Automatic Reasoning and Planning

Course Summary - Master's Degree

Carlos Alberto Botina Carpio
Universidad Internacional de la Rioja
carlos.botina621@comunidadunir.net

November 13, 2025

Abstract

This document contains a summary of the Automatic Reasoning and Planning course syllabus for the Master's Degree. It includes a summary of the main topics covered during the sessions, as well as additional explanations and extensions of the concepts and techniques referenced in class. The purpose of this document is to serve as study material and reference for the course contents.

Notice that this document is fully customized to the author's needs, placing greater emphasis on topics that the author struggles with or has not yet mastered, while covering more briefly those topics that are already well understood by the author.

Contents

Lecture 001

1 Introduction to Decision Making

Decision making is a complex process that requires evaluating and understanding environmental conditions. In many cases, decisions become so intricate that we need to exhaustively evaluate every possible scenario. This is where mathematics becomes useful in helping us make informed decisions.

When making decisions, the following elements are involved:

- **Future effect**: What is the duration of the decision's impact? (short-term, long-term)
- **Reversibility**: Is it easy to reverse the decision?
- **Impact**: How many areas or domains are affected?
- **Quality**: Ethics, legality, behavioral principles, workplace relationships, etc.
- **Periodicity**: How frequently must this decision be made?

1.1 Decision Classification

Decisions can be classified into two main categories: **high-level** and **low-level** decisions. High-level decisions are strategic, have long-term consequences, are difficult to reverse, and affect multiple areas of an organization or system. They are typically exceptional and require careful consideration of various quality factors. In contrast, low-level decisions are operational, have minimal future impact, are easily reversible, and affect fewer areas. They are made frequently and have limited impact on important quality factors.

Table 1 summarizes the characteristics that distinguish high-level from low-level decisions based on the elements involved in decision making.

Table 1: Classification of decisions: High-level vs. Low-level

	High Level	**Low Level**
Future Effect	Affect the future	Don't affect the future
Reversibility	Difficult reversibility	Reversible
Impact	Broad impact	Little impact
Quality Factors	Affect many important	Affect few important
Periodicity	Exceptional	Frequent

Decisions can also be classified as **programmed** or **non-programmed**. Programmed decisions have a well-defined step-by-step sequence that is known and can be followed. For example, in case of an emergency, one calls the emergency number. Non-programmed decisions are unique and specific to the situation, with no defined rules or steps to follow.

1.2 Problem Classification

Problems can be classified as **structured** or **non-structured**:

- **Structured problems**: The problem contains all the information needed to solve it. All necessary data, constraints, and conditions are available from the start.
  - *Example*: Solving a system of linear equations where all coefficients and constants are given.
- **Non-structured problems**: The problem does not contain all the information needed to solve it. To solve it, we need to search for additional information.
  - *Example*: Diagnosing a medical condition where symptoms are present but additional tests, patient history, or expert consultation are required to reach a diagnosis.

2 Problem Solving Stages

2.1 First Stage: Understand the Problem's Complexity

There are many techniques to understand a problem's complexity, which can be organized into two main categories:

- **Identify the problem**: Self-questioning about the problem (origin, magnitude, focus, or history), SWOT analysis (Strengths, Weaknesses, Opportunities, Threats), discussion meetings (Scrum), etc.
- **Explain the problem**: Go beyond the surface and investigate the underlying causes of the problem. This can be done using various techniques such as ERIM problem classification, the 20 causes technique, the Ishikawa diagram (fishbone diagram), and other root cause analysis methods.

We must, therefore, create problem definitions that present characteristics that allow us to work with them efficiently. Some characteristics or assumptions that can be made in simple environments (well-defined problems) are:

- **Discrete**: The world can be conceived in states. In each state there is a finite set of perceptions and actions.
- **Accessible**: The agent can access the relevant characteristics of the environment. It can determine the **current state** of the world and the **state it would like to reach**.
- **Static and deterministic**: There is no temporal pressure nor uncertainty. The world changes only when the agent acts. The result of each action is totally defined and predictable.

2.2 Second Stage: Create a Strategy

The steps for creating a strategy are:

1. **Define strategies**: Generate potential strategies using techniques such as brainstorming, 4x4x4, etc.
2. **Choose a strategy**: Select a strategy by evaluating:
    - Benefits
    - Probability of success
    - Dependencies
    - Resources needed (time, cost)
3. **Design the strategy**: Create a roadmap defining which actions will be performed. This involves planning the sequence and details of the actions to be executed.

2.3 Third Stage: Solve the Problem

This stage is achieved by implementing the strategy, evaluating the outcome, and if necessary, refining the strategy.

3 Intelligent Agents

An **intelligent agent** is an autonomous entity that perceives its environment through sensors and acts upon that environment through actuators to achieve its goals or objectives. Intelligent agents are characterized by their ability to operate independently, make decisions based on their perceptions, and take actions that affect their environment in pursuit of their goals.

An agent is considered intelligent based on its **autonomy** and **rationality**:

- **Autonomy**: The agent operates independently without direct human intervention or control. It has control over its own actions and internal state.
- **Rationality**: The agent acts in a way that maximizes its performance measure, given the available information and its knowledge. A rational agent selects actions that are expected to achieve its goals most effectively.

3.1 Phases of Agent Decision Making

An agent must go through three fundamental phases: **feel**, **think**, and **act**.

- **Feel**: Using perception of the environment through their sensors, the agent extracts and processes information. This phase involves gathering raw data from the environment and converting it into a usable format.
- **Think**: The agent reasons and decides through a deliberative process, using the information obtained from the environment and its internal memory. This phase involves analyzing the current situation, considering possible actions, and selecting the best course of action to achieve its goals.

- **Act**: Through actuators, the agent produces changes in the environment that will help it achieve its goal. For this, the agent must convert its decisions into information that the actuators can understand and execute.

Example: Consider a robotic vacuum cleaner agent. In the **feel** phase, it uses sensors (cameras, bump sensors, dirt detectors) to perceive the room layout, detect obstacles, and identify dirty areas. In the **think** phase, it processes this information along with its internal map and battery level, deciding which areas to clean next and planning an efficient path. In the **act** phase, it converts these decisions into motor commands that control its wheels and vacuum mechanism, moving through the room and cleaning the identified areas.

4 Intelligent Agents Architectures

Agent architectures define the internal structure and organization of an intelligent agent, determining how it processes information, makes decisions, and acts. There are several main types of agent architectures: **deliberative**, **reactive**, **hybrid**, and **cognitive**.

4.1 Deliberative Architecture

A **deliberative architecture** (also known as a *symbolic* or *thinking* architecture) is characterized by the agent's ability to maintain an internal model of the world and engage in explicit reasoning and planning before taking action. The agent uses symbolic representations of knowledge and performs logical reasoning to determine the best course of action.

Key characteristics:

- Maintains an internal model or representation of the world
- Uses symbolic knowledge representation
- Performs explicit reasoning and planning
- Makes decisions based on logical inference
- Typically slower to respond but more thoughtful

This process involves explicit reasoning and planning, making it a deliberative approach. The agent "thinks before it acts," considering multiple possibilities and their outcomes.

Figure 1 illustrates the classic deliberative agent architecture, which follows a Sense-Plan-Act cycle. The architecture consists of three main internal components:

- **Deliberative Planner**: Generates high-level plans based on goals and current state information. It receives failure signals from the monitoring component and creates or refines plans accordingly.
- **Monitoring**: Monitors the execution of the plan, compares the current state with the expected state, and sends failure signals to the planner if deviations occur. It receives plans from the planner and sends specific actions to the execution component.

- **Execution**: Interacts directly with the environment through sensors and actuators. It receives sensor data from the environment, updates the monitoring component with the current state, receives action commands from monitoring, and executes actions in the environment.

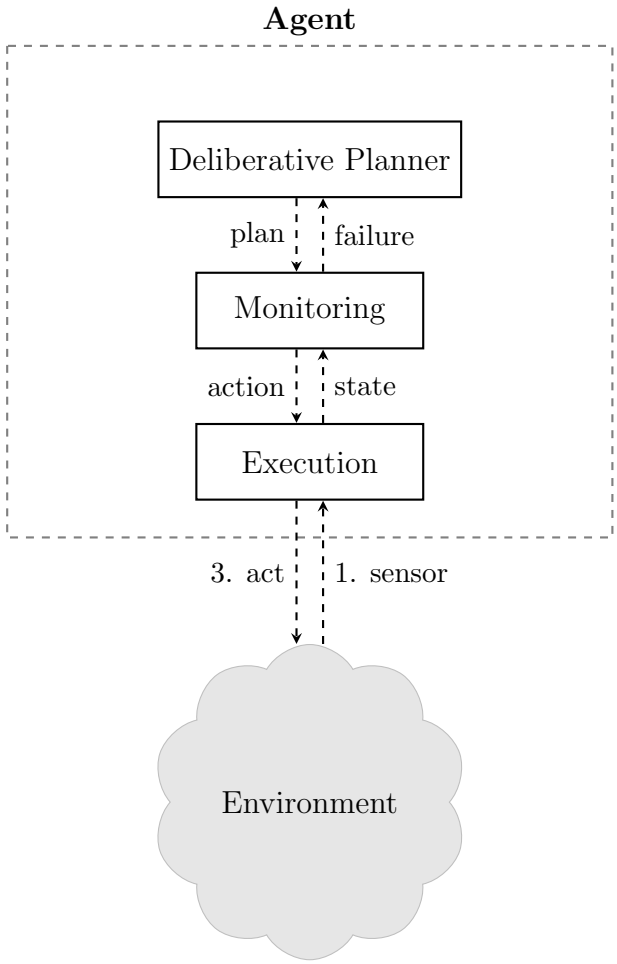

Figure 1: Classic deliberative agent architecture (Sense-Plan-Act cycle)

The flow operates as a continuous cycle:

1. The agent **senses** the environment through sensors (1. sensor), providing raw data to the Execution component.
2. Execution updates Monitoring with the current **state**.
3. Monitoring compares the state with the **plan** received from the Deliberative Planner and determines the next **action** for Execution. If the plan is not progressing as expected, Monitoring sends a **failure** signal to the Deliberative Planner.
4. The Deliberative Planner receives failure signals and generates or refines a **plan**, which is sent to Monitoring.
5. Execution **acts** (3. act) upon the environment based on the action received from Monitoring.

This cycle represents how a deliberative agent continuously perceives, plans, and acts to achieve its goals, with a mechanism for detecting and responding to plan failures.

4.1.1 Examples of Deliberative Architectures

- **Planning agents**: These agents use automated planning algorithms to generate sequences of actions that achieve specific goals. They maintain a symbolic representation of the world state and use search algorithms to find optimal or near-optimal plans. Planning agents are commonly used in robotics, autonomous systems, and game AI where complex sequences of actions need to be coordinated.

- **Belief Desire & Intention (BDI)**: This architecture models agents based on three mental attitudes: **Beliefs** (what the agent knows about the world), **Desires** (the agent's goals or objectives), and **Intentions** (the commitments to specific plans of action). BDI agents reason about their beliefs, select desires to pursue, and commit to intentions (plans) to achieve those desires. This architecture is particularly useful for modeling complex, goal-oriented behavior in multi-agent systems and autonomous agents.

4.2 Reactive Architecture

A **reactive architecture** (also known as a *behavior-based* architecture) is characterized by the agent's direct mapping from perceptions to actions without maintaining an internal world model or engaging in complex reasoning. The agent responds quickly to environmental stimuli through simple stimulus-response rules.

Key characteristics:

- No internal world model or symbolic representation
- Direct mapping from sensors to actuators
- Simple stimulus-response rules or behaviors
- Fast response time
- Emergent behavior from simple rules

Example: Consider a simple obstacle-avoidance robot with a reactive architecture. The robot has sensors on its front and sides, and it follows these simple rules:

- If the front sensor detects an obstacle, turn right
- If the right sensor detects an obstacle, turn left
- If no obstacles are detected, move forward
- If both sensors detect obstacles, move backward

The robot doesn't maintain a map of its environment or plan a path. It simply reacts to what it perceives at each moment. This makes it very fast and efficient for simple tasks, though it may not find the optimal path. The robot's navigation behavior emerges from these simple reactive rules.

Figure 2 illustrates a reactive agent architecture that incorporates a **Reactive Planner** component:

- **Reactive Planner**: Positioned at the top, this component generates reactive action sequences in direct response to the current state. Unlike deliberative planners, it does not maintain a complex world model but instead quickly generates actions based on immediate perceptions. It receives failure signals from Monitoring and sends actions back to Monitoring.
- **Monitoring**: Positioned in the middle, it monitors the execution of actions and the current state. It receives state information from Execution, sends failure signals to the Reactive Planner when deviations occur, receives actions from the Reactive Planner, and sends action commands to Execution. It also receives external plans.
- **Execution**: Located at the bottom, it interacts directly with the environment through sensors and actuators. It receives sensor data from the environment, updates Monitoring with the current state, receives action commands from Monitoring, and executes actions in the environment.

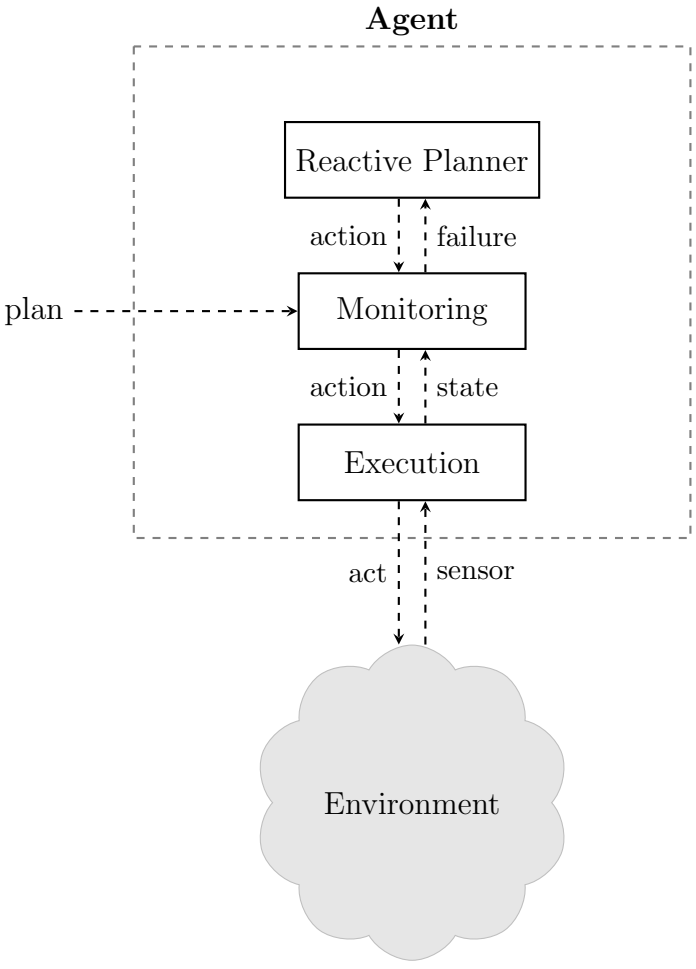

Figure 2: Reactive agent architecture with Reactive Planner

The flow operates as follows:

1. An external **plan** is provided to the Monitoring component (this could come from a higher-level planner or user input).

2. The Environment sends **sensor** data to the Execution component, providing raw information about the current state of the environment.

3. Execution processes the sensor data and updates Monitoring with the current **state**.

4. Monitoring evaluates the state against the plan. If there is a deviation or problem, it sends a **failure** signal to the Reactive Planner.

5. The Reactive Planner receives the failure signal and immediately generates a reactive **action** response, which it sends back to Monitoring.

6. Monitoring receives the action from the Reactive Planner and forwards the **action** command to Execution.

7. Execution **acts** upon the environment based on the action received from Monitoring.

This architecture emphasizes fast, reactive responses to environmental changes while still allowing for external plan guidance. The Reactive Planner provides quick adaptation without the overhead of maintaining complex internal models. The bidirectional flow between Monitoring and the Reactive Planner enables rapid failure detection and action generation.

4.2.1 Examples of Reactive Architectures

- **Subsumption architecture**: A layered architecture where behaviors are organized in levels of increasing complexity. Lower-level behaviors (like obstacle avoidance) can subsume or override higher-level behaviors (like exploration) when triggered by environmental conditions. Each layer operates independently and reactively, with no central control or world model.

- **Agent network architecture from Pattie Maes**: A distributed reactive architecture where multiple simple agents (or behaviors) are connected in a network. Each agent has local rules and can activate or inhibit other agents. The overall behavior emerges from the interactions between these agents, without centralized planning.

- **Reactive execution model**: Is domain independent and operates with structures precalculated at runtime.

4.3 Hybrid Architecture

A **hybrid architecture** combines elements of both deliberative and reactive architectures. It typically has multiple layers: a reactive layer for fast, immediate responses to critical situations, and a deliberative layer for complex planning and reasoning. This architecture attempts to get the best of both worlds: the speed of reactive systems and the intelligence of deliberative systems.

Key characteristics:

- Combines reactive and deliberative components

- Multiple layers of control (reactive at the bottom, deliberative at the top)

- Fast response for urgent situations (reactive layer)
- Complex reasoning and planning for strategic decisions (deliberative layer)
- Coordination between layers

Example: Consider an autonomous vehicle with a hybrid architecture. The vehicle has two main layers:

- **Reactive layer**: Handles immediate, critical situations. For example:
  - If a pedestrian suddenly appears in front, immediately apply brakes (no time for planning)
  - If another vehicle swerves into the lane, quickly adjust steering
- **Deliberative layer**: Handles strategic planning and navigation. For example:
  - Plans the route from origin to destination
  - Analyzes traffic conditions and selects the best path
  - Decides when to change lanes based on traffic patterns
  - Maintains a map and tracks the vehicle's position

The reactive layer ensures safety by responding instantly to immediate threats, while the deliberative layer handles the overall navigation strategy. The layers work together: the deliberative layer sets the general plan, and the reactive layer handles unexpected situations that require immediate action.

4.3.1 Examples of Hybrid Architectures

- **Procedural Reasoning System (PRS)**: A BDI (Belief-Desire-Intention) architecture that combines deliberative reasoning with reactive capabilities. The agent maintains **Beliefs** (facts about the world expressed in first-order logic), **Desires** (system behaviors or goals), and **Intentions** (the current set of active plans). PRS includes a library of partially specified plans called Knowledge Areas (KAs), each with an activation condition. KAs can be activated by goals or by data, and can be reactive, allowing PRS to respond quickly to environmental changes while also reasoning about which plans to execute.
- **COSY (Cooperative System)**: A BDI architecture that combines elements from both PRS and IRMA architectures. It has five main components: **Sensors** (receive perceptual inputs), **Actuators** (perform actions), **Communications** (send messages), **Cognition** (mediates between intentions and knowledge to choose actions), and **Intention** (contains long-term goals and control elements). COSY combines deliberative reasoning with reactive communication and action capabilities, making it suitable for interactive and collaborative environments.

4.4 Cognitive Architecture

A **cognitive architecture** is a computational framework that models the structure and processes of human cognition. It provides a unified theory of how the mind works, including perception, memory, reasoning, learning, and decision-making. Cognitive architectures can be defined as a hypothesis about the fixed structures that provide a mind, whether in natural or artificial systems, and how they work together—along with the knowledge and skills incorporated within the architecture—to produce intelligent behavior in a diversity of complex environments. Cognitive architectures aim to create artificial agents that can exhibit human-like intelligence and behavior.

Key characteristics:

- Models human cognitive processes and structures
- Provides unified framework for multiple cognitive functions
- Includes memory systems (short-term and long-term)
- Supports learning and adaptation
- Integrates perception, reasoning, and action
- Based on cognitive science and psychology principles

4.4.1 Examples of Cognitive Architectures

- **ACT-R**: A cognitive architecture developed primarily by John Robert Anderson at Carnegie Mellon University. The most important assumption of ACT-R is that human knowledge can be divided into two irreducible types of representations:
  - **Declarative knowledge**: Represented as **chunks** (vector representations of individual properties, each accessible from a labeled slot)
  - **Procedural knowledge**: Represented as production rules

  Chunks are maintained and accessed through **buffers**, which are the front-end of **modules** (specialized and largely independent brain structures). ACT-R has two main types of modules:
  - **Perceptual-motor module**: Manages interaction with the environment, handling the flow of perception and action to connect the agent with the world.
  - **Memory module**: Divided into:
    - **Long-term memory** (production memory): Contains production rules
    - **Short-term memory** (working memory or declarative memory): Contains current facts about the world
- **SOAR**: A cognitive architecture created at Carnegie Mellon University by Laird, Newell, and Rosenbloom (1987). The ultimate goal of SOAR is to provide a foundation for a system capable of general intelligent behavior, supporting the full range of cognitive tasks, problem-solving methods, and knowledge representations. SOAR is both a theory of cognition and a computational implementation of that theory.

The design of SOAR is based on the hypothesis that all goal-oriented deliberate behavior can be understood as the selection and application of **operators** to a **state**:

- **State**: A representation of the current problem situation
- **Operator**: Transforms a state (performs changes in the representation)
- **Goal**: A desired result for the problem

SOAR runs continuously, attempting to apply the current operator and select the next operator (a state can have only one operator at a time) until the goal is achieved.

SOAR has separate memories with different representation modes:

- **Short-term memory**: Stores sensor data, intermediate inferences about current data, currently active goals, and active operators (actions and plans)
- **Long-term memory** (production memory): Maintains knowledge for responding to situations through procedures. It stores problem-solving knowledge, inference rules, and knowledge for selecting and applying operators in specific states

4.5 Other Architectures

- Three layer architecture
- Multilayer architecture
- Three tower architecture

5 Summary - Lecture 001

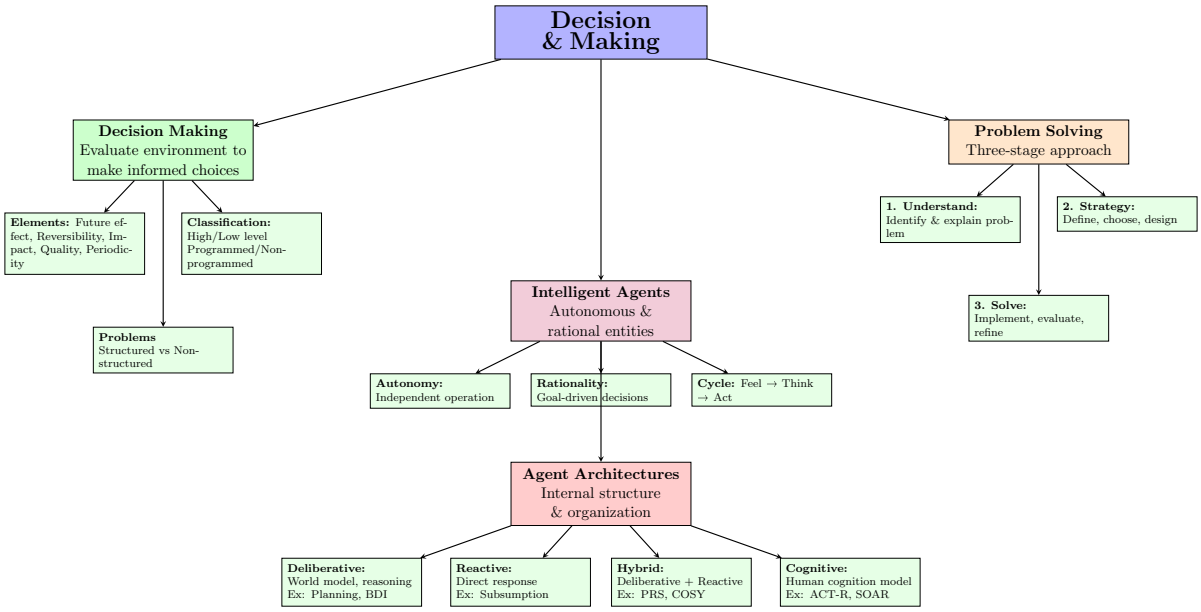

Figure 3: Conceptual map of the Reasoning and Planning course - Lecture 001

Lecture 002

6 Symbolic Representation of Knowledge

When agents make decisions based on changes in the environment, a fundamental question arises: **how do we represent the environment and the problem's information?** Agents need a way to understand and work with information about the world, which is where symbolic representation becomes essential.

Reasoning is an internal process that operates on external entities. The key insight is that **operations with representations substitute operations with the real world**. Instead of directly interacting with the environment, agents can manipulate symbolic representations of it, allowing them to reason about possible actions and outcomes before actually taking action.

Symbolic representation allows agents to create **internal models of the external world using symbols** (such as variables, predicates, logical formulas, or other formal structures). These representations enable reasoning processes that can:

- **Simulate** different scenarios and outcomes
- **Predict** the consequences of actions
- **Plan** sequences of actions before execution
- **Learn** from experience without direct interaction

This makes decision-making more efficient and safer, as agents can explore possibilities and evaluate strategies in a controlled, internal environment before committing to actions

in the real world. It is important to note that reasoning and action are **complementary** rather than substitutes: reasoning often precedes and guides action, enabling agents to make informed decisions rather than acting randomly.

6.1 Requirements for Knowledge Representation

According to (Molina González, 2006) a knowledge representation must satisfy:

1. **Formal**: The representation must be unambiguous. Natural language is not considered a knowledge representation because of its ambiguities.

   Example: The sentence "I saw the man with binoculars" is ambiguous—it could mean "I used binoculars to see the man" or "I saw the man who had binoculars." A formal representation would explicitly distinguish these meanings, such as `Saw(I, man)` $\wedge$ `Used(I, binoculars)` versus `Saw(I, man)` $\wedge$ `Has(man, binoculars)`.

2. **Expressive**: The representation must be rich enough to capture the different aspects that need to be distinguished. For example, first-order predicate logic formulas are more expressive than propositional calculus.

   Example: Propositional logic can only express simple statements like "It is raining" (P) or "The ground is wet" (Q). Predicate logic can express relationships and quantification, such as $\forall$`x (Bird(x)` $\rightarrow$ `CanFly(x))` meaning "All birds can fly," or `Loves(john, mary)` expressing the relationship "John loves Mary."

3. **Natural**: The representation should be sufficiently analogous to natural ways of expressing knowledge. Traditional quantitative mathematical representations (e.g., matrices) can be too artificial for emulating reasoning processes.

   Example: Representing "John loves Mary" as a matrix entry (row 1, column 2 = 1) is mathematically precise but doesn't match how humans naturally think about relationships. A more natural representation would be a semantic network with nodes and edges: `[John] --loves--> [Mary]`, or a frame structure with properties, which aligns better with human cognitive patterns.

4. **Tractable**: The representation must be computationally tractable, meaning there must exist sufficiently efficient procedures to generate answers through manipulation of the knowledge base elements.

   Example: While a representation might be perfectly formal and expressive, if answering queries requires exponential time or is undecidable, it becomes impractical. For instance, certain logical formalisms may be too complex to reason about efficiently, making them unsuitable for real-time applications despite their theoretical expressiveness.

It is important to note that "natural" in this context refers to **cognitive naturalness**—how well the representation aligns with human thinking patterns—not to natural language. Natural language is ambiguous and therefore not formal, while we seek representations that are both formal (unambiguous) and natural (intuitive). The ideal representation balances both requirements.

In general, it is convenient to create information representations based on a **single representation**. This approach improves knowledge base maintenance, as it provides consistency and simplifies updates across the system.

Example: A system might use only first-order predicate logic to represent all knowledge:

- Facts: `Student(john)`, `Course(cs101)`, `Enrolled(john, cs101)`
- Rules: $\forall$`x (Student(x)` $\wedge$ `Enrolled(x, y)` $\rightarrow$ `TakesCourse(x, y))`
- Relationships: `Teaches(profSmith, cs101)`, `Prerequisite(cs101, cs201)`

Alternatively, a system might use only semantic networks, representing everything as nodes and edges: `[John] --is_a--> [Student]`, `[John] --enrolled_in--> [CS101]`, etc.

However, in some cases, attempting to represent all knowledge in a single representation can **limit the application of techniques and algorithms**. Different problems and subtasks may require different representation formalisms or specialized techniques that are not well-suited to a unified representation.

For complex systems that require making decisions at different levels, it is common to employ the idea of generating **multi-layer agents** (Molina González, 2006) that decompose the problem into levels. For each level, a treatment is established oriented to a specialized agent that needs a **concrete representation of part of the environment information**. Each agent at its respective layer uses a representation tailored to its specific needs and the techniques it employs.

6.2 Types of Knowledge

To solve problems in a natural way, it is necessary to carry out a precise analysis of knowledge. For this, we must take into account the different classifications of knowledge used in artificial intelligence (Molina González, 2006). The main types of knowledge are:

1. **Domain Knowledge**: Knowledge about a specific context or problem, represented in a **declarative** way (describes what exists, properties, and constraints). It can be incomplete and does not require order or relationships between elements.
2. **Explicit Knowledge**: **Knowledge extracted from introspective analysis of one's own reasoning and problem-solving processes**. It is usually expressed through frames or rules about how problems are solved.
3. **Implicit Knowledge**: Knowledge about innate capacities or abilities that are not easily expressed verbally. Bayesian networks or neural networks are used for its representation or modeling.
4. **Superficial Knowledge**: Knowledge obtained through experience in solving similar problems. It uses practical rules or heuristics that work but don't explain the underlying theoretical principles.
5. **Deep Knowledge**: Knowledge based on a well-structured theoretical framework that explains the underlying principles and mechanisms in detail. However, it is not present in many problems because it is not easy to have a theoretical analysis of the environment's functioning in all problems.

6. **Control Knowledge**: The **strategy/organization for the problem-solving process** — the execution order and approach for how to solve the problem, not just the steps of the task itself. In many cases, it can lead to coding a program in the expansion sequence of search algorithms.

7. **Metaknowledge**: **Knowledge about how to generate, transfer, and learn from knowledge**. It allows generating new models from previous problem models and establishes relationships between levels of knowledge bases.

7 Introduction to Reasoning

Reasoning refers to a set of mental activities that connect ideas based on rules that justify an idea, allowing problem-solving through conclusions.

As can be observed in these definitions, all authors refer to the same concepts: **premises** (initial propositions, what is already known) and **conclusion** (final proposition obtained from the premises, representing new knowledge).

7.1 Elements of Reasoning: Content and Form

In all reasoning, there exist two elements: **content** and **form**.

- **Content**: What makes a proposition true or false. It is the reference to objects and properties. Content deals with the actual meaning and truth value of statements in the real world.

  Example: In the proposition "It is raining," the content refers to the actual weather condition. This proposition can be true or false depending on whether it is actually raining.

- **Form**: The logical connection between the antecedents (what is already known, the premises) and the consequents (the conclusion inferred from the antecedents). This connection that implies inference is expressed through conjunctions. Form is what makes the proposition **valid**, and consists of using symbols to express the validity of propositions. Form deals with the logical structure of reasoning, independent of whether the statements are actually true or false.

  Example: Consider the following reasoning:

  - Premise 1: "If it rains, then the ground gets wet"
  - Premise 2: "It is raining"
  - Conclusion: "Therefore, the ground gets wet"

  The form of this reasoning is: `If P, then Q. P. Therefore, Q.` This logical structure is valid regardless of whether it is actually raining or not. The same form can be applied to different content:

  - Premise 1: "If John is a student, then John studies"
  - Premise 2: "John is a student"
  - Conclusion: "Therefore, John studies"

Both examples share the same valid logical form, even though they have different content.

The distinction between content and form is crucial: **content** determines whether propositions are true or false in the real world, while **form** determines whether the reasoning structure is valid (whether the conclusion follows logically from the premises).

We speak of **valid reasoning** when the conclusion follows from the premises. A reasoning is considered valid based on its logical form, even if the conclusion or the premises are false. On the other hand, **invalid reasoning** occurs when, from true premises, a false conclusion is obtained.

Table 2 shows the different scenarios for reasoning validity based on the truth values of premises and conclusion:

Table 2: Validity or Non-Validity of a Reasoning

If the premises are...	**And the conclusion is...**	**The reasoning is...**
True	True	Valid
True	False	Invalid
False	True	Valid
False	False	Valid

The concept of reasoning validity is directly analogous to the truth conditions of a conditional proposition. Table 3 shows the truth table for a conditional proposition ($p \rightarrow q$), where p represents the premises (antecedent) and q represents the conclusion (consequent):

Table 3: Truth Table of a Conditional Proposition

p	**q**	**p → q**
1 (True)	1 (True)	1 (True)
1 (True)	0 (False)	0 (False)
0 (False)	1 (True)	1 (True)
0 (False)	0 (False)	1 (True)

As can be observed, the highlighted row in both tables (premises True, conclusion False) represents the only scenario where the reasoning is invalid and the conditional proposition is false. This emphasizes that an argument is invalid if and only if it is possible for its premises to be true and its conclusion false.

The following examples illustrate each scenario from Table 2:

1. **True premises → True conclusion (Valid)**:
    - Premise 1: "If it rains, then the ground gets wet" (True)
    - Premise 2: "It is raining" (True)

- Conclusion: "Therefore, the ground is wet" (True)

This reasoning is **valid** because the conclusion follows logically from the premises (modus ponens), and all statements are true.

2. **True premises → False conclusion (Invalid)**:

   - Premise 1: "All dogs are animals" (True)
   - Premise 2: "Lassie is an animal" (True)
   - Conclusion: "Therefore, Lassie is a dog" (False - Lassie could be any animal)

   This reasoning is **invalid** because the conclusion does not follow from the premises (fallacy of affirming the consequent). Even though the premises are true, the logical form is incorrect, making it possible for the conclusion to be false.

3. **False premises → True conclusion (Valid)**:

   - Premise 1: "If it is July, then it is winter" (False - in the Northern Hemisphere)
   - Premise 2: "It is July" (False - assume it is actually January)
   - Conclusion: "Therefore, it is winter" (True - it is winter in January)

   This reasoning is **valid** because the logical form (modus ponens) is correct, even though both premises are false and the conclusion happens to be true. The validity depends on the form, not the truth values.

4. **False premises → False conclusion (Valid)**:

   - Premise 1: "All insects are mammals" (False)
   - Premise 2: "Spiders are insects" (False - spiders are arachnids)
   - Conclusion: "Therefore, spiders are mammals" (False)

   This reasoning is **valid** because the conclusion follows logically from the premises using a valid syllogistic form. Even though both premises and the conclusion are false, the logical structure is correct—if the premises were true, the conclusion would necessarily follow.

8 Reasoning Classification

Although there are many types of reasoning, we will focus on the most important ones for artificial intelligence: **deductive**, **inductive**, and **abductive** reasoning.

8.1 Deductive Reasoning

Reasoning is **deductive** when it requires that the conclusion necessarily and forcibly derives from the premises. For this reason, it is considered rigorous.

Example of deductive reasoning:

- "If it snows, then it is cold"
- "It is snowing"

- "Therefore, I am cold"

It is understood that validity exists when, from true premises, a false conclusion cannot be obtained. From false premises, true conclusions can be derived, and yet the argument can still be valid.

Truth occurs when what is described in the premises corresponds to reality. This type of reasoning goes from **general to particular**.

8.1.1 Types of Deductive Reasoning

Within deductive reasoning, several types are distinguished:

- **Categorical deductive reasoning**: Starts from two true premises that will lead to a true conclusion.

  Example:

  - Premise 1: "All humans are mortal" (True)
  - Premise 2: "Socrates is a human" (True)
  - Conclusion: "Therefore, Socrates is mortal" (True)

- **Propositional deductive reasoning**: Relates two premises where one is a condition of the other, antecedent and consequent.

  Example:

  - Premise 1: "If it rains, then the ground gets wet" (antecedent: it rains, consequent: ground gets wet)
  - Premise 2: "It is raining" (antecedent is true)
  - Conclusion: "Therefore, the ground is wet" (consequent follows)

- **Disjunction or dilemma**: The relationship between the premises is one of contraries, therefore the conclusion discards one of them.

  Example:

  - Premise 1: "Either it is day or it is night"
  - Premise 2: "It is not day"
  - Conclusion: "Therefore, it is night" (discards the first option)

8.1.2 Forms of Deductive Reasoning

There are two forms of deductive reasoning:

- **Immediate**: The only logical operation is the change of judgment.

  Example:

  - Original judgment: "All students are learners"
  - Immediate conclusion: "No students are non-learners" (direct conversion/-transformation of the same judgment)

In immediate reasoning, the conclusion is obtained directly from a single premise by changing its form, without needing additional premises.

- **Mediate**: A mediation relationship is established between judgments to reach the conclusion.

  Example:

  - Premise 1: "All mammals are warm-blooded"
  - Premise 2: "All dogs are mammals" (middle term: "mammals")
  - Conclusion: "Therefore, all dogs are warm-blooded"

  In mediate reasoning, the conclusion is reached by connecting two premises through a middle term (in this case, "mammals"), which mediates the relationship between the other terms.

The deductive method goes from **general to particular**. Table 4 illustrates a classic syllogism example:

Table 4: Example of a syllogism

Part	**Abbreviation**	**Statement**
Major Premise	MP	Humans are mortal.
Minor Premise	SM	Greeks are humans.
Conclusion	SP	Greeks are mortal.

In syllogistic logic, the abbreviations follow a standard terminology:

- **S (Subject)**: The subject of the conclusion ("Greeks")
- **P (Predicate)**: The predicate of the conclusion ("mortal")
- **M (Middle)**: The term that appears in both premises but not in the conclusion ("humans")

In this syllogism:

- **Major Premise (MP)**: Contains the Major Term (P = "mortal") and Middle Term (M = "humans")
- **Minor Premise (SM)**: Contains the Minor Term (S = "Greeks") and Middle Term (M = "humans")
- **Conclusion (SP)**: Contains the Minor Term (S = "Greeks") and Major Term (P = "mortal")

The conclusion is labeled **SP** because it contains the Subject (S) and Predicate (P) terms. The middle term (M = "humans") connects the premises but does not appear in the conclusion.

8.2 Inductive Reasoning

Inductive reasoning creates probable conclusions according to the given premises. It is based on the idea that if various events present the same situation as their premises, there is a probability that the result will be identical. To induce means precisely to extract general conclusions from particular experiences.

The difference with deductive reasoning is that the conclusion is **not necessarily obtained** from the premises. The conclusion of inductive reasoning is obtained through the direct observation of particular cases.

8.2.1 Types of Inductive Reasoning

Within inductive reasoning, there are different types:

- **Complete inductive reasoning** (also called perfect inductive reasoning): Occurs when all particular cases are included in the premises.

  Example:

  - Observation: "Student 1 passed the exam"
  - Observation: "Student 2 passed the exam"
  - Observation: "Student 3 passed the exam"
  - Observation: "Student 4 passed the exam"
  - Observation: "Student 5 passed the exam"
  - (These are all the students in the class)
  - Conclusion: "Therefore, all students in the class passed the exam"

  Since all particular cases (all students) have been observed, this is complete inductive reasoning.

- **Incomplete inductive reasoning** or imperfect inductive reasoning: Only certain particular cases are included in the premises.

  Example:

  - Observation: "The swan I saw in the park is white"
  - Observation: "The swan I saw at the lake is white"
  - Observation: "The swan I saw in the zoo is white"
  - Observation: "The swan I saw in the river is white"
  - (These are only some of all the swans that exist)
  - Conclusion: "Therefore, all swans are white"

  Since only some particular cases (some swans) have been observed, this is incomplete inductive reasoning. The conclusion is probable but not certain, as there might be swans that haven't been observed (e.g., black swans in Australia).

The inductive method goes from **particular to general**. Table 5 illustrates how inductive reasoning works, showing the reverse direction compared to deductive reasoning:

Table 5: Example of inductive reasoning

Part	**Abbreviation**	**Statement**
Minor Premise	SM	Greeks are human beings.
Conclusion	SP	Greeks are mortal.
Major Premise	MP	Human beings are mortal.

In inductive reasoning, we start by observing particular cases (Greeks are humans, Greeks are mortal) and then infer the general rule (Human beings are mortal). This is the opposite direction of deductive reasoning, which starts with the general rule and applies it to particular cases.

8.3 Abductive Reasoning

Abductive reasoning (also called retroduction) is a method used to find explanations for observed facts. From a fact, we arrive at the actions that caused it. Aristotle was the first to describe this type of reasoning.

Key characteristics:

- Starts from **facts** and seeks a **theory** (from effect to cause)
- The key concept is the **syllogism**, where:
  - Major premise is considered **certain**
  - Minor premise is considered **probable**
  - Conclusion has the same level of **probability** as the minor premise

  Example:

  - Major Premise (certain): "If it rains, then the ground gets wet"
  - Minor Premise (probable): "The ground is wet" (observed fact)
  - Conclusion (probable): "Therefore, it probably rained" (inferred cause from the effect)
- Relates the observable with something that cannot be directly observed
- For Charles S. Peirce (Peirce, 1867), it is an inferential process related to the **generation of hypotheses**

Abductive reasoning process (three steps):

1. The object or fact (observation)
2. Hypothesis of why the object or fact occurs
3. Affirm that the cause was responsible for the object or fact

Scheme: "I see A with characteristic Z. Since all A I see are Z, then any element A has characteristic Z."

Importance:

- Allows thinking in an **alternative way**, without following usual reasoning paths
- Leads to **disruptive and novel solutions**
- Contrary to deductive reasoning, which keeps us in the comfort zone
- **Innovation** is strongly linked to abductive reasoning
- Enriches processes in the testing phase, providing a perspective of change

9 Summary - Lecture 002

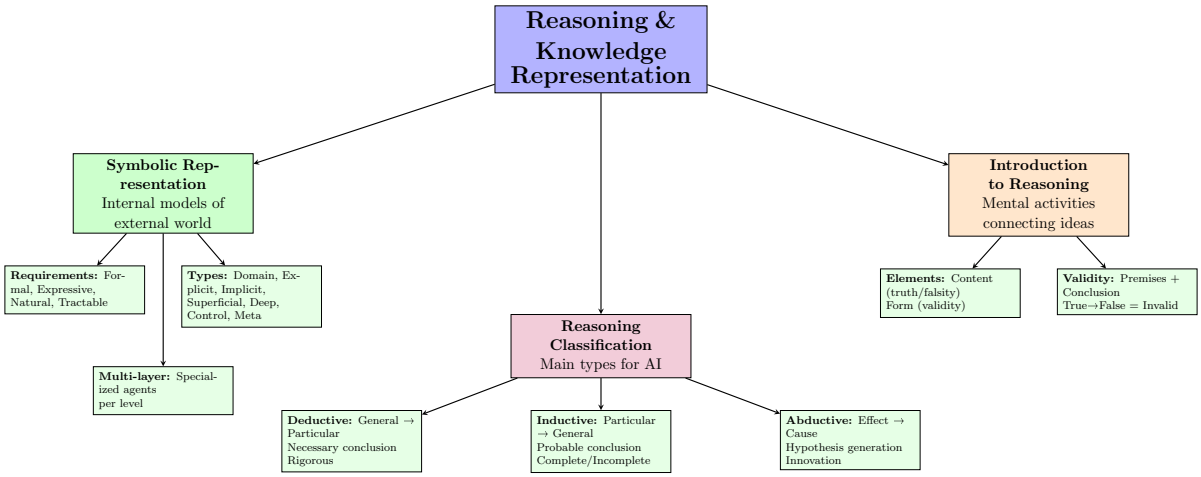

Figure 4: Conceptual map of the Reasoning and Planning course - Lecture 002

References

Martín Molina González. *Métodos de resolución de problemas: Aplicación al diseño de sistemas inteligentes*. Fundación General de la UPM, 2006. URL `https://oa.upm.es/14207/`.