agents (Agent 1 and Agent 2) and their respective actions and payoffs.
- **Task**: Predict the outcome using the Nash Equilibrium definition.
- **Payoff Matrix**: | Agent 2 | Left | Right | Stay | |---|---|---|---| | **Agent 1 Up** | 1, 0 | 3, 2 | 0, 1 | | **Agent 1 Down** | 0, 3 | 0, 1 | 1, 2 | | **Agent 1 Stay** | -1, 4 | 2, 1 | 2, 3 |

Exercise 2 (Duopoly Scenario - Nash Equilibrium):

- **Scenario**: Two firms (Firm 1 and Firm 2) control the production of an identical good. Each firm spends $1 to produce a unit. Consumer demand determines the price `P = 12 - 2Q`, where `Q = Q1 + Q2` (total quantity). If `Q >= 6`, then `P = 0`. Each firm has six plausible production choices: 0, 1, 2, 3, 4, 5. Profit of firm `i` is `(P - 1)Qi`.
- **Tasks**: Identify the number of agents, action sets, payoffs, and predict the outcome using the Nash Equilibrium definition.

5. Rational Agents

5.1 Rational Agents and Decision Making

- **Rationality**: An agent's ability to act (i.e., make decisions) in a way that maximizes some utility function.
- **Key Questions**: What is a utility function? How can it be used for decision making? How do humans make decisions?
- **Decision Examples**:
  - **Lottery**: Receive prize today vs. next month (straightforward).
  - **Plane Ticket**: Rerouting + cash vs. keeping ticket (depends on preferences).
  - **Vacation**: Hawaii vs. Cancun (depends on preferences).
- **Conclusion**: Many decisions are based on personal preferences. Key questions for agents are how to convey preferences and if decision making can be algorithmic.

5.2 Utility Theory for Decision Making

5.2.1 Binary Relations

- A binary relation R on a set of outcomes X is a set of ordered pairs (x, y) where $x, y \in X$.
- **Example**: "is shorter than" relation. If John (x) is 1.75m and Harry (y) is 1.85m, then $x \ R \ y$ (John is shorter than Harry).
- **Properties of Binary Relations**:
  - **Reflexive**: $x \ R \ x$ (e.g., "is equal to").
  - **Irreflexive**: $\neg(x \ R \ x)$ (e.g., "is shorter than").
  - **Symmetric**: $x \ R \ y \Rightarrow y \ R \ x$ (e.g., "is a sibling of").
  - **Asymmetric**: $x \ R \ y \Rightarrow \neg(y \ R \ x)$ (e.g., "is shorter than").
  - **Antisymmetric**: $x \ R \ y \land y \ R \ x \Rightarrow x = y$ (e.g., "is less than or equal to").
  - **Transitive**: $x \ R \ y \land y \ R \ z \Rightarrow x \ R \ z$ (e.g., "is shorter than").
  - **Negatively transitive**: $\neg(x \ R \ y) \land \neg(y \ R \ z) \Rightarrow \neg(x \ R \ z)$.
  - **Connected or Complete**: For all x, y, $x \ R \ y$ or $y \ R \ x$.
  - **Weakly connected**.

5.2.2 Preferences

- **Strict Preference ($\succ$)**: $x \succ y$ denotes that x is preferred to y (or x is better than y).
- **Indifference ($\sim$)**: $x \sim y$ denotes that the two outcomes are indifferent (or x is neither better nor worse than y). Can arise from no perceived difference or uncertainty.
- **Preference-Indifference ($\succcurlyeq$)**: $x \succcurlyeq y$ denotes x is not worse than y (union of strict preference and indifference).
- **Properties of Preferences**:
  - **Strict preference**: antisymmetric, transitive, and negatively transitive.
  - **Indifference**: reflexive, symmetric, and transitive.
  - **Preference-indifference**: complete and transitive.
- **Rational Preference**: A binary relation that is complete and transitive. Thus, preference-indifference is a rational preference.

5.2.3 Utility

- **Why use utility instead of preferences?**: Preferences are cumbersome to maintain computationally.
- **Concept**: Preferences express an ordering between outcomes. This ordering can be expressed with an order-preserving function.
- **Existence**: An order-preserving function (utility function) exists if preferences are rational, allowing consistent sorting of outcomes.
- **Theorem**: If preferences are rational, there exists a utility function u such that $x \succ y \Leftrightarrow u(x) > u(y)$ and $x \sim y \Leftrightarrow u(x) = u(y)$.

5.3 Making Decisions

- **Under Certainty**: An agent selects an action a from a set of actions A that maximizes its utility. $Q(a) = u(O\_a)$, where O_a is the outcome of action a. The agent chooses $\text{argmax\_a} \in A \ Q(a)$ or $\text{argmax\_a} \in A \ u(O\_a)$.
- **Under Uncertainty**: An agent selects an action that maximizes its *expected* utility.
  - O: Finite set of outcomes.

- `P(o|a)`: Probability of outcome `o` when action `a` is selected.
- **Expected Value of an Action**: `Q(a) = E[u(o_a)] = Σ_o∈O u(o)P(o|a).`
- The agent chooses `argmax_a∈A Q(a)` or `argmax_a∈A Σ_o∈O u(o)P(o|a).`

5.4 Example: Robot Coffee Machine

- **Outcomes (O)**: {"coffee + mess", "coffee + no mess", "no coffee + no mess"}.
- **Actions (A)**: {"get coffee", "do nothing"}.
- **Probabilities (P)**:
  - `a = "get coffee"`:
    - `P("coffee + mess" | "get coffee") = 0.2`
    - `P("coffee + no mess" | "get coffee") = 0.8`
    - `P("no coffee + no mess" | "get coffee") = 0`
  - `a = "do nothing"`:
    - `P("coffee + mess" | "do nothing") = 0`
    - `P("coffee + no mess" | "do nothing") = 0`
    - `P("no coffee + no mess" | "do nothing") = 1.0`
- **Utility Function (u)**:
  - `u("coffee + mess") = 5`
  - `u("coffee + no mess") = 10`
  - `u("no coffee + no mess") = 0`
- **Decision Tree (Expected Utility Calculation)**:
  - For `a = "get coffee"`: `(0.2 * 5) + (0.8 * 10) + (0 * 0) = 1 + 8 + 0 = 9`.
  - For `a = "do nothing"`: `(0 * 5) + (0 * 10) + (1.0 * 0) = 0 + 0 + 0 = 0`.
- **Conclusion**: The best action is "get coffee" as it yields the maximum expected utility (9).

5.5 Final Remarks

- This section considered decision-making problems with only one agent. When there are two or more utility-maximizing agents whose actions affect each other, a different framework is needed: **Game Theory**.

4.9 Application of a Nash Equilibrium: Duopoly (Cournot Model)

- **Origin**: Created by Antoine Augustin Cournot in 1838, a century before John Nash.
- **Focus**: Analysis of an oligopoly, specifically a duopoly (2 firms).
- **Features**:
  - More than one firm (fixed number).
  - Firms produce a homogeneous product (no differentiation).
  - Firms do not cooperate (no collusion).
  - Firms have market power (each firm's output affects market price).
  - Firms choose quantities simultaneously (one-shot game).
  - Firms compete in quantities.

    - Firms are economically rational and act strategically to maximize profit.
- **Model Setup**:
    - `q1`, `q2`: Quantities produced by firms 1 and 2.
    - `P(Q) = a - Q`: Market-clearing price, where `Q = q1 + q2` (aggregate quantity).
        - `P(Q) = a - Q` for `Q < a`.
        - `P(Q) = 0` for `Q ≥ a`.
    - **Cost**: `Ci(qi) = cqi` (no fixed costs, constant marginal cost `c`, where `c < a`).
- **Normal-form game elements**:
    - **Agents**: Two firms.
    - **Action Sets**: `Ai = [0, ∞)` (nonnegative real numbers), representing quantities `qi`.
    - **Payoffs (Profit)**: `πi(qi, qj) = qi * P(qi + qj) - c*qi = qi * (a - (qi + qj) - c)`.
- **Finding Nash Equilibrium**:
    - The quantity pair `(q1*, q2*)` is a Nash equilibrium if, for each firm `i`, `qi*` solves `max(qi) πi(qi, qj*)`.
    - This involves taking the partial derivative of the profit function with respect to `qi` and setting it to zero.
    - **Best Response Functions**:
        - `b1(q2) = (a - q2 - c) / 2`
        - `b2(q1) = (a - q1 - c) / 2`
    - **Nash Equilibrium Solution**: `q1* = q2* = (a - c) / 3` (found at the intersection of the best response functions).

4.10 Mixed Strategy

- **Motivation**: Some games (e.g., Matching Pennies) do not have a pure strategy Nash Equilibrium because agents want to outguess each other.
- **Matching Pennies Game**:
    - **Agents**: Two.
    - **Action Set**: `{Heads, Tails}` for each agent.
    - **Payoffs**: If pennies match (HH or TT), Agent 2 wins Agent 1's penny (-1, 1). If they don't match (HT or TH), Agent 1 wins Agent 2's penny (1, -1).
    - **Payoff Matrix**: | Agent 2 | Heads | Tails | |---|---|---| | **Agent 1 Heads** | -1, 1 | 1, -1 | | **Agent 1 Tails** | 1, -1 | -1, 1 |
    - This is a zero-sum game.
- **Definition**: A mixed strategy for an agent `i` is a probability distribution over his actions `ai ∈ Ai` (pure strategies).
    - For Matching Pennies, a mixed strategy for agent `i` is `(q, 1-q)`, where `q` is the probability of `Heads` and `1-q` is the probability of `Tails`.
- **Extended Nash Equilibrium Definition**: Each agent's mixed strategy must be a best response to the other agents' mixed strategies.
    - This subsumes pure strategy NE, as pure strategies are mixed strategies with probability 1 on one action.
    - Computing best response to a mixed strategy involves interpreting the opponent's mixed strategy as uncertainty.
- **Mixed Strategy Nash Equilibrium in Matching Pennies**:

    - Let Agent 1 believe Agent 2 plays `(q, 1-q)`.
        - `EU1(Heads) = q(-1) + (1-q)(1) = 1 - 2q`
        - `EU1(Tails) = q(1) + (1-q)(-1) = 2q - 1`
    - Agent 1 is indifferent between Heads and Tails when `EU1(Heads) = EU1(Tails)`, which means `1 - 2q = 2q - 1`, so `q = 1/2`.
    - Similarly, for Agent 2, if Agent 1 plays `(r, 1-r)`, Agent 2 is indifferent when `r = 1/2`.
    - **Mixed Strategy NE**: Both agents play `(1/2, 1/2)`.
- **Meaning of Mixed Strategy**:
    - **Randomize to confuse opponent**: As in Matching Pennies, where an equilibrium only exists if there's uncertainty.
    - **Randomize when uncertain about other's action**: As in Battle of the Sexes.
    - **Concise description of repeated play**: Mixed strategies can represent the limiting behavior of pure strategies in repeated games.
    - **Population dynamics**: Mixed strategies can describe the probability of getting each pure strategy when agents are chosen from a population.

4.11 Exercise (Mixed Strategy)

- **Battle of the Sexes**:
    - **Scenario**: A man and woman want to go out, but have no communication. Man prefers fight, woman prefers ballet. Both prefer being together than alone.
    - **Tasks**: Identify agents, action sets, payoffs. Determine if pure strategy NE exists. Determine if mixed strategy NE exists.

6. Multiagent Coordination

6.1 Introduction to Multiagent Coordination

- **Definition**: Managing the interdependencies between activities.
- **Examples**: Agents needing to coordinate to use a non-sharable resource, or two soccer robots deciding who should go for the ball.
- **Relationship with Cooperation**: Coordination is relevant for multiagent cooperation. Coordination mechanisms ensure agents do not obstruct each other and efficiently perform joint actions to serve a common goal.
- **Informal Definition**: Coordination can be regarded as the process where every agent’s individual decision leads to a good joint action for the group.

6.2 Coordination Games

- Coordination problems can be modeled as coordination games using game theory (normal-form representation, action sets, payoffs, Nash equilibrium).
- **Stag Hunt Game (Example)**:
    - **Scenario**: Two hunters. If both hunt hares, they share. If one hunts stag and other hares, stag hunter gets nothing, hare hunter gets all hares. If both hunt stag, they share the stag, which is more valuable than all

hares.
    - **Payoff Matrix**: | Hunter 2 | Stag | Hare | |---|---|---| | **Hunter 1 Stag** | 3, 3 | 0, 2 | | **Hunter 1 Hare** | 2, 0 | 1, 1 |
    - **Nash Equilibria**: (Stag, Stag) and (Hare, Hare).
    - **Coordination Problem**: Which equilibrium should they choose?
        - Hunting Stag is risk-taking (highest payoff if coordinated, lowest if not).
        - Hunting Hare is conservative (guaranteed payoff, but not highest).
- **Coordination with Agreement**: If hunters agree in advance, they might choose a Pareto optimal joint action.
    - **Pareto Dominance**: Joint action a Pareto dominates a' if for all $i \in N$, $u_i(a) \geq u_i(a')$, and there exists some $j \in N$ for which $u_j(a) > u_j(a')$.
    - **Pareto Optimal (Strictly Pareto Efficient)**: A joint action a is Pareto optimal if no other joint action a' Pareto dominates a.
    - In Stag Hunt, (Stag, Stag) is Pareto optimal.
- **Formal Definition of Coordination**: The process in which a group of agents choose a single Pareto optimal Nash equilibrium in a game.
- **Autonomous Vehicles at a Crossroad (Example)**:
    - **Scenario**: Two cars want to cross first, but will crash if both cross simultaneously.
    - **Payoff Matrix**: | Car 2 | Cross | Stop | |---|---|---| | **Car 1 Cross** | -1, -1 | 1, 0 | | **Car 1 Stop** | 0, 1 | 0, 0 |
    - **Nash Equilibria**: (Cross, Stop) and (Stop, Cross). Both are Pareto optimal.
- **Robot Soccer (Teamwork Example)**:
    - **Scenario**: Two robots want to get the ball, but will crash if both try simultaneously.
    - **Payoff Matrix**: | Robot 2 | Run | Stay | |---|---|---| | **Robot 1 Run** | -1, -1 | 1, 1 | | **Robot 1 Stay** | 1, 1 | 0, 0 |
    - **Pure Coordination Games (Team Games)**: All agents in the team share the same payoff function ($u_1(a) = \ldots = u_n(a) \equiv u(a)$).
    - **Nash Equilibria**: (Run, Stay) and (Stay, Run). Both are Pareto optimal.

6.3 Social Conventions

- **Purpose**: To solve coordination problems by providing a recipe for agents to choose the same Nash equilibrium.
- **Definition**: A social convention (or social law) places constraints on agents' action choices. It dictates how agents should choose actions to reach an equilibrium. Once established and common knowledge, no agent benefits from not abiding by it.
- **Boutilier (1996) Proposal**: A general convention assuming a unique ordering scheme of joint actions that is common knowledge.
    - Agents compute all equilibria, then select the first equilibrium according to this ordering scheme.
- **Movie Example**:
    - **Scenario**: Two agents want to go to movies together (Thriller or Comedy).
    - **Payoff Matrix**: | Agent 2 | Thriller | Comedy | |---|---|---| | **Agent 1 Thriller** | 1, 1 | 0, 0 | | **Agent 1 Comedy** | 0, 0 | 1, 1 |
    - **Nash Equilibria**: (Thriller, Thriller) and (Comedy, Comedy).
    - **Ordering Scheme**: Agent 1 $\succ$ Agent 2; Thriller $\succ$ Comedy.
    - **Result**: (Thriller, Thriller) is chosen unanimously.
- **Car Coordination Game Example (with Social Convention)**:

- **Ordering Scheme**: Driver coming from the right has priority. Car 2 has priority over Car 1. Actions: Cross $\succ$ Stop.
- **Result**: (Stop, Cross) is chosen unanimously.

6.4 Social Conventions with Communication

- **When communication is available**: Only an ordering of agents is needed, and this ordering must be common knowledge.
- **Algorithm**:
  1. Each agent `i` (except agent 1) waits until previous agents `1, ..., i-1` have broadcast their chosen actions.
  2. Agent `i` computes its component `ai*` of an equilibrium consistent with previous choices.
  3. Agent `i` broadcasts `ai*` to all agents that haven't chosen an action yet.
- **Car Coordination Game Example (with Communication)**:
  - **Ordering**: (Car 1, Car 2).
  - Car 1 chooses `Cross` and broadcasts it.
  - Car 2 receives `Cross` from Car 1, computes its best response (`Stop`), and the joint action is (Cross, Stop).

6.5 Coordination Games with 3 Agents (Crossroad Example)

- **Scenario**: Three autonomous vehicles at a crossroad. Each wants to cross first, but if two cross, they crash.
- **Joint Actions and Payoffs**: A table lists payoffs for various combinations of (cross, stop) actions for three cars.
- **Best Response Functions**: Calculations are shown for each car based on the actions of the other two cars.

6.6 Roles

(Content truncated due to size limit. This section would contain further details on roles in coordination.)

6.7 Roles with Communication

(Content truncated due to size limit. This section would contain further details on roles with communication.)

6.8 Coordination Graphs

(Content truncated due to size limit. This section would contain further details on coordination graphs.)

7. Multiagent Decision Making and Games in Extensive Form

7.1 Extensive-Form Games

- **Contrast with Normal-Form Games**: Normal-form games do not incorporate sequence or time and assume simultaneous actions (one-shot games).
- **Extensive-Form Games (Tree-Form Games)**: An alternative representation that makes the temporal structure explicit.
- **Perfect-Information Extensive-Form Games (Finite Games)**:
    - Represented as a tree (in graph theory sense).
    - Each node represents an agent's choice.
    - Each edge represents an agent's action.
    - Leaves represent final outcomes (payoffs/utility).
- **Example: The Sharing Game**:
    - **Scenario**: Brother and sister share two indivisible, identical presents. Brother suggests a split (2-0, 1-1, or 0-2). Sister accepts or rejects. If accepted, they get allocated presents; otherwise, neither gets gifts. Both value presents equally and additively.
    - **Game Tree Description**: A tree where the root is the Brother (Player 1) choosing a split (2-0, 1-1, 0-2). From each of these choices, the Sister (Player 2) decides to accept (yes) or reject (no). Terminal nodes show payoffs (e.g., (2,0) for Brother gets 2, Sister gets 0).
- **Definition (Perfect-Information Game)**: A tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ where:
    - N: Set of n agents.
    - A: Single set of actions.
    - H: Set of nonterminal choice nodes.
    - Z: Set of terminal nodes, disjoint from H.
    - $\chi: H \longmapsto 2^A$: Action function, assigns a set of possible actions to each choice node.
    - $\rho: H \longmapsto N$: Player function, assigns to each nonterminal node a player $i \in N$ who chooses an action at that node.
    - $\sigma: H \times A \longmapsto H \cup Z$: Successor function, maps a choice node and an action to a new choice node or terminal node. If $\sigma(h1, a1) = \sigma(h2, a2)$, then $h1 = h2$ and $a1 = a2$.
    - $u = (u1, ..., un): ui: Z \longmapsto \mathbb{R}$ is a real-valued utility function for player i on the terminal nodes Z.

7.2 Strategies and Equilibria

- **Pure Strategy**: For an agent in a perfect-information game, it is a complete specification of which deterministic action to take at every node belonging to that agent.
    - Formally: $\Pi(h \in H, \rho(h)=i)\ \chi(h)$.
- **Example: Sharing Game Pure Strategies**:
    - Brother (Player 1): $S1 = \{2\text{-}0, 1\text{-}1, 0\text{-}2\}$ (his initial choices).
    - Sister (Player 2): $S2 = \{(yes,yes,yes), (yes,yes,no), ..., (no,no,no)\}$ (her response to each of Brother's choices).
- **Computing Nash Equilibria**: The definitions of best response and Nash equilibria are the same as for normal-form games.
- **Conversion to Normal-Form**: A perfect-information game can be converted to an equivalent normal-form game.
    - **Note**: The temporal structure of the extensive-form game can create redundancy in the normal-form game (e.g., 16 outcomes in normal form vs. 5 in extensive form for an example game).
- **Theorem**: Every (finite) perfect-information game in extensive form has a pure-strategy Nash equilibrium.

    - **Intuition**: Since agents take turns and have full information, randomness is not needed to find an equilibrium.
- **Example Game Nash Equilibria**: For a given game tree, Nash equilibria might be `{(A, G), (C, F )}`, `{(A, H), (C, F )}`, and `{(B, H), (C, E)}`.

7.3 Subgame-Perfect Equilibrium (SPE)

- **Issue with Normal-Form Conversion**: Examining the converted normal-form game obscures the temporal nature and can lead to

  equilibria that are not credible (e.g., threats).

- **Example of Non-Credible Threat**: In an example game, a Nash equilibrium `{(B, H), (C, E)}` might involve Agent 1 threatening to play `H` if Agent 2 plays `F`, even though playing `H` would be suboptimal for Agent 1 in that subgame. If Agent 2 does not consider Agent 1's threat credible, this NE is problematic.
- **Subgame**: Given a perfect-information extensive-form game `G`, the subgame of `G` rooted at node `h` is the restriction of `G` to the descendants of `h`.
- **Subgame-Perfect Equilibrium (SPE)**: The SPE of a game `G` are all strategy profiles `s` such that for any subgame `G'` of `G`, the restriction of `s` to `G'` is a Nash equilibrium of `G'`.
- **Relationship between SPE and NE**:
    - Every SPE is also a NE.
    - Not every NE is a SPE.
    - Every perfect-information extensive-form game has at least one SPE.

7.4 Backward Induction

- **Purpose**: A procedure to compute the subgame-perfect equilibrium.
- **General Idea**: Identify equilibria in the "bottom-most" subgame trees, assume these equilibria will be played, and then back up to consider increasingly larger trees.
- **Process**: This procedure is a single depth-first traversal of the game tree, requiring time linear in the size of the game representation.
    - It returns the payoff when it reaches a terminal node (end of recursion).
    - It keeps track of the best payoff of a node.
    - It iterates over all actions of a node.
    - It recursively calls the function for a child node.
    - If the agent's payoff in the child node is greater than the agent's payoff in `best_util`, then `best_util` is updated.
- **Example**: Applying backward induction to the example game reveals that `{(A,G),(C,F)}` is a subgame-perfect equilibrium.

7.5 Imperfect-Information Games in Extensive Form

- **Motivation**: Perfect-information games assume agents know the exact node they are at and all prior choices. Imperfect-information games address situations where agents act with partial or no knowledge of other agents'

actions, or even with limited memory of their own past actions.

- **Examples**: Poker, Cribbage, and other games involving hidden actions.
- **Definition**: An extensive-form game where each agent's choice nodes are partitioned into **information sets**.
    - **Information Set**: If two choice nodes are in the same information set, the agent cannot distinguish between them. The set of actions available at each choice node within an information set must be the same.
    - Formally: $G = (N, A, H, Z, \chi, \rho, \sigma, u, I)$, where I is a set of equivalence classes on the choice nodes for each agent.
- **Example Game Tree Description**: A game tree where Agent 1 (Player 1) makes a choice, then Agent 2 (Player 2) makes a choice, and then Agent 1 makes another choice, but Agent 1 does not know what Agent 2 chose when making the second choice (represented by a dashed line connecting two nodes for Agent 1).

7.6 Strategies and Equilibria in Imperfect-Information Games

- **Pure Strategy**: For an agent in an imperfect-information game, it is one of the available actions in each information set of that agent.
    - Formally: $\Pi(I_{i,j} \in I_i)\ \chi(I_{i,j})$.
- **Nash Equilibrium**: The definition of best response and Nash Equilibrium (both pure and mixed) remains the same for imperfect-information extensive-form games.
- **Conversion to Normal-Form**: An imperfect-information game can be converted to an equivalent normal-form game to compute Nash equilibria.
    - **Example**: A game tree with imperfect information can be converted to a normal-form matrix. Nash equilibria are then found in this matrix.
- **Mapping between Game Forms**:
    - Imperfect-information extensive-form game -> Normal-form game.
    - Normal-form game -> Imperfect-information extensive-form game.
    - Applying these mappings in turn might not result in the exact same game, but it will result in strategically equivalent games (same strategy space and equilibria).

7.7 Mixed and Behavioral Strategies

- **Two types of randomized strategies**:
    - **Mixed Strategies**: Randomize over pure strategies.
        - Definition: A mixed strategy si is any probability distribution over agent i's pure strategies.
    - **Behavioral Strategies**: Randomize every time an information set is encountered.
        - Definition: A behavioral strategy bi is a mapping from an agent's information sets to a distribution over the actions at that information set, sampled independently each time the agent arrives at the information set.
- **Equivalence in Perfect-Information Games**: In perfect-information games, mixed strategies and behavioral strategies can emulate each other (Kuhn, 1953).
- **Equivalence in Imperfect-Information Games**: This also applies to imperfect-information games if they have **perfect recall**.
    - **Perfect Recall**: Agents have full recollection of their experience in the game, meaning they know all information sets they visited and all actions they have taken.

    - **Kuhn (1953) Theorem**: In a game of perfect recall, any mixed strategy can be replaced by an equivalent behavioral strategy, and vice versa.
- **Game of Imperfect Recall (Example)**:
    - A game where an agent cannot remember where they started or previous actions.
    - In such games, mixed strategy equilibria and behavioral strategy equilibria can be different.
    - **Example Calculation**: For a game of imperfect recall, calculating the mixed strategy Nash equilibrium and a behavioral strategy equilibrium can yield different results, demonstrating that the equivalence does not hold without perfect recall.

8. Multiagent Decision Making: Repeated Games

8.1 Introduction to Repeated Games

- **Concept**: Playing the same normal-form game (called the **stage game**) over and over.
- **Key Questions**:
    - Can agents observe other agents' actions?
    - Can agents remember the past?
    - What is the agent's utility for the whole game?
- **Types of Repeated Games**:
    - Finitely-repeated games.
    - Infinitely-repeated games.

8.2 Finitely-Repeated Games

- **Modeling**: Can be modeled as an extensive-form game with imperfect information.
    - At each round, players don't know what others have done, but they do afterward.
- **Overall Payoff Function**: Additive (sum of payoffs in stage games).
- **Example: Prisoner's Dilemma in a Two-Stage Game**:
    - The game tree shows choices (C1, D1 for stage 1; C1,1, D1,1, etc. for stage 2 based on stage 1 outcomes).
    - Pure strategies become very complex (e.g., for a 2-stage game, Agent 1 has 16 pure strategies).
    - **Nash Equilibrium**: The strategy profile where both agents defect in every stage (e.g., (D1,D1,4), (D1,D2,4) for the 2-stage PD) is a Nash equilibrium.
- **Proposition**: If the stage game `G` has a unique Nash equilibrium, then for any finite `T`, the repeated game `G(T)` has a unique outcome: the Nash equilibrium of `G` is played in every subgame-perfect stage.

8.3 Infinitely-Repeated Games

- **Challenge**: Cannot be represented as a finite extensive-form game (infinite tree, infinite sum of payoffs).
- **Payoff Measures**:
    - **Average Reward**: `lim(k→∞) (1/k) Σ(j=1 to k) rj.`

    - **Future Discounted Reward**: $\Sigma_{j=1 \text{ to } \infty} \beta^j * r_j$, where $0 < \beta < 1$ is the discount factor. This implies agents care more about near-term well-being.
- **Pure Strategy**: A choice of action at every decision point, meaning an action at every stage game for an infinite number of actions.
    - **Histories**: $H_t = \{h_t: h_t = (a_1, ..., a_t) \in A_t\}$. All finite histories: $H = \cup H_t$.
    - **Pure Strategy Definition**: $s_i: H \rightarrow A_i$.
- **Example: Repeated Prisoner's Dilemma Strategies**:
    - **Tit-for-tat**: Start cooperating. If opponent defects, defect in next round, then return to cooperation.
    - **Trigger**: Start cooperating. If opponent ever defects, defect forever.
- **Subgame-Perfect Equilibrium**: A profile of strategies that are a Nash equilibrium in every subgame (i.e., following every possible history).
    - Repeatedly playing a Nash equilibrium of the stage game is always a subgame-perfect equilibrium of the repeated game.

8.4 Folk Theorem

- **Concept**: A collection of theorems for infinitely-repeated games concerning Nash equilibria. Widely known among game theorists before formal publication.
- **Main Idea**: Consider a finite normal form game G (stage game). Let $a = (a_1, ..., a_n)$ be a Nash equilibrium of G. If $a' = (a'_1, ..., a'_n)$ is such that $u_i(a') > u_i(a)$ for all i, then there exists a discount factor β ($0 < \beta < 1$) such that $\beta_i \geq \beta$ for all i, and there exists a subgame-perfect equilibrium of the infinite repetition of G that has a' played in every period on the equilibrium path.
- **Outline of Proof (using Trigger strategy)**:
    - Play a' as long as everyone is playing a'.
    - If any agent deviates, play a (the stage game NE) forever after.
    - This is a subgame-perfect equilibrium for high enough discount factors.
    - **Condition for no deviation**: $\beta_i / (1 - \beta_i) * M - m <= 0$, where M is max gain from deviating and m is min per-period loss from future punishment. This simplifies to $\beta_i >= M / (M + m)$.
- **Example: Prisoner's Dilemma - Can we sustain cooperation?**
    - If agents always cooperate, the payoff is $3 / (1 - \beta)$.
    - If an agent defects once and then others play the NE (defect forever), the payoff is $5 + \beta / (1 - \beta)$.
    - To sustain cooperation, the payoff from cooperating must be greater than or equal to the payoff from defecting: $3 / (1 - \beta) >= 5 + \beta / (1 - \beta)$. This simplifies to $\beta >= 1/2$.
    - **Interpretation**: To sustain cooperation, agents must care about future payoffs at least half as much as current payoffs.

8.5 Evolutionary Learning and Other Large-Population Models

- **Focus**: Modeling the evolution of behavior in large populations (e.g., in biology, social networks, human populations).
- **Replicator Dynamics**: Models a population undergoing frequent replicator interactions.
    - **Symmetric Game**: Basic interaction between any two agents, where agents have no distinct roles.

    - **$\theta_t(A)$**: Ratio of agents playing strategy A at time t.
    - **$u_t(a)$**: Expected utility of strategy a at time t.
    - **$\dot{\theta}_t(A)$**: Rate of change of the ratio of agents playing strategy A. This change is proportional to the difference between the strategy's utility and the average utility of the population.
- **Steady State**: A population state θ where $\dot{\theta}(a) = 0$ for all a.
- **Stable Steady State**: A steady state θ is stable if small perturbations in the population state lead back to θ.
- **Asymptotically Stable State**: A steady state θ is asymptotically stable if it is stable and all nearby population states converge to θ over time.
- **Relationship with Nash Equilibrium**:
    - If (S, S) is a symmetric mixed strategy Nash equilibrium, then $\theta = (S(a1), ..., S(ak))$ is a steady state of the replicator dynamic.
    - If θ is a stable steady state of the replicator dynamic, then (S, S) is a mixed strategy Nash equilibrium.
    - If θ is an asymptotically stable steady state, then (S, S) is a Nash equilibrium that is trembling-hand perfect and isolated.
- **Evolutionarily Stable Strategies (ESS)**: A static solution concept that does not require replicator dynamics. A mixed strategy S is a weak ESS if it is "resistant to invasion" by new strategies.

9. Markov Decision Process (MDP)

9.1 Introduction to MDP

- **Framework**: For sequential decision making of a single agent.
- **Characteristics**:
    - Markovian transition model and fully observable (verifies the Markov property).
    - Planning horizon can be infinite.
- **Elements of an MDP**:
    - **Discrete time**: $t = 0, 1, 2, ...$
    - **Discrete set of states**: $s \in S$
    - **Discrete set of actions**: $a \in A$
    - **Stochastic transition model**: $P(s'|s, a)$ (world transitions stochastically to state s' when agent takes action a at state s).
    - **Reward function**: $R: S \times A \to \mathbb{R}$ (agent receives reward $R(s, a)$ when taking action a at state s).

9.2 Value Functions

- **State-Value Function ($V^{\pi}(s)$)**: The expected return an agent can receive when starting in state s and then following policy π.
    - $V^{\pi}(s) = E[\Sigma(t=0 \text{ to } \infty) \gamma^t * R(s_t, a_t) \mid s_0 = s, a_t = \pi(s_t))]$
- **Action-Value Function (Q-values, $Q^{\pi}(s, a)$)**: The expected return an agent can receive when starting in state s, taking action a, and then following policy π.
    - $Q^{\pi}(s, a) = E[\Sigma(t=0 \text{ to } \infty) \gamma^t * R(s_t, a_t) \mid s_0 = s, a_0 = a, a_{t>0} = \pi(s_t))]$
- **Optimal Policy (π^*)**:

    - A policy π is better than or equal to π' if $V^\pi(s) \geq V^{\pi'}(s)$ for all $s \in S$.
    - There is always at least one optimal policy (π^*) that is better than or equal to all other policies.
    - MDPs might have more than one optimal policy.
- **Optimal State-Value Function ($V^*(s)$)**: $V^*(s) = \max_\pi V^\pi(s)$ for all $s \in S$.
- **Optimal Action-Value Function ($Q^*(s, a)$)**: $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ for all $s \in S$ and $a \in A$.

9.3 Value Iteration

- **Purpose**: To compute the optimal Q^* and V^* when the stochastic transition model and reward function are known.
- **Process**:
    1. Initialize a state-value function (e.g., with zeros).
    2. Iteratively apply the Bellman equation:
        - $Q(s, a) := R(s, a) + \gamma * \Sigma_{s' \in S} P(s'|s, a) * V(s')$ for all s, a.
        - $V(s) := \max_{a \in A} Q(s, a)$ for all s.
    3. Repeat until V does not change significantly.
- **Convergence**: Value iteration converges to the optimal Q^* for any initialization.
- **Extracting Policy**: After computing Q^*, the optimal policy $\pi^*(s)$ is $\text{argmax}_{a \in A} Q^*(s, a)$.

9.4 Reinforcement Learning (Q-learning)

- **Purpose**: To compute the optimal policy when the stochastic transition model and reward function are *unknown*.
- **Interaction with Environment**: The agent interacts with the environment to collect data.
    - At each time step t:
        - Agent observes state s_t.
        - Agent takes action a_t.
        - Agent observes reward R_t and new state s_{t+1}.
    - **Data Point**: (s_t, a_t, R_t, s_{t+1}).
- **Agent's Goal**: Compute the optimal policy of the MDP using these data points.
- **Q-learning Algorithm**:
    1. Start with some estimate Q (e.g., initialized to 1 for all state-action pairs).
    2. Initialize current state s.
    3. Loop for each step:
        - Choose an action a (e.g., using ε-greedy exploration).
        - Take action a and observe next state s' and reward r.
        - Update Q estimate according to the Q-learning update rule: $Q(s, a) \leftarrow Q(s, a) + \alpha * [r + \gamma * \max_{a'} Q(s', a') - Q(s, a)]$
        - Set $s \leftarrow s'$.
- **Observations about Q-learning**:
    - **Bootstrap**: Update depends on previous estimate.
    - **Reward Propagation**: Update rule propagates reward information.
    - **Exploration**: To compute Q accurately, the algorithm must visit every state-action pair.

- **Example Walkthrough**: A step-by-step example demonstrates how Q-learning updates `Q` values over iterations, eventually converging to values close to the optimal `Q*` obtained by Value Iteration. The optimal policy is then extracted from the learned `Q` values.

10. Multiagent Learning

10.1 Stochastic Games

- **Concept**: A generalization of repeated games where agents repeatedly play several normal-form games from a collection.
- **Key Feature**: The particular game played at any given iteration depends probabilistically on the previous game played and on the actions taken by all agents in that game.
- **Definition (Stochastic Game / Markov Game)**: A tuple `(Q, N, A, P, r)` where:
  - `Q`: A finite set of games (states).
  - `N`: A finite set of `n` agents.
  - `A = A1 × ... × An`: Where `Ai` is a finite set of actions available to agent `i`.
  - `P: Q × A × Q ⟼ [0,1]`: Transition probability function. `P(q, a, q emp)` is the probability of transitioning from state `q` to state `q emp` after action profile `a`.
  - `R = r1, ..., rn`: Where `ri: Q × A ⟼ ℝ` is a real-valued payoff function for agent `i`.
- **Generalization**: A stochastic game is a generalization of:
  - **Repeated game**: Where `Q` has only one game (one state).
  - **Markov decision process (MDP)**: Where `N` has only one agent.

10.2 Learning in Markov Games: Approaches

- **Independent Learning**: Each agent learns separately, ignoring the presence of other agents and treating them as part of its environment. Does not model other agents or predict their actions.
  - **Flaw**: The (hypothetical) transition model `p(s'|s, ai)` of agent `i` may be continuously changing due to the policies of other learning agents. Q-learning convergence relies on a stationary underlying transition model.
  - **Practical Use**: Despite theoretical flaws, it has been employed with reported success. Can lead to individual policies that form a Nash equilibrium in single-state coordination problems, but the equilibrium may not be Pareto optimal.
  - **Algorithm**: Each agent `i` maintains its own `Qi(s, ai)` and updates it based on its own observed rewards and next states, treating the joint action of others as part of the environment.
- **Centralized Learning**: A central entity decides everything, even if there are several agents.
- **Joint-Action Learners (JAL)**: Agents are aware of others and learn and decide based on a given criterion (e.g., equilibria, worst-case).
  - **Challenge**: The Q-function is multidimensional (`Q(s, a1, ..., an)`), so the best value depends on the actions of others.
  - **Assumptions**: Need to make assumptions about other agents (e.g., they are doing their best, or improving the worst-case scenario).

    - **Standard Approach**: Each agent `i` maintains an individual value function `Vi(s)` and an individual Q-function `Qi(s, a)` (where `a` is the joint action).
    - **Value Iteration Generalization**: `Vi(s) := C(Q1(s, a), ..., Qn(s, a))` where `C` applies a solution concept (e.g., Nash equilibrium, minimax) to the strategic game formed by the Q-values.
    - **Q-learning Update**: When the transition model is unavailable, a coupled Q-learning update is used for each agent.
- **Agent-Modelling (JAL-AM)**: Agents are aware of others and learn and decide based on a model of the other agents.

10.3 Learning Criteria in Multi-Agent Systems

- **Criteria for Learning**: What should agents optimize for?
    - Assume others are doing their best.
    - Consider my preferences.
    - Find a stable equilibrium.
    - Improve the worst-case scenario.
    - Ignore others.
    - Try to predict others.

10.4 Goalkeeper Example (Illustrating Learning Approaches)

- **Scenario**: Player 1 (left) moves to cells (up, middle, bottom), winning if Player 2 is not on the cell. Player 2 (right) blocks Player 1 (moving up, down, or center).
- **Independent Learning**: Agents do not observe others. Each agent has its own `Q` function for its own state and actions (e.g., `Q(1, ↑)` for Player 1).
- **Full Observation (Distributed Learning)**: Agents observe others as part of the environment, without assuming intentionality. `Q` functions include the state of other agents (e.g., `Q((1,2), ↑)` for Player 1, where (1,2) is the joint state).
- **Joint Action-Learning**: Both agents have `Q` functions with the same shape, but each computes them individually. The `Q` function is multidimensional, taking joint actions as input (e.g., `Q((1,2), ↑ ↑)` for Player 1, where `↑ ↑` is the joint action).
    - **Challenges in JAL**: Choosing an action (e.g., with `ε-greedy`) is complex because the `Q` function is multidimensional. The `max Q(s', ?)` in the update rule also depends on the actions of all other agents, requiring assumptions about their behavior.

11. Multiagent Decision Making and Auctions

11.1 Introduction to Auctions

- **Definition**: A mechanism for allocating (normally scarce) resources among self-interested agents.

- **Widespread Use**: Selling art, privatizing public companies, trading stocks, selling used goods (eBay), procuring parts, etc.

11.2 Canonical Auctions

11.2.1 English Auction

- **Type**: Open-outcry ascending auction.
- **Process**:
    1. Auctioneer starts bidding at a reserve price.
    2. Bidders shout out ascending prices.
    3. The highest bidder has the standing bid.
    4. The standing bid wins if no competing bid challenges within a given time frame.
    5. Item is sold to the highest bidder at their submitted price.
- **Properties**:
    - Bidders know the number of bidders.
    - Bids are public.
    - Bidders can submit several bids.
    - Commonly used for art, antiques, wine.

11.2.2 Dutch Auction

- **Type**: Open-outcry descending auction.
- **Process**:
    1. Auctioneer starts a clock at a high asking price.
    2. Price lowers at each time step until a bidder accepts the current ask price.
    3. The first bidder to accept pays that price.
- **Properties**:
    - Bidders know the number of bidders.
    - Bid is public (only one bid submitted).
    - Commonly used for selling flowers, fresh produce, tobacco.

11.2.3 First-Price Sealed-Bid Auction

- **Type**: Sealed-bid auction.
- **Process**:
    1. All bidders submit sealed bids simultaneously.
    2. No bidder knows others' bids.
    3. The highest bidder wins and pays the submitted price.
- **Properties**:
    - Bidders know the number of bidders.
    - Bids are private.
    - Bidders can only submit one bid.
    - Commonly used for privatization of public companies, selling concessions.

11.2.4 Second-Price Sealed-Bid Auction (Vickrey Auction)

- **Type**: Sealed-bid auction.
- **Process**:
    1. All bidders submit sealed bids simultaneously.
    2. No bidder knows others' bids.
    3. The highest bidder wins and pays the second highest bid.
- **Properties**:
    - Bidders know the number of bidders.
    - Bids are private.
    - Bidders can only submit one bid.
    - Bidders can be invited/selected.
    - Used in digital ads (Google, Facebook).
    - Has very interesting theoretical results.

11.3 Bidding in First-Price Auctions

- **Incentive**: An agent has an incentive to bid less than its true valuation to maximize profit.
- **Tradeoff**: Between probability of winning (higher bid = higher probability) and amount paid (lower bid = higher profit).
- **Dominant Strategy**: Bidders do not have a dominant strategy; their strategy depends on others.
- **Theorem (for 2 risk-neutral bidders, valuations i.i.d. from U[0,1])**: Agent 1 bids $(1/2)v1$, Agent 2 bids $(1/2)v2$. This is a Bayesian-Nash equilibrium.
    - **Proof Outline**: Maximize expected profit $E[u1] = P(win|b1)(v1 - b1)$. For $b2 = (1/2)v2$, $P(win|b1) = P(b2 < b1) = P(v2 < 2b1) = Fv2(2b1) = 2b1$ (for uniform distribution). Maximizing $2b1(v1 - b1)$ with respect to $b1$ yields $b1 = v1/2$.
- **Theorem (for N risk-neutral bidders, valuations i.i.d. from U[0,1])**: Each agent i bids $((N-1)/N)vi$. This is a Bayesian-Nash equilibrium.

11.4 Bidding in Second-Price Auctions

- **Theorem (for N risk-neutral bidders, valuations vi)**: Bidding their true valuation ($v1, v2, ..., vN$) is a Nash equilibrium.
- **Proof (Intuition)**:
    - **Losers**: No incentive to deviate. Reducing bids still results in losing (profit 0). Increasing bids above valuation might lead to winning at a price higher than their valuation (negative profit).
    - **Winner**: No incentive to deviate. Increasing bid changes nothing (still wins, pays second highest). Decreasing bid might still win (pays second highest), or might lose (profit 0) if the bid drops below the second highest.
    - Therefore, bidding one's true valuation is a dominant strategy in a second-price auction.

11.5 Revenue Equivalence

- **Question**: Which auction should an auctioneer choose?

- **Answer**: To some extent, it does not matter.
- **Revenue Equivalence Theorem**: Assumes `n` risk-neutral agents with independent private valuations for a single good, drawn from a common cumulative distribution `F(v)` that is strictly increasing and atomless on `[v_min, v_max]`. Any auction mechanism where:
    - The good is allocated to the agent with the highest valuation.
    - Any agent with valuation `v` has an expected utility of zero.
    - Yields the same expected revenue, and results in any bidder with valuation `v` making the same expected payment.
- **Applicability to First-Price and Second-Price Auctions**: Yes, because they are symmetric games with symmetric equilibria where higher bids correspond to higher valuations, ensuring the good goes to the highest valuation and expected utility can be zero.
- **Proof using k-th Order Statistic**: The expected value of the second-highest valuation (which is what the winner pays in a second-price auction) can be calculated using order statistics. For `n` i.i.d. draws from `U[0, vmax]`, the `k`-th order statistic is `((n+1-k)/(n+1)) * vmax`.
    - For a first-price auction, the winning bidder bids their expected payment conditional on being the winner of a second-price auction. This is `((n-1)/n) * vi` for `n` bidders, which aligns with the earlier theorem for first-price auctions.
- **Caveat**: The theorem does not guarantee that every revenue-equivalent strategy profile is an equilibrium; it only states that the expected revenue is the same under certain conditions.

12. Social Choice / Voting

12.1 Introduction to Voting

- **Core Question**: How to make a decision that respects everyone's wishes?
    - This involves concepts like consensus, majority, and democratic processes.
- **Problems**: Wishes might be incompatible, and there are challenges in making choices or electing representatives due to discretizations.

12.2 Social Choice Theory

- **Setting**: A set of outcomes, and agents have preferences across these outcomes.
- **Assumption (for now)**: No incentive issues; the center knows agents' preferences or agents declare truthfully.
- **Goal**: To define a **social choice function**, which is a mapping from everyone's preferences to a particular outcome that is then enforced.
- **Key**: How to pick such functions with desirable properties.

12.3 Non-Ranking Voting Schemes

- **Plurality**: Pick the outcome that is preferred by the most people.
- **Cumulative Voting**: Distribute a fixed number of votes (e.g., 5 votes each), with the possibility to vote for the same outcome multiple times.
- **Approval Voting**: Allow voters to accept as many outcomes as they "like."

12.4 Ranking Voting Schemes

- **Plurality with Elimination (Instant Runoff)**:
    1. Everyone selects their favorite outcome.
    2. The outcome with the fewest votes is eliminated.
    3. Repeat until one outcome remains.
- **Borda Count**:
    1. Assign each outcome a number based on rank.
    2. The most preferred outcome gets `n-1` points, the next `n-2`, down to the `n`-th outcome which gets 0 points.
    3. Sum the points for each outcome.
    4. The outcome with the highest total score wins.
- **Pairwise Elimination (Cup System)**:
    1. A schedule for the order of pairwise comparisons is decided in advance.
    2. For each pair, voters determine their preferred outcome.
    3. The less preferred outcome is eliminated, and the process continues with the schedule.

12.5 Condorcet Condition

- **Definition**: If there is a candidate who is preferred to every other candidate in pairwise runoff, that candidate should be the winner.
- **Importance**: Considered an important property for a voting system.
- **Challenge**: There is not always a Condorcet winner; sometimes, a cycle exists (e.g., A defeats B, B defeats C, and C defeats A).
- **Condorcet Example**:
    - **Preferences**: 499 agents: A > B > C; 3 agents: B > C > A; 498 agents: C > B > A.
    - **Condorcet Winner**: B (defeats A and C in pairwise comparisons).
    - **Plurality Winner**: A.
    - **Plurality with Elimination Winner**: C.

12.6 Sensitivity of Voting Schemes

- **Sensitivity to Losing Candidate**: Removing a candidate can change the winner under different voting systems.
    - **Example**: With preferences (35: A>C>B, 33: B>A>C, 32: C>B>A), A wins under plurality and Borda. If C is dropped, B wins under both.
- **Sensitivity to Agenda Setter (Pairwise Elimination)**: The order in which candidates are compared can determine the winner.
    - **Example**: With the same preferences, different orderings (A;B;C, A;C;B, B;C;A) lead to different winners (C, B, A respectively).
- **Pareto Domination Problem**: A selected candidate might be Pareto-dominated (i.e., all agents prefer another candidate).
    - **Example**: In a pairwise elimination scenario, a candidate D might win even if all agents prefer B to D.

12.7 Representative Choice

- **Problem**: How to allocate seats in a representative body based on votes (e.g., 5 seats for parties with varying percentages of votes).
- **D'Hondt Method**: A common method for proportional representation.
    - **Process**: Successive quotients are calculated for each party (`Quot = V / (s+1)`, where `V` is total votes and `s` is seats allocated so far). The party with the largest quotient wins a seat, and its quotient is recalculated. This is repeated until all seats are allocated.

12.8 Gerrymandering

- **Concept**: Manipulating electoral district boundaries to favor one party or group over another.
- **Impact**: Can significantly influence election outcomes, even if the overall popular vote is close.

13. Multiagent Decision Making: Bayesian Games

13.1 Introduction to Bayesian Games

- **Problem Addressed**: So far, games assumed common knowledge of the game being played (number of agents, actions, payoffs).
- **Bayesian Games (Games of Incomplete Information)**: Allow representation of agents' uncertainties about the game being played.
    - Uncertainty is represented as a probability distribution over a set of possible games.
- **Key Assumptions**:
    - Games differ only in their payoffs (same number of agents, same action set for each agent).
    - Beliefs of different agents are posteriors, obtained by conditioning a common prior on individual private signals.

13.2 Bayesian Games – First Definition (Information Sets)

- **Definition**: A tuple `(N, G, P, I)` where:
    - `N`: A set of agents.
    - `G`: A set of games with `N` agents each, where action sets are identical across games.
    - `P` $\in$ `Π(G)`: A common prior over games (probability distribution over `G`).
    - `I = (I1, ..., IN)`: A tuple of partitions of `G`, one for each agent.
- **Interpretation**: Agents decide without fully knowing the game that will be played. They reason about other agents without knowing what others are thinking.
- **Agent Knowledge**: Agents know the set of games, the common prior, and the equivalence classes (information sets) of all agents.

13.3 Bayesian Games – Second Definition (Types)

- **Alternative Definition**: Mathematically equivalent to the first, but with a different presentation based on **types**.

- **Definition**: A tuple (N, A, Θ, p, u) where:
  - N: A set of agents.
  - $A = A1 \times ... \times An$: Action sets for each agent.
  - $\Theta = \Theta 1 \times ... \times \Theta n$: Type space of each agent.
  - $p: \Theta \longmapsto [0,1]$: A common prior over types.
  - $u = (u1, ..., un)$: Utility function for agent i, $ui: A \times \Theta \longmapsto \mathbb{R}$.

13.4 Analyzing Bayesian Games

- **Reasoning Approach**: Using types (second definition).
- **Bayesian Nash Equilibrium**: A plan of action for each player as a function of types that maximizes each type's utility, expecting over the actions and types of other players.
- **Strategies**:
  - **Pure Strategy**: $\alpha i: \Theta i \longmapsto Ai$. A mapping from every type (from agent i) to an action (from agent i).
- **Notions of Expected Utility (Timing of Decision)**:
  - **Ex-ante**: Agent knows nothing about anyone's actual type (including their own).
  - **Interim**: Agent knows their own type but not the types of other agents.
  - **Ex-post**: Agent knows all agents' types.
- **Agent i's Interim Expected Utility**: $EUi(\alpha \mid \theta i) = \Sigma(\theta{-}i \in \Theta{-}i)\ p(\theta{-}i \mid \theta i)\ ui(\alpha, \theta i, \theta{-}i)$.
- **Bayesian Nash Equilibrium Definition**: A pure strategy profile α such that $\alpha i \in argmax(\alpha' i)\ EUi(\alpha' i, \alpha{-}i \mid \theta i)$ for each i and $\theta i \in \Theta i$.

13.5 Exercises

- **Exercise 1 (Sheriff and Suspect)**:
  - **Scenario**: Sheriff and suspect (criminal or innocent) simultaneously decide whether to shoot. Payoffs depend on types and actions.
  - **Task**: Model this as a Bayesian game and find the Bayesian Nash Equilibrium.
- **Exercise 2 (Firm and Worker)**:
  - **Scenario**: Firm recruits a worker (high or low ability). High ability worker wants to work, low ability wants to shirk. Firm wants to hire working worker, not shirking. Worker knows ability, firm doesn't. Firm believes worker is high ability with probability p.
  - **Modeling**: Nature chooses worker ability. Worker observes, firm does not.
  - **Bayesian Game Elements**:
    - $N = \{Firm, Worker\}$
    - $AF = \{hire, don't\}, AW = \{work, shirk\}$
    - $\Theta F = \{tf\}, \Theta W = \{high, low\}$
    - $p(tf, high) = p, p(tf, low) = 1 - p$
    - Utility functions $u(aF, aW, \theta F, \theta W)$ are defined by payoff tables for high and low ability workers.
  - **Task**: Check if a given pure strategy profile $\alpha* = (\alpha F*, \alpha W*)$ is a Bayes Nash equilibrium for a specific p value.