MSML610: Advanced Machine Learning

Knowledge Representation

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

References:

- Mostly papers and Internet
- AIMA 7: Logical agents
- AIMA 8, First-order logic
- AIMA 9: Inference in first-order logic
- AIMA 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

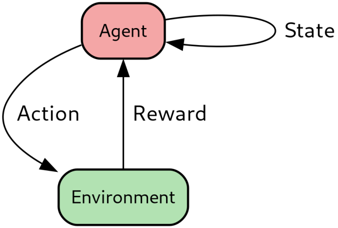
- Knowledge Representation
    - ***Basics of Knowledge Representation***
    - Examples of Logic
    - Logical Agents
    - Ontologies
    - Reasoning in Ontologies
- Propositional logic
- First-order Logic
- Non-classical Logics

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)

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
  - Declarative knowledge can be compiled into procedural code
  - E.g., a planner generates procedures (plans) from declarative goals

Propositional Logic

- Uses atomic statements (propositions) and logical connectives
  - **Syntax**
    - Atomic formulas: P, Q
    - Connectives: NOT ($\neg$), AND ($\wedge$), OR ($\vee$), IMPLIES ($\implies$)
  - **Semantics**
    - Based on truth tables
    - Each proposition has a binary truth value: true or false
  - **Inference mechanisms**
    - Modus ponens: from P and $P \implies Q$, 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

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

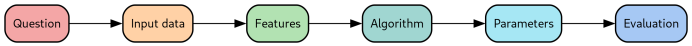
Rule-Based Systems (2/2)

- **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
  - Legal reasoning tools

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

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

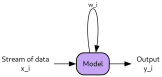
Logic / Knowledge Base (1/2)

- **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
    - $\mathit{Belief}(\alpha) = 0.5$
  - **Probabilistic logic** allows sentences to have different probability of being true
    - $\mathrm{Pr}(\alpha) = 0.3$

Model and Possible Worlds: Examples



- **Example**: represent worlds 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 the 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 men, y women)
  - Sentence $x + y = 4$ is true in certain worlds, false in others
  - In worlds with $x = 2$ men and $y = 2$ women,

Satisfaction of a Sentence in a Model

- A model m fixes all the variables $x_1, ..., x_n$ 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
  - α : "$Rain \implies WetGround$" entails β : "$(Rain = T, WetGround = T)$"
- 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

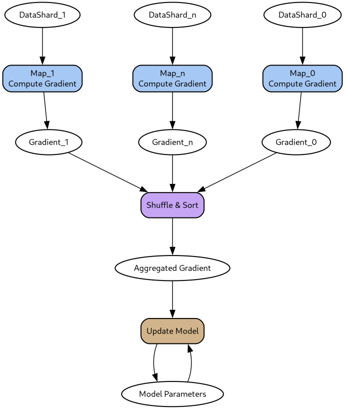
- $M(KB)$ represents all the models / possible worlds that are true given our KB
- **Problem**:
  - We want to verify whether "a sentence α is entailed by KB" ($KB \models \alpha$)
- **Solution**:
  - According to the definition, we need to verify that α is true in all the models in which KB is true
    - 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
    - $Rain = T, WetGround = F \rightarrow$ 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 ... $\rightarrow$ 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`

Ontology: Example University

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

	$\text{pred} = 1$	$\text{pred} = 0$
$\text{act} = 1$	**TP**	**FN**
$\text{act} = 0$	**FP**	**TN**

- **Properties**: relationships between Classes

- 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`

E.g., 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
- Provides tools to 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
    - Systems that 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
  - 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
- Complex sentence:
  - Is constructed from simpler (sentences) using parentheses and logical connectives
  - Note: it is a recursive definition that allow to build more complex sentences
- Each sentence (atomic or complex) can be only true or false
- 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: $\iff$

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 (e.g., $Rain$) or negated (e.g., $\neg Rain$ = "it's not raining")
- Negation
  - E.g., $\neg(Rain \vee Cloudy)$ = "it's not the case that it's raining or cloudy"
- Conjunction
  - E.g., $Rain \wedge Cold$ = "it's raining and it's cold"
- Disjunction
  - E.g., $Rain \vee Snow$ = "it's either raining or snowing"
- Implication is a sentence containing a premise (aka antecedent), the connective $\implies$, and a conclusion (aka consequent)

Grammar in BNF form

- Use BNF to formally represent 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 $(\neg A) \vee B$

- 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
  - We want to 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"

Computing the truth value of a sentence

- The truth value of a sentence can be derived from the truth of the proposition symbols (recursively from the model m), e.g.,
- If the KB is based on proposition symbols $P_{1,2}, P_{2,2}, P_{3,1}$:

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

- All sentences α are constructed from atomic sentences (assigned by the model m) and the five 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 the truth value of a sentence (no matter how complex) for each possible assignment of truth values to its components
  - 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
  - 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
    - E.g., "5 is even implies Sam is smart" is true, even if Sam is not smart
    - The reason is that $P \implies Q$ is saying "If P is true, I claim that Q is true. Otherwise I am making no claim"

Model Checking is Sound and Complete

- **Model checking algorithm:**
  - Enumerate all 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

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
  - 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:
$$(\alpha \implies \beta) \equiv (\neg\alpha \lor \beta)$$
- Biconditional elimination:
$$(\alpha \iff \beta) \equiv (\alpha \implies \beta) \land (\beta \implies \alpha)$$
- De Morgan:
$$\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta)$$
$$\neg(\alpha \lor \beta) \equiv (\neg\alpha \land \neg\beta)$$

Valid sentence

- 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 contraction 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, i.e., is equivalent to *True*

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 un-satisfiable (i.e., a contradiction)
- In other words in a proof by contradiction:
  - Assume α
  - Assume that the sentence β is false and
  - Prove that this leads to a contradiction
  - Thus β must be true

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

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

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**
  - Understanding of the world is influenced by language
  - Language influences thought (even through arbitrary grammatical features, e.g., gender of nouns)
  - Some languages lack words for certain concepts (e.g., direction)

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
- **Cons**:
  - Programming is procedural (vs declarative)
  - 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

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 `b6`" requires all the possible states to be enumerated

First-Order Logic (FOL): Intro

- First-order logic (FOL) extends propositional logic by:
  - Introducing quantifiers ($\forall$, $\exists$)
  - 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 blue eyes"
    - "Squares neighboring the Wumpus are smelly"
- 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., $Human(x)$)
- **Functions**: map tuples of objects to objects (e.g., $Mother(x)$)
- **Variables**: placeholders (e.g., x, y)
- **Quantifiers**: $\forall x$ (for all x), $\exists x$ (there exists an x)

Sentences

- Term is a logical expression that refers to an object in a FOL model
- Atomic sentence = predicate symbol (i.e., which corresponds to relations) followed by a list of terms in parenthesis (i.e., constant or function symbol) $Predicate(Term1, Term2, ...)$
  - E.g., $Brother(Richard, John)$, under the model / interpretation, Richard is the brother of John
  - E.g., $Married(Father(Richard), Mother(John))$
- Complex sentences = sentences 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 and Scope

- 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)
  - True if $P(x)$ is true for at least one x
- Scope determines the portion of a formula a quantifier applies to
- Variables are **bound** by quantifiers or **free** (unbound)
- Sentences with no free variables are called **closed formulas**
- Example:
  - $\forall x(Cat(x) \rightarrow Mammal(x))$

Nested quantifiers

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

Connection between $\forall$ and $\exists$

- The two quantifiers are connected through negation and De Morgan rules

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

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

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

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

- 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 $GP \rightarrow$ me
- Example:
  - Sentence: $\forall x\, (Human(x) \rightarrow Mortal(x))$
  - True if for every x in the domain, $Human(x)$ implies $Mortal(x)$

Inference in First-Order Logic

- 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
- **Modus Ponens** and other propositional rules apply
- FOL inference is semi-decidable:
  - If a sentence is entailed, a proof can be found
  - If not entailed, proof search may not terminate

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 built from axioms and facts
- Enables reasoning about objects, properties, and their relationships

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

- Knowledge Representation
- Propositional logic
- First-order Logic
- Non-classical Logics
    - ***Intro***
    - 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
    - First-order logic:
        - The world consists of objects with relations among them that hold or do not hold
    - 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 states of knowledge by an agent with respect to each fact
  - Ontological commitment = what exists in the world
  - Epistemological commitment = what an agent believes about facts
- 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

Non-monotonic Logic

- Non-monotonic logic is a type of logic where adding new information can invalidate previous conclusions
- Contrast with Classical (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"
  - Conclusion: "Tweety is a bird, so Tweety can fly"
  - New information: "Tweety is a penguin"
  - Revised conclusion: "Tweety cannot fly"
- **Why it Matters**
  - 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
- **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**
  - In real life, information is often incomplete or uncertain
  - 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
  - *CS*101 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 *CS*101
  - Conclusion: *Alice takesCourse CS*101
- **New Information**
  - *Alice* is an exchange student who does not meet the prerequisites for *CS*101
- **Revised Reasoning**
  - New conclusion: *Alice* does not *takeCourse CS*101

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*
  - *CS*101 is a *Course* offered by the *ComputerScience* department
- **Common sense knowledge**
  - Students typically enroll in courses offered by their department
  - Students usually attend classes they are enrolled in
  - Professors usually teach the courses they are assigned
- **Reasoning steps**
  - *Alice* belongs to the *ComputerScience* department
  - *CS*101 is offered by the *ComputerScience* department
  - Common sense suggests *Alice* is likely enrolled in *CS*101, even if enrollment is not explicitly stated
  - Therefore, it is reasonable to assume: *Alice takesCourse CS*101
- **New information**
  - *Alice* is pursuing research only and not taking courses
  - The assumption that *Alice takesCourse CS*101 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
  - Common in databases and logic programming
- **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: Cannot conclude if Bob takes CS101: it is unknown
  - Common in Semantic Web, RDF, ontologies
- **Applications**
  - OWA
    - Semantic Web (RDF, OWL)
    - Knowledge representation with incomplete or growing data
  - CWA
    - Traditional relational databases (SQL)
    - Business rules and systems requiring complete data

Inductive Logic Programming

- **Inductive Logic Programming**
  - Learns logical rules from examples and common sense knowledge
  - Given positive and negative examples, and background facts, infer logical rules that explain the examples
- **Example**
  - Background: “Birds have wings”
  - Positive example: “Tweety 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
    - ***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:**
  - Concepts / classes: abstract groups
    - E.g., *Person*, *Animal*
  - Roles / properties: binary relations between individuals
    - E.g., *hasChild*, *ownsPet*
  - Individuals / instances: specific objects
    - E.g., *GP*, *Nuvolo*
- Supports reasoning tasks such as:
  - Concept subsumption: "is A a subset of B?"
  - Instance checking: "does a belong to A?"
- **Syntax often combines:**
  - Atomic concepts and roles
  - Logical constructors ($\sqcap$, $\sqcup$, $\neg$, $\forall$, $\exists$)
  - E.g.,
    - $\mathit{Father} \equiv \mathit{Man} \sqcap \exists \mathit{hasChild}.\mathit{Person}$
- Widely used in ontologies, e.g., OWL (Web Ontology Language)

ALC

- 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
  - Concepts as sets, roles as binary relations
- Example:
  - "All students take some course": $\text{Student} \sqsubseteq \exists \text{takes.Course}$
  - "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 basis for more complex logics used in OWL
  - Practical for moderate-sized ontologies

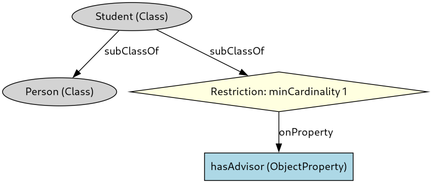
SHOIN

- 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 passport"
- E.g.,:
  - "Exactly two children" Person $\sqsubseteq$ $(= 2\ \text{hasChild}.\top)$
- Characteristics
  - More powerful but reasoning is harder (exponential complexity)
  - Model richer real-world scenarios
  - Foundation for OWL DL reasoning capabilities

OWL

- OWL = Web Ontology Language
  - Semantic web language designed to represent complex knowledge about things and their relationships
  - Enables rich knowledge representation on the web (based on SHOIN)
  - "OWL" easier to pronounce than "WOL"
  - 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



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

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

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
    - 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, e.g.,
    - `Q42` $\rightarrow$ Douglas Adams
  - Property: describes a relationship or attribute, e.g.,
    - `P31` (instance of), `P27` (country of citizenship)
  - Value: specific data linked to an item via a property, e.g.,
    - `Q42` (Douglas Adams) $\rightarrow$ `P31` (instance of) $\rightarrow$ `Q5` (human)
    - `Q42` $\rightarrow$ `P106` (occupation) $\rightarrow$ `Q36180` (science fiction writer)
  - Reference: supports a claim by citing a source, e.g.,
    - Stating Douglas Adams's citizenship with a reference to a biography
  - Qualifier: adds context or additional information to a statement
    - `Q90` (Paris) $\rightarrow$ `P1082` (population) $\rightarrow$ "2,165,423"
    - With qualifier: `P585` (point in time) $\rightarrow$ "2021"
    - Meaning: "The population of Paris was 2,165,423 in the year 2021"
- **Applications:**
  - Knowledge graph
  - Semantic search
  - AI reasoning
  - Data enrichment
  - AI and machine learning training datasets

DBPedia

- **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 3.7M`
  - Enables semantic queries over Wikipedia data via SPARQL endpoints
- **Applications**
  - Semantic Web research
  - 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
  - Designed to model the semantic relationships between words
  - Groups words into sets of synonyms
  - Manually curated, ensuring high-quality semantic relations
  - Can be incomplete for domain-specific language
- Key Components:
  - Synsets: 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
- Structure:
  - Semantic network where nodes are synsets and edges are relations
  - Organized hierarchically, especially for nouns and verbs
- Applications:
  - Word sense disambiguation: choose the correct meaning of a word in context
  - Semantic similarity measures: how close two concepts are
  - 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
- Key Characteristics:
  - Designed to capture knowledge that people generally assume but often leave unstated
  - Focuses on making AI systems more human-like in their understanding
- 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`)
- Example Triple:
  - (`bicycle`, `UsedFor`, `transportation`)
- Applications:
  - Natural language understanding
  - Question answering and chatbots
  - Commonsense reasoning in AI
  - Semantic search and recommendation systems

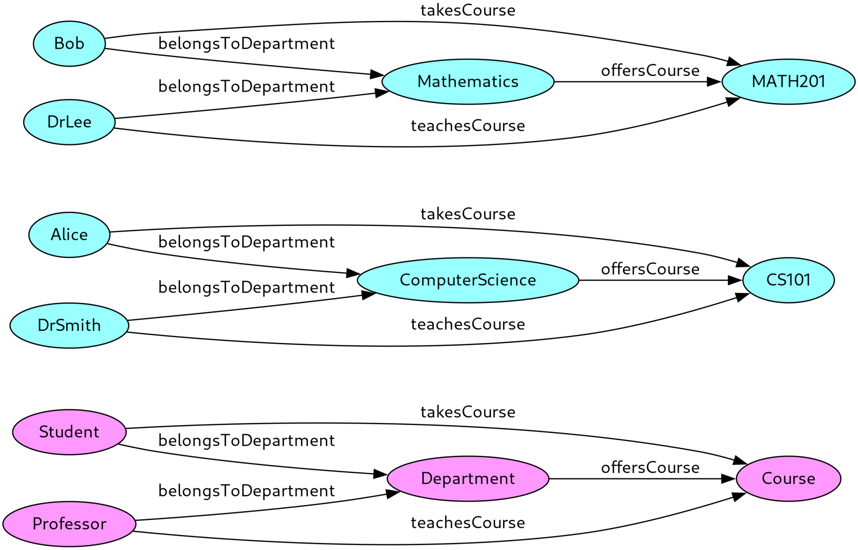
Frame-Based Representations

- Frame-based systems represent structured knowledge about objects, events, or situations
- Key Concepts:
  - Frame: A data structure for a concept or entity
    - E.g., a frame for `Dog` might include properties like `hasLegs`, `hasFur`, `barks`
  - Slots: attributes or relationships associated with the frame
    - E.g., slot `hasLegs` with value `4`
  - Slot fillers: values or links to other frames that fill the slots
    - E.g., slot `eats` might link to another frame `Meat`
- Example:
  - Frame: `Dog`
  - Slots:
    - `isA`: `Animal`
    - `hasLegs`: `4`
    - `sound`: `Bark`
    - `canDo`: [`Run`, `Fetch`]
- Features:
  - Inheritance: frames can inherit slots and slot values from more general frames, e.g.,

Knowledge Graphs (KGs)

- KGs represent entities and their relationships as a graph structure
  - Nodes = entities
  - Edges = relations
  - E.g., "Paris $\rightarrow$ isCapitalOf $\rightarrow$ 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



Technologies

- **TransE (Translation Embedding)**
  - Embedding model for knowledge graph completion
  - Represents relationships as translations in vector space: $h + r \approx t$
  - Good for 1-to-1 relations, less effective with complex patterns
- **RotatE**
  - Embeds entities in complex space
  - Models relations as rotations: $t = h \circ r$ where $\circ$ is complex multiplication
  - Captures symmetry, antisymmetry, inversion, and composition
- **DeepProbLog**
  - Combines ProbLog (probabilistic logic) with deep learning
  - Supports neural predicates in logic programs
  - Learns probabilistic facts and neural components jointly
- **PyMLN**
  - Python-based Markov Logic Network (MLN) system
  - MLNs combine first-order logic with probabilistic graphical models
  - Allows reasoning with weighted logical rules
- **ProbLog**
  - Probabilistic logic programming language
  - Extends Prolog by attaching probabilities to facts
  - Computes success probabilities of queries
- **Tuffy**