CHAPTER 10: KNOWLEDGE REPRESENTATION

Introduction

This chapter basically discusses what content should be fed into the knowledge base of an agent i.e. how to represent facts about the real world. This is done to ensure that the agent can make inferences based on this knowledge base to reason and solve real world problems.

The following are various representation languages or formalisms, but our focus would be on the first-order logic:

1. First-order logic

2. hierarchical task networks (for reasoning about plans)

3. Bayesian networks (for reasoning with uncertainty)

4. Markov models (for reasoning over time)

5. Deep neural networks (for reasoning about images, sounds, and other data)

10.1: Ontological Engineering

Ontology basically talks about organizing everything in the world into a hierarchy of categories. Ontological engineering encompasses creating real world representations by using some general abstract concepts such as events, time, beliefs, etc. that occur in many different domains.

For an example, we can say that animal can be seen as the general concept for every type of animal in existence, the details of what kind of animal can be filled in later.

Upper Ontology refers to the general framework of concepts and this is due to how it is represented in a graph format, i.e. with the general concepts at the top and the detailed concepts at the bottom.

An issue to consider when it comes to using first-order logic for representation is that it ignores possible exceptions that may arise. An example given is "tomatoes are red" this is true but not in all cases as some are green, yellow, or even orange.

There are two major characteristics of general-purpose ontologies that distinguishes them from collections of special-purpose ontologies:

1. General-purpose ontologies should be usable in basically every special-purpose domain, but for it to work perfectly with the aid of rules or details (domain-specific axioms), everything detail must be properly handled and not ignored.

2. In complex domains, different areas of knowledge can be unified and applied to get better knowledge (in terms of reasoning and problem solving) in that domain. For better understanding, an instance of a circuit-repair system robot was given and how it would need to apply knowledge of time, space and electrical systems and connectivity.

Note that it is easier to create an ontology (a limited-purpose ontology) when there are a limited number of involved parties because they can easily agree on a common motive or goal, than to create a general-purpose ontology.

Limited-purpose ontology, at least the ones that exist have been created through the following routes:

1. By a team of expert or trained ontologists or logicians. They design, plan and create the ontology and write axioms (assumed rule or principle that can't be proven or disproven), e.g. the CYC system.

2. By importing categories, attributes, and values from already existing database(s), e.g. DBPEDIA was built from structured facts from Wikipedia.

3. By parsing text documents and extracting information from them, e.g. TEXTRUNNER from Web pages.

4. By enticing unskilled amateurs to enter commonsense knowledge, e.g. OPENMIND system built from volunteers who proposed facts in English.

10.2: Categories and Objects

Categorization is key. We organize objects into categories to make sense of the world. While we interact with individual objects, we often think and reason in terms of categories. For example, when shopping, you might want to buy "a basketball" in general, not a specific one like "BB9."

Once we classify an object into a category, we can make predictions about it. For instance, if we see something with green and yellow skin, a round shape, and black seeds in the fruit aisle, we can guess it's a watermelon and assume it’s good for a fruit salad.

Categories can be represented in two ways in first-order logic:

1. As predicates (e.g., Basketball(b)).

2. As objects (e.g., b ∈ Basketballs), where we can say things like "Basketballs are a subset of Balls."

Categories can inherit properties from broader categories. For example, if all "Food" is edible, and "Fruit" is a type of "Food," then "Apples" (a type of fruit) are also edible. This is called inheritance.

Categories are organized into hierarchies (taxonomies). For example, species are classified into a huge hierarchy, and libraries use systems like the Dewey Decimal System to organize knowledge.

First-order logic helps us state facts about categories, like:

1. An object belongs to a category (e.g., BB9 is a basketball).

2. One category is a subset of another (e.g., basketballs are a type of ball).

3. All members of a category share certain properties (e.g., all basketballs are spherical).

4. Categories themselves can have properties (e.g., dogs are domesticated species).

Sometimes, categories can include other categories. For example, "Dogs" is a category, and it’s a member of "DomesticatedSpecies," which means "DomesticatedSpecies" is a category of categories. However, there are exceptions to rules (e.g., a punctured basketball isn’t perfectly spherical), which we handle later.

Beyond subclasses and members, we can define other relationships between categories:

1. Disjoint Categories: Two categories are disjoint if they share no members (e.g., "Animals" and "Vegetables" are disjoint).

2. Exhaustive Decomposition: A set of categories can fully cover another category. For example, "Americans," "Canadians," and "Mexicans" together make up all "North Americans."

A partition is an exhaustive decomposition where the categories are also disjoint (e.g., "Animals," "Plants," "Fungi," etc., partition "Living Things").

Categories can be defined by necessary and sufficient conditions. For example, a "bachelor" is defined as an unmarried adult male. However, strict definitions like this are easier for formal terms than for everyday objects, and definitions aren’t always necessary.

10.2.1: Physical composition

Objects can also be related through part-whole relationships. For example:

A nose is part of a head, Romania is part of Europe, and this chapter is part of this book.

The PartOf relation is transitive (if PartOf(x, y) and PartOf(y, z), then PartOf(x, z)) and reflexive (PartOf(x, x)). For example, since Bucharest is part of Romania, and Romania is part of Europe, we can conclude Bucharest is part of Europe.

Composite Objects: Some categories are defined by their structure. For example, a biped is an object with exactly two legs attached to a body. Representing "exactly two" in logic can be tricky, but tools like description logic (discussed later) make it easier.

Part Partition: Similar to how categories can be partitioned, objects can also be divided into parts. For example, the mass of a composite object is the sum of the masses of its parts. This is different from categories, which don’t have mass, even if their members do.

Bunch: When dealing with groups of objects (like a bag of apples), we use the concept of a bunch. A bunch is a composite object made up of its parts, not just a set. For example, BunchOf({Apple1, Apple2, Apple3}) represents the three apples as parts of a single object. Bunches are defined using the PartOf relation and are the smallest object containing all their parts.

Logical Minimization: Defining objects like bunches involves logical minimization, where an object is defined as the smallest one satisfying certain conditions. For example, BunchOf(s) is the smallest object that has all elements of s as parts.

10.2.2: Measurements

Objects have properties like height, mass, and cost, which are represented by measures. Measures are abstract values, such as lengths or weights, that can be expressed in different units (e.g., 1.5 inches = 3.81 centimeters). We can write facts like:

The diameter of a basketball is 9.5 inches.

The weight of a bunch of apples is 2 pounds.

The duration of a day is 24 hours.

Natural Kinds and Typical Instances: Many categories, like "tomatoes," don’t have strict definitions. Instead, they have typical features (e.g., red, round, juicy) but allow for exceptions (e.g., yellow or green tomatoes). To handle this, we distinguish between all instances of a category and typical instances. For example:

x ∈ Typical(Tomatoes) implies Red(x) and Round(x).

This approach allows us to reason about categories without needing exact definitions.

Philosophers like Wittgenstein and Quine have shown that many categories (e.g., "games" or "bachelors") can’t be strictly defined. Instead, they share family resemblances—overlapping features rather than strict rules. For example, the Pope might technically fit the definition of a bachelor but calling him one feels wrong because it ignores social context.

It is important to note that some properties, like beauty or difficulty, can’t be measured numerically but can still be compared. For example:

Exercise A is harder than Exercise B or that a tougher exercise is expected to result in a lower score.

This kind of reasoning, based on ordering rather than exact numbers, is the foundation of qualitative physics, which studies how to reason about systems without detailed equations.

10.2.3: Objects: Things and stuff

The world consists of things (countable objects, like aardvarks) and stuff (uncountable substances, like butter). The key difference is:

Things can be divided into distinct objects (e.g., cutting an aardvark in half doesn’t give you two aardvarks).

Stuff can be divided indefinitely, and any part of it is still the same substance (e.g., cutting butter in half gives you two smaller pieces of butter).

To represent stuff, we treat lumps of it as objects (e.g., Butter3 for a specific lump of butter) and define categories for the substance (e.g., Butter). Any part of a butter-object is still butter:

b ∈ Butter ∧ PartOf(p, b) ⇒ p ∈ Butter.

Stuff has intrinsic properties (e.g., melting point, color) that remain the same even when divided, and extrinsic properties (e.g., weight, shape) that change when divided. Categories based on intrinsic properties (e.g., Butter) are mass nouns, while those based on extrinsic properties (e.g., PoundOfButter) are count nouns.

10.3: Events

In simple worlds, we represent actions (e.g., shooting an arrow) and fluents (e.g., having an arrow) as propositions. We use successor-state axioms to describe how fluents change over time based on actions. For example:

If you shoot an arrow at time t, you no longer have it at time t+1.

However, this approach is limited to discrete, instantaneous actions.

Event Calculus: To handle more complex scenarios (e.g., continuous actions like filling a bathtub or simultaneous actions like brushing teeth while waiting), we use event calculus. Event calculus introduces:

Events: Actions or occurrences (e.g., E1 = Flying(Shankar, SF, DC)).

Fluents: Properties that can change over time (e.g., At(Shankar, Berkeley)).

Time Points: Specific moments in time.

Key predicates include:

T(f, t1, t2): Fluent f is true from time t1 to t2.

Happens(e, t1, t2): Event e occurs from time t1 to t2.

Initiates(e, f, t): Event e causes fluent f to become true at time t.

Terminates(e, f, t): Event e causes fluent f to become false at time t.

Example of Event Calculus: If Shankar flies from San Francisco to Washington, D.C., we can say:

Terminates(E1, At(Shankar, SF), t1) (he’s no longer in SF).

Initiates(E1, At(Shankar, DC), t2) (he arrives in DC).

Event calculus can also handle simultaneous events, continuous events (e.g., rising tide), and nondeterministic events (e.g., flipping a coin).

10.3.1: Time

Event calculus allows us to reason about time intervals, which can be either moments or extended intervals.

Moments: Intervals with zero duration (e.g., a specific instant in time).

Extended Intervals: Intervals with a duration (e.g., a day, a year).

We can define relationships between intervals, such as:

Meet: One interval ends exactly when another begins.

Before: One interval ends before another begins.

Overlap: One interval starts before another but ends during it.

During: One interval is entirely contained within another.

For example:

The reign of Elizabeth II immediately followed that of George VI (Meets(ReignOf(GeorgeVI), ReignOf(ElizabethII)).

Elvis’s reign overlapped with the 1950s (Overlap(Fifties, ReignOf(Elvis)).

Time Scale: We can define a time scale (e.g., seconds since January 1, 1900) and use functions like Begin, End, and Duration to describe intervals. For example:

Duration(AD2001) = Seconds(31536000) (the year 2001 lasted 31,536,000 seconds).

Time(Begin(AD2001)) = Date(0, 0, 0, 1, Jan, 2001) (midnight on January 1, 2001).

10.3.2: Fluents and objects

Physical objects can be viewed as generalized events that exist over time. For example, the United States (USA) can be thought of as an event that began in 1776 and continues today. Properties of objects, like the population of the USA, can change over time and are represented using state fluents (e.g., Population(USA)).

Changing Properties: Some properties, like the president of the USA, change over time. However, in logic, a term like President(USA) must refer to a single object. To handle this, we treat President(USA) as an object that consists of different people at different times. For example:

From 1789 to 1797, President(USA) refers to George Washington.

From 1797 to 1801, it refers to John Adams.

To say George Washington was president in 1790, we write:

T(Equals(President(USA), GeorgeWashington), Begin(AD1790), End(AD1790)).

Here, Equals is used to indicate that President(USA) and GeorgeWashington refer to the same object during that time period.

10.4 Mental Objects and Modal Logic

The text begins with a way to represent the mental objects in the human mind and the processes to manipulate them.

It kicks off with Propositional attitudes that an agent can have toward mental objects (eg; beliefs, knows, wants and informs) with the added difficulty is that these attitudes do not behave like normal predicates. For example, suppose we try to assert that Lois knows that Superman can fly:

Knows(Lois,CanFly(Superman)).

One minor issue with this is that we normally think of CanFly(Superman) as a sentence, but here it appears as a term. That issue can be patched up by reifying CanFly(Superman); making it a fluent. A more serious problem is that, if it is true that Superman is Clark Kent, then we must conclude that Lois knows that Clark can fly, which is wrong because (in most versions of the story) Lois does not know that Clark is Superman.

(Superman = Clark)∧Knows(Lois,CanFly(Superman))

|= Knows(Lois,CanFly(Clark))

This is a consequence of the fact that equality reasoning is built into logic. Normally that is a good thing; if our agent knows that 2+2 = 4 and 4 < 5, then we want our agent to know

that 2 + 2 < 5. This property is called referential transparency—it doesn’t matter what term a logic uses to refer to an object, what matters is the object that the term names. But for propositional attitudes like believes and knows, we would like to have referential opacity the terms used do matter, because not all agents know which terms are co-referential.

10.4.1 Other modal logics

Many modal logics have been proposed, for different modalities besides knowledge. One proposal is to add modal operators for possibility and necessity: it is possibly true that one of the authors of this book is sitting down right now, and it is necessarily true that 2+2 = 4.

In linear Linear temporal logic temporal logic, we add the following modal operators:

• X P: “P will be true in the next time step”

• F P: “P will eventually (Finally) be true in some future time step”

• G P: “P is always (Globally) true”

• P U Q: “P remains true until Q occurs”

10.5 Reasoning system for categories

10.5.1 Semantic networks

Semantic networks are knowledge models represented graphically, in which nodes represent objects, categories, or concepts and labeled arcs represent the relationship between them. They are an inherent way of visualizing and reasoning about hierarchical and relational knowledge. Eg; A node for "Mary" connected to a node for "FemalePersons" with a "MemberOf" link represents the logical assertion Mary∈FemalePersons*Mary*∈*FemalePersons*.

In semantic networks nodes represent objects, edges represent relationships but can also represent properties, special notation such as double boxed links asserts properties of all members of a catergory and the single boxed links asserts properties of individual objects.

Inheritance: Objects inherit properties from their categories. eg: If "Persons" have two legs, and "Mary" is of the category "Persons," then Mary inherits the property of having two legs.

Multiple Inheritance occurs when an entity is of more than one category (e.g., "Mary" is a "FemalePerson" and also a "Student"). This can cause conflicts if categories have opposing properties.

10.6 Reasoning with Default information

10.6.1 Circumscription and default logic

Default reasoning allows agents to draw conclusions that can be retracted when new evidence is found (e.g., "birds fly" unless refuted).

Nonmonotonic logics (e.g., circumscription and default logic) support such reasoning:

* Circumscription: Restricts the domain of certain predicates (e.g., "abnormal") to allow default assumptions.
* Default Logic: Uses default rules (e.g., "If Bird(x) and it's consistent, then assume Flies(x)") to draw temporary conclusions.

Problems:

* Default reasoning can create ambiguities (e.g., the "Nixon diamond" problem: Nixon is a Quaker and a Republican, and so defaults conflict).
* Prioritized schemes (e.g., prioritized circumscription) can remove some ambiguities by taking precedence for some defaults.

10.6.1 Truth Maintenance Systems

Truth maintenance systems handle knowledge updates and revisions efficiently.

Belief revision is eliminating beliefs that are no longer tenable due to new evidence.

Truth Maintenance Systems (TMS) keep justifications for the beliefs in their attempt to maintain retractions and revisions in a cost-effective manner:

* Justification-Based TMS (JTMS): The justification for each sentence is labeled. Deleting a sentence removes only those sentences that depend solely on it.
* Assumption-Based TMS (ATMS): Keeps many hypothetical states simultaneously by labeling sentences with sets of assumptions.

Applications:

* Effectively switching between hypothetical scenarios (e.g., imagining different Olympic event venues).
* Constructing explanations by determining minimal sets of assumptions leading to a conclusion.