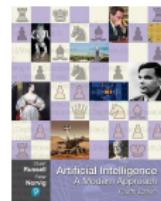


Knowledge Representation

Instructor: Dr. GP Saggese - gsaggese@umd.edu

References:

- Papers and Internet
- Russell et al.: “*Artificial Intelligence: A Modern Approach*” (4th ed, 2020)
 - Chap 7: Logical agents
 - Chap 8, First-order logic
 - Chap 9: Inference in first-order logic
 - Chap 10, Knowledge representation



- ***Knowledge Representation***

- Basics of Knowledge Representation
- Examples of Logic
- Logical Agents
- Ontologies
- Reasoning in Ontologies

- Propositional logic
- First-order Logic
- Non-classical Logics

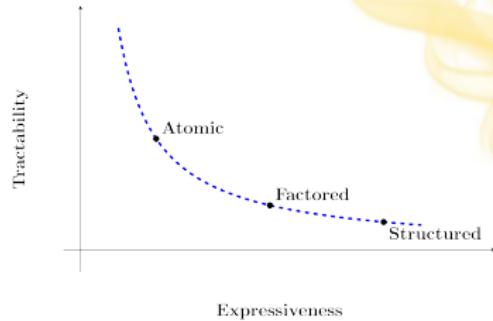
- Knowledge Representation
 - *Basics of Knowledge Representation*
 - Examples of Logic
 - Logical Agents
 - Ontologies
 - Reasoning in Ontologies
- Propositional logic
- First-order Logic
- Non-classical Logics

What is Knowledge Representation?

- **Knowledge Representation (KR)** is the study of how to formally encode information so that machines can reason with it
 - E.g., rules, logic, ontologies, semantic networks
 - It is at the heart of symbolic AI and complements learning-based approaches
- Defines:
 - **structure** (how knowledge is organized)
 - **semantics** (what it means)
- Serves as a bridge between perception (data) and reasoning (logic)
 - Essential for explainability and transparency in intelligent systems
- Enables machines to:
 - Draw conclusions
 - Perform planning
 - Answer queries
 - ...

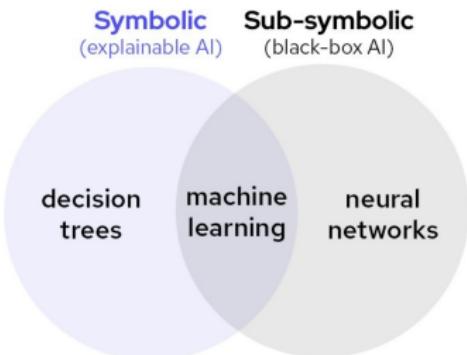
Expressiveness vs. Tractability

- **Tradeoff in AI / ML**
 - **Expressiveness:** richness of concepts that can be captured
 - **Tractability:** whether reasoning can be performed efficiently
 - More expressive languages lead to harder computation
- Choosing the right knowledge representation formalism depends on the application needs
 - **Atomic**
 - Treats each state as a single, indivisible entity
 - E.g., depth-first search algorithms (e.g., E3 in Chess)
 - Simple and fast but limited in capturing complex relationships
 - **Factored**
 - E.g., propositional logic
 - E.g., $P_{1,1}$: "Pit in square (1,1)", $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$
 - Captures relationships between variables but can't express complex structures
 - **Structured**
 - E.g., first-order logic
 - $\forall x \forall y Father(x, y) \Rightarrow Parent(x, y) = "A\ father\ of\ a\ person\ is\ their\ parent"$
 - More expressive but undecidable in general



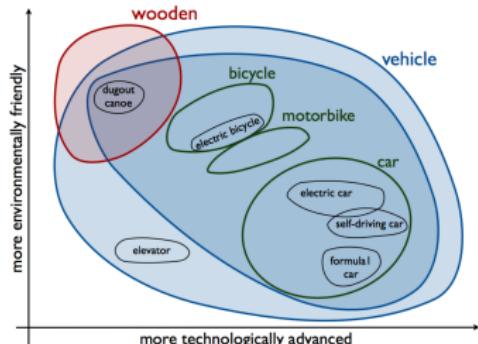
Symbolic vs. Sub-symbolic Representation

- **Symbolic knowledge** representation uses discrete, human-readable symbols
 - E.g., logic, knowledge graphs
 - Interpretable and suitable for rule-based reasoning
 - Struggle with ambiguity
- **Sub-symbolic knowledge** representation uses learned, distributed representations
 - E.g., vector embeddings
 - E.g., deep learning excels at perception and pattern recognition
 - Lack transparency
- **Neuro-symbolic approaches** blends the two approaches
 - Reason over learned concepts using structured logic



Neuro-symbolic Approach: Conceptual Spaces

- **Conceptual spaces** are frameworks for representing knowledge using geometric structures
 - A concepts is a region in a multidimensional space defined by quality dimensions
 - Similarity between objects is modeled by spatial distance
 - Each dimension represents an interpretable feature



- **Example**
 - Dimensions: Color, Size, Shape
 - "Apple" occupies a region that is typically red / medium-sized / round
 - "Banana" occupies a different region: yellow / medium / curved and long
- Differences from symbolic representations
 - Symbolic systems use discrete symbols without structure
 - E.g., Apple vs Banana
- **Benefits**
 - Natural modeling of similarity and vagueness
 - Useful for grounding symbols in perception (link between sensory inputs and symbolic language)

Natural languages

- **Natural languages** (e.g., English, Italian) are:
 - **Expressive**
 - Medium for communication rather than representation
 - **Ambiguous**
 - E.g., “spring” is both a “season” and “something that goes boing”
 - **Context-dependent**
 - Meaning depends on the sentence and context
 - E.g., “Look!”
- **Sapir-Whorf hypothesis**
 - The language you speak shapes how you perceive, think about, and experience the world
 - Even through arbitrary grammatical features, such as gender of nouns
 - Some languages lack words for certain concepts (e.g., direction)
 - Some languages have many words for the same concepts
 - E.g., Arctic languages have many words for snow (fresh, hard)
 - Newspeak from 1984 (Orwell)
 - You can't think certain concepts since you don't have words for it

Procedural vs Declarative Approaches

- **Procedural approach**

- Focuses on *how* a task is done
- Encodes desired behavior directly into the program
- E.g., a robot programmed with specific steps to navigate a maze

- **Declarative approach**

- Specifies *what* the goal is, *not how* to achieve it
- Describes relationships between actions and goals
- Leaves solution search to the system
- E.g., describing the goal “reach the exit” and letting the system find the path

- **Comparison**

- Procedural: more control, less flexibility
- Declarative: more abstraction, easier to modify or extend

- **Integration of approaches**

- Many successful AI systems use a hybrid approach
- Declarative knowledge can be compiled into procedural code
- E.g., a planner generates procedures (plans) from declarative goals

Programming Languages

- A **programming language** (e.g., C++, Python) is a formal language
 - Data structures represent facts
 - Code updates data structures in a domain-specific way
 - Programming is procedural (vs declarative)
- **Limitations:**
 - Programming languages lack:
 1. A general mechanism for deriving facts from other facts
 - Code updates data structures based on programmer's domain knowledge
 2. Expressiveness to handle partial information
 - A variable represents a single value or unknown
 - Can't easily handle partial information or quantify uncertainty
 - E.g., "A white knight is in b1 or in f6"
- **Declarative language**
 - E.g., propositional logic, first order logic
 - Knowledge and inference are separate:
 1. Knowledge represents the domain-specific problem
 2. Inference is domain independent
 - Compositional semantics
 - The meaning of a sentence is a function of the meaning of its parts

- Knowledge Representation
 - Basics of Knowledge Representation
 - *Examples of Logic*
 - Logical Agents
 - Ontologies
 - Reasoning in Ontologies
- Propositional logic
- First-order Logic
- Non-classical Logics

Propositional Logic

- Uses atomic statements (propositions) and logical connectives
 - **Syntax**
 - Atomic formulas: P, Q
 - Connectives: NOT (\neg), AND (\wedge), OR (\vee), IMPLIES (\Rightarrow)
 - **Semantics**
 - Based on truth tables
 - Each proposition has a binary truth value: true or false
 - **Inference mechanisms**
 - Modus ponens: from P and $P \Rightarrow Q$, infer Q
 - Resolution: derive contradictions to infer conclusions
- **Applications:** best used in closed and well-defined environments
 - Digital circuit design
 - Rule-based systems
 - Simplified AI models
- **Limitations**
 - Cannot represent objects, relations, or quantifiers
 - Not suitable for open or dynamic domains

First-Order Logic (FOL)

- **Extension of propositional logic**
 - Introduces predicates, variables, and quantifiers
 - Variables x
 - Predicate $Human(x)$
 - Universal quantifier “for all” \forall
 - Existential quantifier “there exists” \exists
 - E.g., $\forall x(Human(x) \implies Mortal(x)) = "All\ humans\ are\ mortal"$
 - Represents more complex and structured knowledge than propositional logic
 - Can model properties, relationships, and quantification over objects
- **Inference mechanisms**
 - Unification: matches predicates with variables
 - Resolution: deduces new facts from known statements
 - Model checking: verifies truth of statements under specific interpretations
- **Computational properties**
 - Inference is semi-decidable: valid conclusions may require infinite time
 - More powerful but computationally more complex than propositional logic
- **Applications**
 - Knowledge representation
 - Automated theorem proving
 - Semantic web and ontologies



Reasoning and Inference in Logic

- **Logical inference** is the process of deriving new facts from known ones using formal rules
 - Used to make decisions and answer questions based on a Knowledge Base
- **Knowledge base (KB)**:
 - A structured set of facts and rules used for logical reasoning
- **Inference engine**:
 - Mechanism that applies logical rules to a KB to derive conclusions or answer queries
 - **Forward chaining**:
 - Starts with known facts and applies inference rules to extract more data
 - E.g., given $A \rightarrow B$ and A , infer B
 - **Backward chaining**:
 - Begins with a goal and works backward to find supporting facts
 - E.g., to prove B , check if $A \rightarrow B$ and then prove A
 - **Resolution**:
 - A complete inference rule for propositional and first-order logic
 - Useful in automated theorem proving
 - **Entailment ($KB \models \alpha$)**:
 - Sentence α is entailed by KB if it is true in all models where KB is true

Rule-Based Systems (1/2)

- A rule-based system uses “if-then” rules to derive conclusions or make decisions
 - It mimics human decision-making by applying logical rules to a set of facts
- **Key components**
 - Knowledge base: stores facts and rules
 - Inference engine: applies rules to known facts to infer new facts or take actions
 - Working memory: holds current facts being considered
- **How it works**
 - *Match*: find rules whose conditions match current facts
 - *Conflict resolution*: decide which rule to apply if multiple rules match
 - *Act*: apply the chosen rule to modify facts or trigger actions
 - *Repeat*: continue until no more rules can be applied
 - **E.g.,**
 - *Rule*: If a patient has a fever and a rash, then suggest measles
 - *Fact*: Patient has a fever and a rash
 - *Conclusion*: Suggest measles

Rule-Based Systems (2/2)

- **Pros**

- Easy to modify and update rules
- Transparent and explainable reasoning
- Good when expert knowledge can be clearly articulated

- **Cons**

- Hard to scale to very large or complex domains
- Cannot handle uncertainty without extensions (e.g., probabilistic reasoning)
- Rule conflicts and maintenance can become challenging

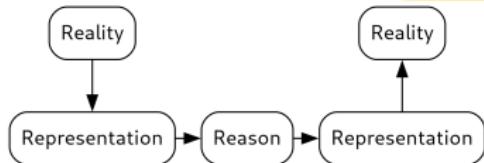
- **Applications**

- Expert systems
 - E.g., medical diagnosis, technical troubleshooting
- Business rule engines
- Game AI
- Tools for legal reasoning

Grounding

- **Grounding**

- Connect abstract symbols to real-world entities or observations
- E.g., link Apple to the fruit "apple"
- Make representations meaningful beyond syntax
 - Enable agents to act meaningfully in the real world
 - Avoid purely symbolic manipulation without real-world relevance



- **Challenges**

- Noisy, incomplete sensory data
- Complex, context-dependent mapping from inputs to concepts

- **Applications**

- Robotics: object recognition, manipulation
- Natural language understanding
- Autonomous agents, cognitive systems

- Knowledge Representation
 - Basics of Knowledge Representation
 - Examples of Logic
 - *Logical Agents*
 - Ontologies
 - Reasoning in Ontologies
- Propositional logic
- First-order Logic
- Non-classical Logics

Reflex Agents

- **Reflex agents** act based on the current percept, ignoring percept history
 - Operate using a condition-action rule: “*if condition, then action*”
 - Rely on predefined rules
 - Have no internal state or memory
 - E.g., a thermostat: “*if temperature < threshold, turn on the heater*”
- **Pros**
 - Fast and efficient in well-defined environments
- **Cons**
 - Struggle with complex or partially observable environments
 - Cannot plan ahead or learn from experience
- **Application**
 - Simple or fully observable environments where quick reactions are sufficient

Knowledge-based Agents

- **Intelligence** is achieved by *reasoning* on an internal *representation of knowledge*
- **Knowledge-based agents:**
 - Form representations of a complex world
 - Use inference to derive new representations
 - Deduce actions from new representations
 - Accept tasks as goal descriptions
 - Achieve competence by learning new knowledge
 - Adapt to changes by updating knowledge
 - Utilize a knowledge base to store information
 - Explain actions based on knowledge
 - E.g., medical diagnosis system infers diseases, suggests treatments
 - E.g., chess program uses move database to plan strategy
 - Handle incomplete or uncertain information through probabilistic reasoning

Logic / Knowledge Base (1/2)

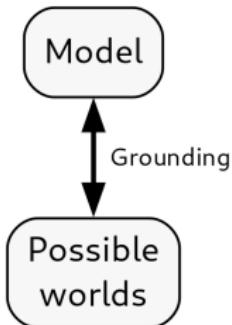
- **Knowledge base (KB)** is a set of:
 - **Sentences** α expressing assertions (observed, assumed or derived) about the world
 - E.g., "it rains", "the ground is dry", "the ground is wet"
 - **Rules**
 - E.g, "If it rains, the ground gets wet"
- **Knowledge representation language** is a formal way of creating sentences about the world
- **Syntax** specifies all the sentences α that are well-formed in a logic / knowledge base
 - E.g., in arithmetic the sentence:
 - " $x + y = 4$ " is well-formed
 - " $x4y+$ " is not well-formed
- **Semantics** is the meaning of sentences (i.e., their truth) with respect to each possible world
 - E.g., the sentence $x + y = 4$
 - Is true in the world (model) in which $x = 2, y = 2$
 - Is false in the world $x = 1, y = 1$

Logic / Knowledge Base (2/2)

- **Axiom** is a sentence taken as given
 - Not derived from other sentences
- **Inference** is the process of deriving new sentences from old ones
 - It should be done in a “logical” way
- **Truth values** of a sentence
 - In most logics every sentence is either true or false
 - **Fuzzy logic** allows sentences to have different degrees of truth
 - $\text{Belief}(\alpha) = 0.5$
 - **Probabilistic logic** allows sentences to have different probability of being true
 - $\text{Pr}(\alpha) = 0.3$

Model and Possible Worlds: Examples

- **Example:** world where there is rain and wet ground
 - In each possible world/model, values are assigned to all relevant variables
 - “Possible worlds” can be thought of as real environments
 - Model m is a mathematical abstraction of “possible world”
 - E.g., m is $(Rain = F, WetGround = T)$
 - Each possible world is a complete assignment of truth values to all relevant propositions
 - World 1: $(Rain = T, WetGround = T)$
 - World 2: $(Rain = T, WetGround = F)$
 - World 3: $(Rain = F, WetGround = T)$
 - World 4: $(Rain = F, WetGround = F)$
- **Example:** represent worlds with “men and women sitting at a table”
 - Model represents all possible worlds as $(x \text{ men}, y \text{ women})$
 - Sentence $x + y = 4$ is true in certain worlds, false in others
 - In worlds with $x = 2$ men and $y = 2$ women, $x + y = 4$ is true



Satisfaction of a Sentence in a Model

- A model m fixes all the variables x_1, \dots, x_n used in sentences
 - E.g., $(Rain = T, WetGround = T)$
- If a sentence α is true in model m , we say “*the model m **satisfies** the sentence α* ”
 - E.g., the model $(Rain = T, WetGround = F)$ satisfies $\alpha : Rain = T$
 - Note: this seems backwards, since in our common way of reasoning, the world is fixed and sentences are evaluated as true or false
- $M(\alpha)$ is the set of all the models in which α is true
 - E.g.,
 - $\alpha : Rain = T$
 - $M(Rain = T) = \{(Rain, WetGround), (Rain, \neg WetGround)\}$

Logical Entailment

- **Logical entailment** between sentences is the fact that a sentence follows logically from another sentence in a KB
- “ α **entails** β ” (written $\alpha \models \beta$) iff (by def) in every model in which α is true, β is also true
 - Equivalent to $M(\alpha) \subseteq M(\beta)$
- E.g., in the “rain and wet ground” world
 - $\alpha : "Rain \implies WetGround"$ entails $\beta : "(Rain = T, WetGround = T)"$
- E.g., in the “sitting table” world
 - α : “ $x = 0$ ”, β : “ $x \cdot y = 0$ ”
 - α entails β since in any model in which $x = 0$ is true, also $x \cdot y = 0$ is true, regardless of the value of y
- **Intuition:**
 - Entailment is not related to a proof, it just “preserves truth” across all models
 - *“If you believe your KB, you must believe the entailed sentences”*

Logical Entailment vs Implication

- Entailment and implication are related but distinct
 - **Logical entailment** is about truth following from known facts
 - **Implication** is about a relationship between two statements
- **Logical entailment** ($KB \models \alpha$):
 - Means α is always true in any world where KB is true
 - E.g.,
 - KB: "It is raining", "If it rains, the ground is wet"
 - Entailed: "The ground is wet"
- **Implication** ($A \implies B$):
 - A statement in logic that says: "If A is true, then B is true"
 - Doesn't guarantee A or B is true by itself
 - Implication is true unless A is true and B is false
 - E.g.,
 - A : "It is raining", B : "The ground is wet"
 - $A \implies B$ is the statement "If it is raining, then the ground is wet"
 - This statement can be true even if it's not raining
- **Intuition:**
 - Entailment is "meta-level truth-following"
 - Implication is "within the logic"

Model Checking Procedure

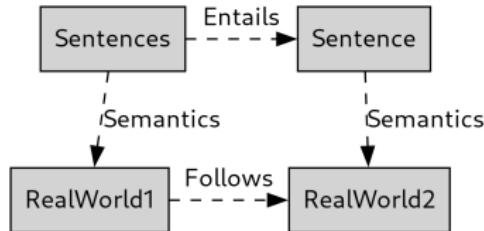
- $M(KB)$ represents all the models / possible worlds that are true given our KB
- **Problem:**
 - We want to verify whether “a sentence α is entailed by KB ” ($KB \models \alpha$)
- **Solution:**
 - According to the definition, we need to verify that α is true in all the models in which KB is true
 - I.e., $M(KB) \subseteq M(\alpha)$
- E.g., **model checking procedure** (brute force)
 1. Enumerate all the models / possible worlds
 2. Find which models are possible given the KB , i.e., $M(KB)$
 3. Check whether the sentence α is true in all the models that are compatible with the KB

Sound and Complete Inference Algorithm

- Inference: a syntactic process of deriving new sentences from others, using formal rules of a proof system (e.g., modus ponens, resolution, etc.)
 - You know: "If it rains, the ground gets wet."
 - You see: "It is raining."
 - You infer: "The ground must be wet."
- The ideal inference algorithm is both sound and complete
- **Sound** inference algorithm
 - Derives only sentences entailed from KB
 - "Whatever the inference algorithm finds, it's correct", i.e., no false positives
 - E.g., model checking is sound
 - It works only when the space of models is finite
 - When it works, it is truth preserving
- **Complete** inference algorithm
 - Can derive any sentence entailed from KB
 - "The inference algorithm doesn't miss anything," i.e., no false negatives

Isomorphism between Model and Possible Worlds

- A sound and complete inference algorithm should yield conclusions guaranteed to be true in any world where the premises are true
 -
- In other words, even if the inference operates on “syntax” (the internal representation):
 - “Sentences in the representation” correspond to “aspects of the real world”
 - “Entailment between sentences in the representation” corresponds to “implication between aspects of the real world”



Entailment vs Inference vs Implication

- **Logical entailment**

- A entails B: if the fact A is true and that automatically guarantees that fact B must also be true
- E.g., Rain entails WetGround iff in every possible world where Rain is true, WetGround is also true
 - $\text{Rain} = T, \text{WetGround} = F \rightarrow$ violation
 - Since there is at least one counterexample, Rain does not entail WetGround

- **Inference**

- This is what you (a person or a computer) figure out based on what you know
- You start with some truths, then reason your way to new truths
- It's "reasoning inside the logic system"

- **Implication**

- "If A, then B"
- It doesn't say whether A is true; it just says, if it happens, then B follows
- It's a "statement inside the logic system"

Grounding

- **Grounding** is the operation of linking abstract symbols to reality
 - E.g., words, variables in the representation ... → objects, entities, or situations in the real world
 - It is the bridge between representation in a KB and the world
- How **can we know** that a KB accurately reflects the real world?
 - We can't be sure!
 - Do we live in a simulation? What is reality?
- We **assume** that is correct
 - Agent's sensors create a sentence in the KB when something happens in the real world
 - IF smell = burning THEN food_is_burning
 - Agent learns rules and acts
 - IF food_is_burning THEN turn_off_stove
- We assume that "learning" (going from particular cases to general cases) is typically correct
 - Learning is still fallible
 - E.g., smell = burning because maybe somebody is cooking on a grill



- Knowledge Representation
 - Basics of Knowledge Representation
 - Examples of Logic
 - Logical Agents
 - ***Ontologies***
 - Reasoning in Ontologies
- Propositional logic
- First-order Logic
- Non-classical Logics

Ontologies (in computer science)

- **Ontology:**

- Is a formal, explicit representation of a domain
- Describes the types of things that exist and how they relate to each other
 - Classes: types of things
 - Individuals: specific objects
 - Properties: how things are related

- **Examples:**

- A medical ontology defines relationships between diseases, symptoms, and treatments
- A geographical ontology describes cities, states, and countries
- Semantic web (an extension of the current web to give meaning to information)

- **Goal**

- Provide a vocabulary for a domain of knowledge
- Enable machines and humans to understand and share information consistently
- Enable reasoning about entities and their relationships

- **Related Concepts**

- Schema: database-oriented structure, often more rigid than ontologies
- Taxonomy: simpler hierarchical tree-like classification

- Knowledge base: a collection of facts and rules, sometimes built from an ontology



Ontologies: Components

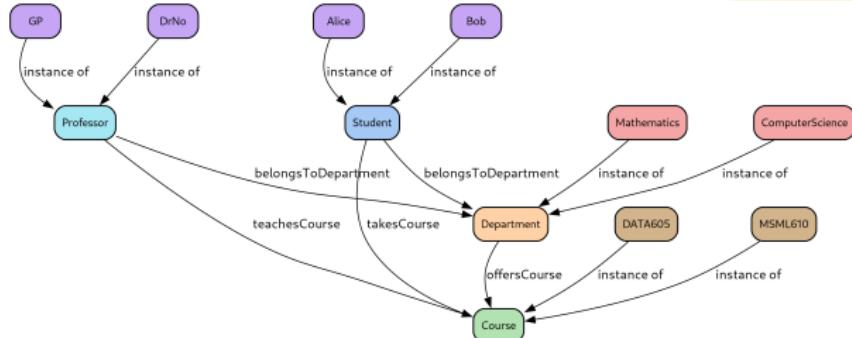
- **Classes / Concepts:**
 - Represent general concepts in a domain
 - E.g., Person, City, Car
- **Individuals / Instances:**
 - Specific, concrete examples of classes
 - E.g., GP (an instance of Person), Rome, Ferrari 458
- **Properties / Relations:**
 - Describe interactions or associations between classes or instances
 - E.g., isMortal, locatedIn, hasAge
- **Attributes / Data values**
 - Specify data associated with instances
 - E.g., (GP, hasAge, <your_guess>)
- **Constraints**
 - Rules that restrict the kinds of values a property can take
 - E.g., (Ferrari 458, mustBe, red)
- **Axioms:**
 - Logical statements that define rules and constraints
 - E.g., all humans are mortal: $\forall x(Person(x) \implies Mortal(x))$
- **Hierarchies:**
 - Organize classes and properties into parent-child relationships
 - E.g., Student is a subclass of Person



SCIENCE ACADEMY

Ontology: Example University

- **Classes**
 - Student
 - Professor
 - Course
 - Department



- **Properties:** relationships between Classes
 - takesCourse (Student → Course)
 - teachesCourse (Professor → Course)
 - belongsToDepartment (Student, Professor → Department)
- **Individuals:** examples of Classes
 - Student: Alice, Bob
 - Professor: GP, DrNo
 - Course: DATA605, MSML610
- **Axioms:** logical rules that must be true
 - Every Course must be taught by exactly one Professor
 - Every Student must belong to exactly one Department

- Knowledge Representation
 - Basics of Knowledge Representation
 - Examples of Logic
 - Logical Agents
 - Ontologies
 - *Reasoning in Ontologies*
- Propositional logic
- First-order Logic
- Non-classical Logics

Example of Reasoning Tasks (1/4)

- **Subsumption**

- “Is class A a subclass of B?”
- Check whether one concept is more general than another
- E.g., if Person subsumes Student, every Student is necessarily a Person
- Important for building taxonomies and ontologies

- **Satisfiability**

- “Can an instance of a concept exist?”
- Test if a concept is logically consistent (i.e., without contradiction)
- E.g., if the concept FlyingPenguin requires flying but is also defined as a penguin (which cannot fly), it might be not satisfiable

- **Classification**

- Organize concepts into a hierarchy
- Automatically organize concepts into a hierarchy by checking subsumption relationships
- E.g., given definitions of Animal, Bird, and Penguin, classification places Penguin under Bird, and Bird under Animal

Example of reasoning tasks in KR (2/4)

- **Instance Checking**
 - “Is a specific individual an instance of a concept?”
 - E.g., is GP an instance of Student?
- **Consistency Checking**
 - “Is the entire knowledge base free of contradictions?”
 - E.g., no Person is both Alive and Dead at the same time
- **Realization**
 - “What is the most specific class an instance belongs to?”
 - E.g., discovering that GP is a Professor rather than just a Human
- **Retrieval**
 - Find all individuals that satisfy a certain condition
 - E.g., retrieve all instances classified as TeachingAssistant

Example of reasoning tasks in KR (3/4)

- **Query Answering**

- Answer complex queries about the knowledge base
- E.g., "Find all Person that study at the university and are not Student"

- **Abduction**

- Given an observation, infer the best explanation
- E.g., seeing a Person carrying a backpack and wearing flip-flops in the snow and infer that is likely a Student

- **Deduction**

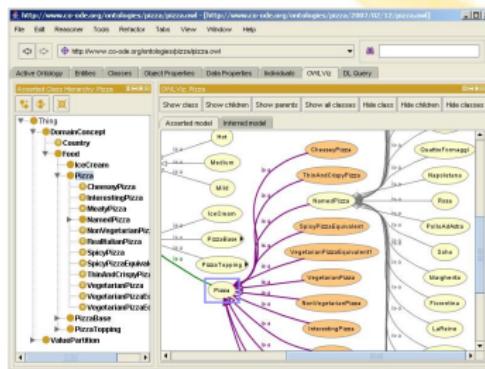
- Infer consequences that logically follow from facts and rules
- E.g., if John is a Student in ComputerScience then he can attend MSML610

Example of reasoning tasks in KR (4/4)

- **Belief Revision**
 - Update the knowledge base when new, possibly conflicting, information arrives
 - E.g., learning that not every student in ComputerScience can take MSML610 and revise a previous rule
- **Temporal Reasoning**
 - Reason about events over time
 - E.g., If EventA happens before EventB, then EventB cannot Cause EventA
- **Causal Reasoning**
 - Infer causes and effects among entities or events
 - E.g., inferring that (Storm, Cause, Flooding) based on temporal and physical knowledge

Ontologies tools: Protege Example

- **Protégé** is a free, open-source platform for building ontologies
 - Developed at Stanford
 - **Construct and visualize ontologies**
 - Users can define classes, properties, individuals, and relationships
 - **Enable reasoning** over ontologies using plugins
 - E.g., checking consistency, inferring new knowledge
 - **Supports**
 - Major ontology languages
 - OWL (Web Ontology Language)
 - RDF (Resource Description Framework)
 - Multiple serialization formats
 - RDF/XML, Turtle, OWL Functional Syntax
 - **Use cases**
 - Domain-specific knowledge modeling (e.g., biomedicine, law)
 - Semantic Web applications
 - AI systems that require structured knowledge



- Knowledge Representation
- *Propositional logic*
 - Syntax
 - Semantics
- First-order Logic
- Non-classical Logics

- Knowledge Representation
- Propositional logic
 - *Syntax*
 - Semantics
- First-order Logic
- Non-classical Logics

Propositional Logic

- **Propositional logic** is a formal system for reasoning about statements that can be true or false
 - **Syntax** defines the allowable sentences
 - Consists of proposition symbol and logical connectives
 - E.g., $P \wedge Q$
 - **Semantics** is the way in which the truth of sentences is determined
 - Truth tables or deduction rules evaluate the truth value of complex sentences
 - E.g., if P is true and Q is false then $P \wedge Q$ is false
- **Atomic representation**
 - No internal structure within atomic propositions
- **Uses**
 - SAT solvers
 - Tools for determining if a propositional logic formula can be satisfied
 - E.g., used in hardware verification and scheduling problems
 - Expert systems
 - Use logic rules to mimic human decision-making
 - E.g., medical diagnosis systems
 - Rule-based agents
 - Agents that operate based on a set of predefined rules
 - E.g., automated customer service chatbots

Proposition Symbol

- **Proposition symbol**

- Is an atomic sentence consisting of a single symbol
 - E.g., P , Q , *North*
- Doesn't have truth value, it is just a symbol for a real-world statement
 - Needs grounding
- Stands for a proposition that can be true or false
 - E.g., $K_{E,5}$ = "the Knight is in E5"
 - $K_{E,5}$ is not composed of any other symbol, it is an atomic symbol
- *True* and *False* are proposition symbols with inherent truth values

Sentences

- **Atomic sentence**

- Is a sentence composed of a single proposition symbol
- E.g., P

- **Complex sentence**

- Is constructed from simpler (sentences) using parentheses and logical connectives
- E.g., $(P \wedge Q) \vee R$
- It is a recursive definition that allow to build more complex sentences

- Common logical connectives

- Not: \neg
- And: \wedge (looks like an “A” for “and”)
- Or: \vee (comes from Latin “vel” which means “or”)
- Implies: \implies
- If and only if: \iff

- Each sentence (atomic or complex) can be only true or false

Proposition Logic: Weather Example

- Proposition symbols are:
 - *Rain* = “it’s raining”
 - *Cold* = “it’s cold”
 - *Sunny* = “it’s sunny”
 - *Snow* = “it’s snowing”
 - *Cloudy* = “it’s cloudy”
- **Atomic sentence** can be positive or negated
 - E.g., *Rain*, $\neg\text{Rain}$ = “it’s not raining”
- **Negation**
 - E.g., $\neg(\text{Rain} \vee \text{Cloudy})$ = “it’s not the case that it’s raining or cloudy”
- **Conjunction / Disjunction**
 - E.g., $\text{Rain} \wedge \text{Cold}$ = “it’s raining and it’s cold”
 - E.g., $\text{Rain} \vee \text{Snow}$ = “it’s either raining or snowing”
- **Implication** is a sentence containing a premise \implies conclusion
 - Aka “if-then statements”, “rules”
 - E.g., $\text{Rain} \implies \neg\text{Snow}$ = “if it’s raining, it’s not snowing”
- **Biconditional:** $A \implies B \wedge B \implies A$
 - E.g., $\text{Sunny} \iff \neg\text{Cloudy}$ = “it’s sunny if and only if it’s not cloudy”

Grammar in BNF form

- **Backus Normal Form** formally represents the grammar of propositional logic
- Ambiguous, i.e., the same sentence can be parsed in multiple ways
 - E.g., $\neg A \vee B = (\neg A) \vee B$ or $\neg(A \vee B)$?
- To eliminate ambiguity define the precedence for each operator
 - E.g., \neg has higher precedence than \wedge, \vee so:
 $\neg A \vee B$ means $(\neg A) \vee B$

```
Sentence → AtomicSentence | ComplexSentence
AtomicSentence → True | False | P | Q | R | ...
ComplexSentence → ( Sentence )
                  |  $\neg$  Sentence
                  | Sentence  $\wedge$  Sentence
                  | Sentence  $\vee$  Sentence
                  | Sentence  $\Rightarrow$  Sentence
                  | Sentence  $\Leftrightarrow$  Sentence
```

OPERATOR PRECEDENCE : $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

- Knowledge Representation
- Propositional logic
 - Syntax
 - *Semantics*
- First-order Logic
- Non-classical Logics

Semantics of Propositional Logic

- **Semantics** are rules for determining the truth of a sentence α with respect to a model m
 - Determine if a sentence is true or false, given a possible world
- In propositional logic, a model m fixes the truth value (true or false) for every proposition symbol/atomic sentence, e.g.,
- The models are **abstractions** of the real world and have no a-priori connection to a specific world, e.g.,
 - $P_{1,2}$ is just a symbol and can mean:
 - “There is a pit in [1, 2]” or
 - “I’m in Paris today and tomorrow”
 - Need grounding

Computing the Truth Value of a Sentence

- The **truth value of a sentence** is derived from the truth of the proposition symbols (recursively from the model m)

$$m = \{P_{1,2} = F, P_{2,2} = F, P_{3,1} = T\}$$

- All sentences α are constructed from atomic sentences (assigned by model m) and connectives:
 - $\neg P$ is T iff P is F in m
 - $P \wedge Q$ is T iff P and Q are both true in m
 - $P \vee Q$ is T iff P or Q are true in m
 - $P \implies Q$ is true unless P is true and Q is false in m
 - $P \iff Q$ is true iff P and Q are both true or both false in m
- Truth table** contains truth value of a sentence for each possible model m
 - E.g., $X = A \wedge B \vee C$

A	B	C	X
F	F	F	F
F	F	F	T



Interpretation of Implication

- In a **logical implication** “ $P \implies Q$ ” there is **no causation** between P and Q
- “ $P \implies Q$ ” says “*If P is true, I claim that Q is true. Otherwise I am making no claim*”
 - E.g., “*5 is odd implies that Tokyo is the capital of Japan*” is a true sentence in propositional logic (although very odd)
- **Pathological cases** for implication
 - An implication is true whenever the antecedent is false
 - E.g., “*5 is even implies pigs fly*” is true

Model Checking is Sound and Complete

- **Model checking algorithm**
 - Enumerate all the models (truth tables)
 - Check if α is true for every model where KB is true
- The model checking algorithm is:
 - **Sound**
 - "Any inference made by the algorithm is correct"
 - Implements the definition of entailment
 - **Complete**
 - "Any true sentence is inferred correctly by the algorithm"
 - Works for any KB and α
 - Always terminates (finite number of models)
- Complexity of model checking with n variables
 - Time complexity is $O(2^n)$ (NP-complete)
 - Worst case is exponential
 - Average case is better than exponential
 - Space complexity is $O(n)$ since enumeration is depth-first

Inference in Propositional Logic

- Inference are the rules of reasoning
 - **Modus Ponens:** if $p \implies q$ and p , infer q
 - If it rains, the ground will be wet. It rains.
 - Therefore, the ground is wet.
 - **Modus Tollens:** if $p \implies q$ and $\neg q$, infer $\neg p$
 - If it rains, the ground will be wet. The ground is not wet.
 - Therefore, it did not rain.
 - **Syllogism (Transitivity)**
 - If $p \implies q$ and $q \implies r$, then $p \implies r$
 - **Disjunctive Syllogism**
 - If $p \vee q$ and $\neg p$, infer q
 - **Addition:** if p , then $p \vee q$
 - **Simplification:** from $p \wedge q$, infer p (or q)
 - **Conjunction:** from p and q , infer $p \wedge q$
 - **Resolution Rule**
 - From $(p \vee q)$ and $(\neg p \vee r)$, infer $(q \vee r)$

Propositional Theorem Proving

- To **prove** a desired sentence α under a knowledge base KB
 - Apply rules of inference to construct a proof of α
 - Any sentence can have only one of the following truth values:
 1. True
 2. False
 3. Undecidable under the KB
- **Theorem proving vs model checking**
 - Model checking involves enumerating all models to show the sentence is true/false in all models where KB is true
 - Propositional theorem proving builds a proof
 - If the proof is short, theorem proving can be more efficient than model checking

Logical Equivalence of Sentences

- Two sentences α and β are logically equivalent $\alpha \equiv \beta$
 - Iff they are true in the same set of models:

$$M(\alpha) = M(\beta)$$

- Iff they entail each other:

$$\alpha \models \beta \wedge \beta \models \alpha$$

- E.g., $P \vee Q \equiv Q \vee P$

Logical Equivalences (1/2)

- Commutativity of \wedge and \vee

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$$

- Associativity of \wedge and \vee

$$(\alpha \wedge \beta) \wedge \gamma \equiv \alpha \wedge (\beta \wedge \gamma) \equiv \alpha \wedge \beta \wedge \gamma$$

$$(\alpha \vee \beta) \vee \gamma \equiv \alpha \vee (\beta \vee \gamma) \equiv \alpha \vee \beta \vee \gamma$$

- Distributivity of \wedge over \vee

$$\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$$

- Distributivity of \vee over \wedge

$$\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$$

- Double negation elimination:

$$\neg(\neg\alpha) \equiv \alpha$$

Logical Equivalences (2/2)

- Contraposition:

$$(\alpha \implies \beta) \equiv (\neg\beta \implies \neg\alpha)$$

- Implication elimination:

$$(\alpha \implies \beta) \equiv (\neg\alpha \vee \beta)$$

- Biconditional elimination:

$$(\alpha \iff \beta) \equiv (\alpha \implies \beta) \wedge (\beta \implies \alpha)$$

- De Morgan equivalence:

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$$

Deduction theorem

- A **valid sentence** α is true for all the models
 - E.g., $P \vee \neg P$
 - Aka “tautology”
 - Every tautology is equivalent to the sentence *True*
- **Contradiction** is a sentence α that is false for all the models
 - E.g., $P \wedge \neg P$
 - Every contradiction is equivalent to the sentence *False*
- **Deduction theorem**
 - The sentence α entails β (written $\alpha \models \beta$) iff the sentence $\alpha \implies \beta$ is a tautology
- The deduction theorem is like a **bridge** between two different but closely related ideas:
 - Entailment \models is a semantic notion
 - It's about truth across all models
 - α entails β means in every possible world where α is true, β is also true
 - Implication \implies is a syntactic notion
 - $\alpha \implies \beta$ is just another formula inside the logic

Satisfiability

- A sentence α is **satisfiable** iff α is true for some model
- **SAT problem** is about determining satisfiability of sentence in propositional logic
 - One can enumerate all the possible models until one is found to satisfy the sentence α
 - It is NP-complete
- A sentence α is **un-satisfiable** iff α is never true (i.e., a contradiction)
- **Validity and satisfiability**
 - α is valid (i.e., a tautology) iff $\neg\alpha$ is un-satisfiable
 - By contrapositive, α is satisfiable iff $\neg\alpha$ is not valid ($\neg\alpha$ is not a tautology)

Proof by Contraction

- The sentence $\alpha \models \beta$ is true iff the sentence " $\alpha \vee \neg\beta$ " is unsatisfiable (i.e., a contradiction)
- In other words in a proof by contradiction:
 - Assume α
 - Assume that the sentence β is false
 - Prove that this leads to a contradiction
 - Thus β must be true

Propositional Logic

- E.g., $P \wedge Q$
- **Pros**
 - Declarative
 - Semantics is based on relation between sentences and possible worlds
 - Can deal with partial information
 - E.g., “A white knight is in b1 or in f6” is represented with $WK1_{b1} \vee WK2_{f6}$
 - Compositional semantics
 - The meaning of a sentence is a function of the meaning of its parts
 - Context independent
 - Unambiguous
- **Cons**
 - Can't concisely describe environment with many objects, e.g.,
 - In English “*The pawn is in a cell around b1*” requires all the possible states to be enumerated
 - Can't represent uncertainty
 - “*There is 50% of probability that the pawn is in b1*”

- Knowledge Representation
- Propositional logic
- ***First-order Logic***
 - Syntax
 - Semantics
- Non-classical Logics

- Knowledge Representation
- Propositional logic
 - *Syntax*
 - Semantics
- First-order Logic
- Non-classical Logics

First-Order Logic (FOL)

- First-order logic (FOL) extends propositional logic by:
 - Introducing **quantifiers** (e.g., \forall , \exists)
 - Using **predicates** to represent properties and relations
- Combines pros of propositional logic with pros of natural language
 - Built around objects and relations
 - Allows to express facts about some or all objects, e.g.,
 - “*Some humans have green eyes*”
 - “*Chess pieces around the Queen are at risk*”
- FOL provides expressive power to represent structured, relational knowledge

First-Order Logic: Syntax

- **Constants:** represent specific objects
 - E.g., *Socrates*
- **Predicates:** describe properties or relations
 - E.g., $\text{Human}(x)$ = “*x is human*”
- **Functions:** map tuples of objects to objects
 - E.g., $\text{Mother}(x)$ = “*the mother of x*”
- **Variables:** act as placeholders
 - E.g., x, y
- **Quantifiers**
 - E.g., $\forall x$ (for all x), $\exists x$ (there exists an x)

Sentence → *AtomicSentence* | *ComplexSentence*
AtomicSentence → *Predicate* | *Predicate(Term, ...)* | *Term = Term*
ComplexSentence → (*Sentence*)
| \neg *Sentence*
| *Sentence* \wedge *Sentence*
| *Sentence* \vee *Sentence*
| *Sentence* \Rightarrow *Sentence*
| *Sentence* \Leftrightarrow *Sentence*
| *Quantifier Variable, ... Sentence*

Term → *Function(Term, ...)*
| *Constant*
| *Variable*

Quantifier → \forall | \exists
Constant → *A* | *X₁* | *John* | ...
Variable → *a* | *x* | *s* | ...
Predicate → *True* | *False* | *After* | *Loves* | *Raining* | ...
Function → *Mother* | *LeftLeg* | ...

OPERATOR PRECEDENCE : $\neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Sentences

- **Term** is a logical expression that refers to an object
 - E.g., *Richard*
- **Atomic sentence** is a predicate symbol (i.e., corresponds to relations) followed by a list of terms in parenthesis *Predicate(Term1, Term2, ...)*
 - E.g., *Brother(Richard, John)* means “*Richard is the brother of John*”, under a given grounding / interpretation,
 - E.g., *Married(Father(Richard), Mother(John))* means “*the father of Richard and the mother of John are married*”
- **Complex sentence** is a sentence using logical connectives complex, with the same syntax and semantics as in propositional logic
- **Variable** is a term that represents a possible object
 - Typically represented as lowercase letter (e.g., *x, y, z*)
 - Can be used as argument of a function, e.g., *LeftLeg(x)*
- **Equality symbol** signifies that two terms refer to the same object
 - E.g., *Father(John) = Henry*

Quantifiers

- **Quantifiers** express properties of entire collections of objects, instead of enumerating objects by name (like in propositional logic)
- **Universal quantifier:** $\forall x P(x)$
 - Universal quantifier makes a statement about *every* object
 - Statement is true if $P(x)$ is true for all x
- **Existential quantifier:** $\exists x P(x)$
 - Existential quantifier makes a statement about *some* object (without naming it)
 - Statement is true if $P(x)$ is true for at least one x
- Variables are **bound** by quantifiers or **free** (unbound)
 - E.g., $\forall x(Cat(x) \rightarrow Mammal(x))$

Nested Quantifiers

- More complex sentences can be expressed using **multiple quantifiers**
 - The order of quantifiers is important, so one can use parentheses to clarify
- **Example:**
 - "*Brothers are siblings*": $\forall x, y \text{ Brother}(x, y) \implies \text{Sibling}(x, y)$
 - *Siblings* is a symmetric relationship: $\forall x, y \text{ Sibling}(x, y) \iff \text{Sibling}(y, x)$
 - "*Everybody loves somebody*": $\forall x \exists y \text{ Loves}(x, y)$
 - "*There is someone loved by everyone*": $\exists y \forall x \text{ Loves}(x, y)$

Connection between \forall and \exists

- The universal and existential quantifiers are **connected** through negation and De Morgan rules

$$\forall x \neg P(x) \iff \neg \exists x : P(x)$$

$$\neg(\forall x P(x)) \iff \exists x \neg P(x)$$

$$\forall x P(x) \iff \neg \exists x \neg P(x)$$

$$\exists x P(x) \iff \neg(\forall x \neg P(x))$$

- Knowledge Representation
- Propositional logic
 - Syntax
 - **Semantics**
- First-order Logic
- Non-classical Logics

First-order Logic: Semantics

- **Semantics** define how sentences are interpreted in a domain
- Symbols represent entities, relationships, and functions in the domain
 - **Constant symbols** represent specific objects
 - E.g., *Alice*, *GP*, *CS101*
 - **Predicate symbols** represent relationships among objects
 - E.g., *EnrolledIn(Student, Class)*, *Teaches(Professor, Class)*, *IsStudent(x)*, *IsProfessor(x)*
 - **Function symbols** represent mappings between objects
 - E.g., *AdvisorOf(Student)*, *DepartmentOf(Professor)*
- An **interpretation** maps the world to its mathematical description, and vice versa
 - There are many possible interpretations
 - The intended interpretation is the one that is the most natural
 - E.g., map the symbol *GP* → me
 - $\forall x (\text{Human}(x) \rightarrow \text{Mortal}(x))$ means that every human is mortal

Representing Knowledge in FOL

- FOL enables representation of:
 - General rules: $\forall x (Bird(x) \rightarrow CanFly(x))$
 - Specific facts: $Bird(Tweety)$
- Complex relations captured through predicates:
 - $Loves(Romeo, Juliet)$, $GreaterThan(3, 2)$
- Functions express object construction:
 - $FatherOf(John)$
- Knowledge base is built from axioms and facts
- Enables reasoning about objects, properties, and their relationships

First-Order Logic: Inference

- **Goal:** derive new sentences from existing ones using sound rules
 - **Universal Instantiation:**
 - From $\forall x P(x)$ infer $P(c)$ for any constant c
 - **Existential Instantiation:**
 - From $\exists x P(x)$ infer $P(c)$ with a new constant c
 - Standard propositional logic rules such as Modus Ponens, Modus Tollens, Resolution, etc., also apply within FOL
 - Modus ponens: from P and $P \implies Q$, infer Q
- FOL inference is semi-decidable:
 - If a sentence is entailed, a proof can be found
 - If not entailed, proof search may not terminate

- Knowledge Representation
- Propositional logic
- First-order Logic
- ***Non-classical Logics***
 - Intro and Examples
 - Description Logics
 - Semantic Web

- Knowledge Representation
- Propositional logic
- First-order Logic
- Non-classical Logics
 - *Intro and Examples*
 - Description Logics
 - Semantic Web

Ontological Commitment

- **Ontological commitments** are assumptions about reality made by a language
- Different formal models make **different assumptions** on how the truth of sentences is defined
 - Propositional logic:
 - The world consists of facts that are either true or false
 - E.g., $P \vee Q$
 - First-order logic:
 - The world consists of objects with relations among them that hold or do not hold
 - $\forall x : Human(x) \implies Mortal(x)$
 - Temporal logic:
 - Facts about objects and relations hold at particular times or intervals
 - Higher-order logic:
 - Relations of first-order logic are objects themselves
 - E.g., can make assertions about relations (e.g., "all relations are transitive")

Epistemological Commitment

- **Epistemological commitment** is a possible state of knowledge by an agent with respect to each fact (i.e., belief), e.g.,
 - Propositional logic, first-order logic
 - 3 possible states of belief regarding any sentence: true, false, or unknown
 - Probability theory
 - There is a degree of belief in $[0, 1]$ about each sentence
 - $\Pr(X = 6) = 0.3$
- Ontological commitment = what exists in the world
- Epistemological commitment = what an agent believes about facts

Non-monotonic Logic

- **Non-monotonic logic** is a type of logic where adding new information can invalidate previous conclusions
- Monotonic (classical) vs non-monotonic logic
 - In classical logic, once something is proven, it stays proven even if more information is added
 - In non-monotonic logic, conclusions can change as new facts are learned
- **E.g.,**
 - Initial knowledge: "*Birds typically fly*"
 - Fact: "*Tweety is a bird*"
 - Conclusion: "*Tweety can fly*"
 - New fact: "*Tweety is a penguin*"
 - Fact: "*penguins are birds that can't fly*"
 - Revised conclusion: "*Tweety cannot fly*"
- E.g., "Swans are white birds" and then black swans are discovered
- Real-world situations often involve incomplete or evolving knowledge
 - Non-monotonic logic allows systems to reason flexibly and adapt to new circumstances



Default Reasoning

- **Default reasoning** is reasoning where assumptions are made by default in the absence of contrary evidence
 - It allows conclusions based on typical situations unless exceptions are found
 - Relates to “common sense”
- **Key idea**
 - Assume the most likely case unless specified otherwise
 - If new information contradicts the assumption, revise the conclusion
- **E.g.,**
 - Default rule: “*Typically, birds can fly*”
 - Fact: “*Tweety is a bird*”
 - Conclusion: “*Tweety can fly*”
 - New fact: “*Tweety is a penguin*”
 - Revised conclusion: “*Tweety cannot fly*”
- **Why it is useful**
 - Default reasoning allows systems to function reasonably without knowing everything

Non-Monotonic Logic: University Example

- **Initial facts**

- Alice is a *Student*
- Alice belongs to the *ComputerScience* department
- CS101 is a *Course* offered by the *ComputerScience* department
- Default rule: *Students* in the *ComputerScience* department take classes in their department

- **Initial reasoning**

- Since Alice is a *Student* in *ComputerScience*, by default *Students* take *CS101*
- Conclusion: *Alice takesCourse CS101*

- **New information**

- Alice is a student who does not meet the prerequisites for *CS101*

- **Revised reasoning**

- New conclusion: *Alice does not takeCourse CS101*

Common Sense Reasoning

- **Common sense reasoning** is the ability to make assumptions, draw conclusions based on everyday knowledge about the world
 - Involves typical, unstated knowledge that humans take for granted, e.g.,
 - *If you drop a glass, it will likely break*
 - Knowing that *“people eat food when they are hungry”* without being explicitly told
- **Characteristics**
 - Deals with incomplete, uncertain, or ambiguous information
 - Relies on defaults, heuristics, and typical patterns rather than strict logical proofs
 - Often flexible and tolerant of exceptions
- **Challenges**
 - Common sense knowledge is vast, informal, and often not precisely defined
 - Difficult to encode all of it explicitly in a machine-readable form
 - Handling exceptions and contradictions is complex
- **Techniques**
 - Knowledge graphs
 - Non-monotonic logic
 - Probabilistic reasoning
 - Machine learning models trained on large, diverse data

Common Sense Reasoning: University Example

- **Initial facts**

- Alice is a *Student*
- Bob is a *Student*
- CS101 is a *Course* offered by the *ComputerScience* department

- **Common sense knowledge**

- *Students* typically *Enroll* in *Courses* offered by their *Department*
- *Students* usually *attend* *Course* they are enrolled in
- *Professors* usually teach the *Courses* they are assigned

- **Reasoning steps**

- Alice belongs to the *ComputerScience* department
- CS101 is offered by the *ComputerScience* department
- Common sense suggests Alice is likely enrolled in CS101, even if enrollment is not explicitly stated
- Therefore, it is reasonable to assume "Alice takesCourse CS101"

- **New information**

- Alice is pursuing research only and not taking courses
- The assumption that Alice takesCourse CS101 must be revised

Open World vs Closed World Assumptions

- **Closed World Assumption (CWA)**

- Missing information is false
- E.g.,
 - Fact: “Alice takes CS101” is known
 - Nothing is said about Bob
 - Under CWA: Conclude Bob does not take CS101

- **Open World Assumption (OWA)**

- Missing information is unknown, not false
- E.g.,
 - Fact: “Alice takes CS101” is known
 - Nothing is said about Bob
 - Under OWA: Can't conclude if Bob takes CS101 or not it is unknown

- **Applications**

- CWA
 - Traditional relational databases (SQL)
 - Logic programming
 - Business rules and systems requiring complete data
- OWA
 - Semantic Web (RDF, OWL)
 - Knowledge representation with incomplete or growing data

Inductive Logic Programming

- **Inductive logic programming**

- Learns logical rules from examples and commonsense knowledge
- Given positive and negative examples, and background facts, infer logical rules that explain the examples

- **Example**

- Commonsense knowledge: "*Birds have wings*"
- Positive example: "*Tweety is a bird that can fly*"
- Negative example: "*Penguin cannot fly*"
- Learned rule: "*Birds can fly unless they are penguins*"

- **Features**

- Produces human-readable logical rules
- Integrates learning with symbolic reasoning
- Supports background knowledge integration

- **Challenges**

- Computational complexity with large datasets
- Handling noisy, incomplete, or ambiguous data

- Knowledge Representation
- Propositional logic
- First-order Logic
- Non-classical Logics
 - Intro and Examples
 - **Description Logics**
 - Semantic Web

Description Logic

- **Description logic**

- Represents structured knowledge about a domain
- Balances expressivity and computational efficiency
- More expressive than propositional logic, less than first-order logic

- **Core building blocks:**

- Classes: abstract groups
 - E.g., *Person*, *Animal*
- Properties: binary relations between individuals
 - E.g., *hasChild*, *ownsPet*
- Instances: specific objects
 - E.g., *GP*, *Nuvolo*

- Supports reasoning tasks such as:

- Concept subsumption: “*Is class A a subset of class B?*”
- Instance checking: “*Does instance a belong to class A?*”

- **Syntax often combines:**

- Atomic concepts and roles
- Logical constructors (\sqcap , \sqcup , \neg , \forall , \exists)
- E.g.,
 - $Father \equiv Man \sqcap \exists hasChild . Person$

- Widely used in ontologies, e.g., OWL (Web Ontology Language)

- **Attributive Concept Language with Complements (ALC)** is a basic but expressive description logic
 - Concepts can be combined using logical operators, e.g.,
 - \sqcap means “and”
 - \sqcup means “or”
 - \neg means “not”
 - Allows for existential and universal quantification, e.g., $\exists R.C$, $\forall R.C$
 - Interpretation is set-theoretic
 - Classes as sets, properties as binary relations
- **Examples**
 - “All students take some course”: $\text{Student} \equiv \exists \text{takes}. \text{Course}$
 - “A mother is a woman who has at least one child”:
 $\text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild}. \top$
- ALC:
 - Is decidable
 - Balances expressiveness and computational complexity
 - Is the basis for more complex logics used in OWL



SHOIN

- SHOIN is a description logic more expressive than ALC
- Components:
 - \mathcal{S} : Allows transitive properties
 - E.g., *ancestorOf* is transitive
 - \mathcal{H} : Supports role hierarchies
 - E.g., *hasSon* \sqsubseteq *hasChild*
 - \mathcal{O} : Introduces specific individuals
 - E.g., *John* is a nominal class
 - \mathcal{I} : Enables roles to be navigated backward
 - E.g., *isChildOf* is inverse of *hasChild*
 - \mathcal{N} : Sets cardinality constraints
 - E.g., “*has exactly 1 children*”

• Examples

- “*Exactly two children*”: $\text{Person} \sqsubseteq (= 2 \text{ hasChild.} \top)$

• Characteristics

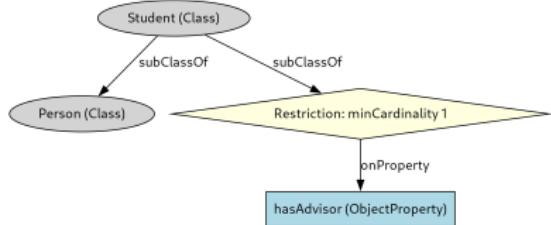
- More powerful but reasoning is harder (exponential complexity)
- Model richer real-world scenarios

SCIENCE
ACADEMY Foundation for OWL DL reasoning capabilities



- **Web Ontology Language (OWL):**
 - “OWL” easier to pronounce than “WOL”
 - Semantic web language designed to represent complex knowledge about things and their relationships
 - Enables rich knowledge representation on the web (based on SHOIN)
 - Supports formal semantics for machine reasoning
 - Key constructs:
 - Classes, properties, individuals, axioms
- **Example**
 - “Every cat is a mammal”: $\text{Cat} \sqsubseteq \text{Mammal}$
- **OWL variants:**
 - OWL Lite: simpler, for classification hierarchies
 - OWL DL: full expressiveness with decidable reasoning
 - OWL Full: maximum expressiveness, but undecidable
- **Applications**
 - Semantic search
 - Biomedical data

Example of OWL in RDF



```
1. ````  
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
3.   xmlns:owl="http://www.w3.org/2002/07/owl#"  
4.   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"  
5.   xmlns:ex="http://example.org/">  
6.   <owl:Class rdf:about="http://example.org/Person"/>  
7.   <owl:Class rdf:about="http://example.org/Student">  
8.     <rdfs:subClassOf rdf:resource="http://example.org/Person"/>  
9.     <rdfs:subClassOf>  
10.       <owl:Restriction>  
11.         <owl:onProperty rdf:resource="http://example.org/hasAdvisor"/>  
12.         <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegat  
13.           1  
14.         </owl:minCardinality>  
15.       </owl:Restriction>  
16.     </rdfs:subClassOf>  
17.   </owl:Class>  
18.   <owl:ObjectProperty rdf:about="http://example.org/hasAdvisor"/>  
19. </rdf:RDF>  
20. ````
```

RDF (Resource Description Framework)

- **Resource Description Framework (RDF)** is a standard model for data interchange on the web
 - Represent structured information in a machine-readable way
- **Basic building block** is a triple:
 - *Subject*: the entity being described, e.g., Nuvolo
 - *Predicate*: the property or relationship, e.g., isA
 - *Object*: the value or another entity, e.g., Dog

Subject	Predicate	Object
Book123	hasTitle	"The Great Gatsby"
Book123	hasAuthor	Author456
Author456	hasName	"F. Scott Fitzgerald"
Book123	publishedYear	"1925"
Book123	belongsToGenre	"Fiction"

- **Key Features:**
 - Statements are directed graphs of nodes and edges
 - Components are URIs (Uniform Resource Identifiers) to ensure global uniqueness or literals
 - E.g., <http://example.org/Nuvolo>
- **Use Cases:**
 - Building knowledge graphs
 - Enabling semantic search
 - Supporting ontologies (e.g., OWL)



SPARQL

- SPARQL is the query language for RDF data
 - Allows users to retrieve and manipulate data stored in RDF format
- **Key Concepts:**
 - *Triple Patterns*: Query fragments that match triples in an RDF graph
 - *Basic Graph Pattern*: A set of triple patterns combined
 - *Variables*: Stand in for unknown parts of the triples
 - E.g., ?person, ?animal
- **Main Query Types:**
 - SELECT: Retrieve specific variables from the data
 - CONSTRUCT: Create new RDF triples based on query results
 - ASK: Return a boolean indicating whether a pattern exists
 - DESCRIBE: Return an RDF graph describing resources
- **Example:**
 - “Find all resources that are of type Bird”
`SELECT ?animal WHERE { ?animal rdf:type ex:Bird }`

- Knowledge Representation
- Propositional logic
- First-order Logic
- Non-classical Logics
 - Intro and Examples
 - Description Logics
 - **Semantic Web**

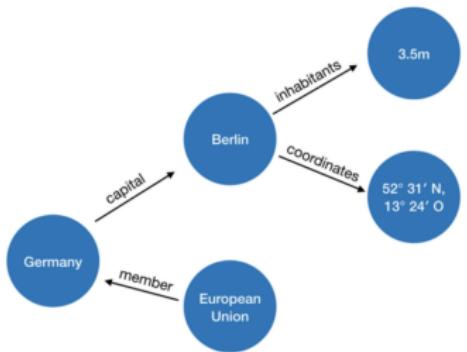
Semantic Web

- The **Semantic Web** extends the current Web by enabling machines to understand and interpret data
 - HTML is human-readable but lacks semantic structure for computers
 - The Semantic Web adds meaning / semantics to data
 - Allow better data integration, automation, and discovery across sites
- **Key Technologies**
 - RDF (Resource Description Framework): base data model
 - SPARQL: query language for RDF data
 - OWL (Web Ontology Language): define rich ontologies
- **Current Status**
 - Some core ideas (e.g., structured data and ontologies) are widely adopted
 - Full vision remains only partially realized
- **Challenges**
 - Complexity of widespread adoption
 - Issues around privacy, data ownership, and feasibility
 - Need for standardization and tools
- **Criticism**
 - Skepticism about practicality and scalability
 - Concerns about centralization and censorship

WikiData

- **WikiData** is a free, open, collaborative knowledge base
 - Stores structured data for Wikipedia
 - Accessible via APIs using SPARQL queries
- **Graph-based data model**
 - *Item*: represents an entity or concept
 - Q42 → Douglas Adams
 - *Property*: describes a relationship or attribute
 - P31 (instance of), P27 (country of citizenship)
 - *Value*: specific data linked to an item via a property
 - Q42 (Douglas Adams) → P31 (instance of) → Q5 (human)
 - Q42 → P106 (occupation) → Q36180 (science fiction writer)
 - *Reference*: supports a claim by citing a source
 - Stating Douglas Adams's citizenship with a reference to a biography
 - *Qualifier*: adds context or additional information to a statement
 - Q90 (Paris) → P1082 (population) → "2,165,423"
 - With qualifier: P585 (point in time) → "2021"
 - Meaning: "The population of Paris was 2,165,423 in the year 2021"
- **Applications:**
 - Knowledge graph
 - Semantic search
 - AI reasoning
 - Data enrichment

- DBpedia extracts structured content from Wikipedia
 - Creates a large-scale, multilingual knowledge graph for querying
 - Data is extracted as RDF triples (*Subject, Predicate, Object*)
 - E.g., “Berlin” entity linked with properties like dbo:country Germany, dbo:populationTotal 3.5M
 - Enables semantic queries over Wikipedia data via SPARQL endpoints
- Applications
 - Semantic Web
 - Enhancing AI models with real-world knowledge

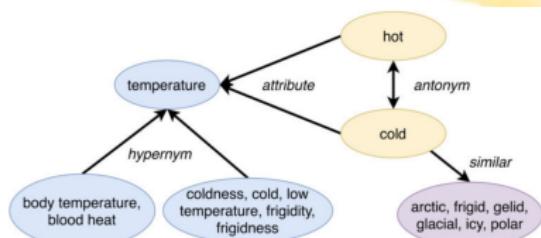


Semantic Networks

- **Semantic Networks** represent knowledge as graphs of concepts and relations
 - Nodes represent concepts
 - Edges represent relations (e.g., “is-a”, “part-of”)
 - E.g., if a Dog is an Animal, it inherits Animal traits
 - Examples: WordNet, ConceptNet
- **Pros**
 - Easy to visualize and traverse
 - Support reasoning
 - Common in early AI systems and current KG applications

WordNet

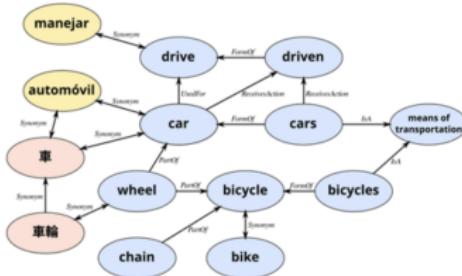
- WordNet is a large lexical database of English words
 - Models semantic relationships between words
 - Manually curated for high-quality semantic relations
 - Incomplete for domain-specific language



- Graph with synsets as nodes and relations as edges
 - Synsets are sets of synonyms expressing a distinct concepts
 - E.g., {car, automobile} share the same synset
 - Relations between synsets
 - Is-a relationships (e.g., Dog is a type of Animal)
 - Part-whole relationships (e.g., Wheel is a part of Car)
 - Opposite meanings
- Applications
 - Word sense disambiguation: choose correct word meaning in context
 - Semantic similarity measures: assess concept closeness
 - Information retrieval and question answering systems

ConceptNet

- ConceptNet is a large knowledge graph
 - Connects words and phrases with labeled semantic relationships
 - Represents commonsense knowledge about the world
- Structure
 - Nodes: concepts (words or phrases)
 - Edges: semantic relationships between concepts, e.g.,
 - IsA: (dog, animal)
 - PartOf: (wheel, car)
 - UsedFor: (knife, cutting)
 - CapableOf: (bird, fly)
 - Causes: (fire, smoke)
 - E.g., (bicycle, UsedFor, transportation)
- Applications
 - Natural language understanding
 - Question answering and chatbots
 - Commonsense reasoning in AI
 - Semantic search and recommendation systems



Knowledge Graphs

- Knowledge Graphs (KGs) represent entities and their relationships as a graph structure
 - Nodes = entities
 - Edges = relations
 - E.g., “Paris → isCapitalOf → France”
- Query languages like SPARQL allow expressive information retrieval
- KGs support reasoning via path traversal and schema inference
- Applications:
 - Question answering
 - Recommendation
 - Semantic search
- Widely used by Google, Facebook, and academic search engines

Knowledge Graph: University Example

