Knowledge Representation

Yashovardhan Srivastava

7 June,2021

Introduction

This chapter builds upon the on First Order Logic.We described a the syntax,semantics,ontology,epistemology of First Order and other logic. In this chapter we try to answer how do we represent facts about the world to our knowledge base.We will introduce new concepts to include more general and flexible representations of the world.

Ontological Engineering

Ontology as described previously refers to the nature of reality. In small domains,ontology does not matter that much,real world on the other hand requires general and flexible definitions or representations to encompass the true reality. Representing these general and abstract concepts is sometimes called **ontological engineering**. The main question is how do we do that ?

Representing everything in the world is a humongous task,but representing an outline of an object where new knowledge can fit in is possible and is similar to the way designers of object oriented programming define general concepts in classes(which later could be modified using inheritance).The details of different types of objects could be fit later. The general framework of these concepts is called **upper ontology**.Upper Ontology makes many simplifying assumptions by generalizing concepts.A more general ontology is better suited for the world because it would allow for simultaneous changes extended over time.General purpose ontologies posses the following features:

- It should be applicable in more or less any special purpose domain.
- In any sufficiently demanding domain,different areas on knowledge must be unified.

It should be noted that none of the top AI applications make use of general shared ontological engineering(or had only limited success).

Categories and Objects

Organization of objects into categories is one of the most important step of knowledge representation.Categories help us make predictions about objects once they are classified Categories also serve to organize and simplify the knowledge through **inheritance**. Inheritance helps us create a hierarchical model of categories where more specific categories could be derived from general categories.

First Order Logic makes it easy to state facts about categories either by relating objects to categories or by quantifying over their members.For example:

- An object is a member of a category
  $FB_{10} \in Footballs$
- An category is a subclass of another category
  $Football \in Balls$
- All members of a category have some property.
  $(x \in Footballs) \Rightarrow Spherical(x)$

- All members of a category have some property.
  $(x \in Footballs) \Rightarrow Spherical(x)$
- Members of a category can be recognized by some properties.
  $White(x) \wedge Round(x) \wedge (x \in Balls) \Rightarrow x \in Football$
- A category as a whole has some properties.
  $Humans \in Homosapiens$

Although subclass and member relations are the most important ones for categories,we also want to be able to state relations between categories that are not sub-classes of each other. Some of them are:

Exhaustive decomposition

From our previous categorization, we cannot deduce relationship between categories that are at the same level in hierarchy.For example: if we just say *Males* and *Females* are subcategories of *Animals*,we have not said that a male cannot be a female.These type of disjoint relations are said to have formed an exhaustive decomposition if a higher class ought to have atleast one of the subcategories. In our example, we will know that animal is not male if it is a female and vice- versa. A disjoint exhaustive decomposition is known as partition.

Physical Composition

Physical composition refers to the idea that one object can be part of another. We can use the general *PartOf* relation to say that one object is part of another.It is similar to the *Subset* category.The *PartOf* relation is transitive and reflexive.

Measurements

Measures refer to the values we assign to the quantitative measures of the world. Some measures such as length,price are easy to represent.Other qualitative aspects present more problem because they have no agreed scale of value.Examples include : difficulty,beauty . Assigning a scale to these aspects is unnecessary because the most important aspect of measures is not that it has particular values but that measures can be ordered and compared .

Objects: Things and Stuff

These refer to the objects that seems to defy any obvious individualization,i.e there is no obvious number of objects that make it because any part of it is still itself only. For example: Suppose you have a glass of water and biscuits in front of you. You can say that you have n number of biscuits , but there is no obvious number of "water-objects" as any part of water is still water. We give these portion a generic name **stuff**.Biscuits on the other hand comes under **things**,objects that can be individualized. This distinction between *stuff* and *things* in English Language is same as between mass nouns and count nouns. How do we come up with ontologies to distinguish between these kind of objects ?.
What actually is going on is that some properties are **intrinsic :** they belong to the very substance of the object,rather than to the object as a whole. When an instance of *stuff* when cut in half retain the intrinsic properties - things like density,boiling point and so on. On the other hand **extrinsic** properties such as - weight,length,shape and so on are not retained under subdivision.

Events and Event Calculus

Previously,we saw how Situation Calculus represents actions and their belief as a set of first order logic formula. One drawback of Situation Calculus is its limited ability to represent simultaneous events. It was designed to represent a world in which actions are discrete,instantaneous and non-simultaneous. Although they are quite useful in areas where these conditions are required,they fail to capture the simultaneous and continuous world. To handle such cases , an alternate formalism was proposed called **event calculus** ,which was based on points of time rather than on situations.

Events **reifies** fluents and objects. Instead of saying whether a situation happens or not , we assert whether they are true or not at a given time. For example: The fluent $Has(Child, Ball)$ refers to the fact that the *Child* has the *Ball* or not ,whereas on the other hand to assert that the *Child* has *Ball* at time t ,we use $T(Has(Child, Ball), t)$. Let us see some of the predicates for a version of event calculus.

- $T(f, t)$: Fluent f is true at time t.
- $Happens(e, i)$: Event e happens over time interval i.
- $Initiates(e, f, t)$: Event e causes fluent f to start to hold at time t.
- $Terminates(e, f, t)$: Event e causes fluent f to cease to hold at time t.
- $Clipped(f, i)$: Fluent f ceases to be true at some point in interval i.
- $Restored(f, i)$: Fluent f becomes true sometime during the interval i.

For our convenience, we can extend the T to include intervals of time as well as time points. a fluent hold over an interval if it holds on every point within the interval.

$$T(f, (t_1, t_2)) \iff [\forall t \ (t_1 < t < t_2) \Rightarrow T(f, t)]$$

With this extension, it is possible to represent complicated events such as exogenous(events whose influence is outside of the model specified),continuous and simultaneous events.

Processes

Processes are sub-classes of events with the property that the process that happen over an interval also happens during any sub-interval.These are also called **liquid-events** categories.

$$(e \in Processes) \wedge Happens(e, (t_1, t_4)) \wedge (t_1 < t_2 < t_3 < t_4) \Rightarrow Happens(e, (t2, t3))$$

The distinction between liquid and non-liquid events is analogous to the difference between *stuff* and *things*.

Time Intervals

In event calculus, since we have the possibility to talk about time and time intervals,we will consider two types time intervals namely: **moments** and **extended intervals** with the distinction that moments have zero duration. We can associate a time scale and associate points on that scale with moments , giving us absolute times. For example : we can arbitrarily say that Midnight(GMT) January 1 1970 (the UNIX time stamp) has time 0 and it increments by each second continuously.The *Begin* and *End* pick out the earliest and latest moments in a interval.

$$Time(Begin(AD1970)) = Seconds(0)$$

$$Time(Begin(AD2020)) = Seconds(1577836800)$$

$$Time(Begin(AD2021)) = Seconds(16209459200)$$

To make these numbers easier to read,we can make use of a function *Date*,which return a time point given seconds since the epoch.

$$Time(Begin(AD1970)) = Date(0, 0, 0, 1, Jan, 1970)$$

$$Date(5, 20, 0, 5, 7, 1979) = Seconds(300000000)$$

Fluents and Objects

Physical objects can be viewed as generalized events in the sense that they are a chunk of space time. For example in a dynasty let $King(Dynasty)$ denote the king of the dynasty. In a continuous world,one might propose that $King(Dynasty)$ is a logical term and denotes different object at different time,but unfortunately a term cannot denotes only one object in a given model structure[1] To say Napoleon was the King in the year 1800 , we say:

$$T(Equals(King(Dynasty), Napoleon), 1800)$$

Notice here that we have used *Equals* rather than logical predicate =, because

- We cannot have predicate as an argument to T.
- Our interpretation is not that $Napoleon$ and $King(Dynasty)$ are logically equivalent to 1800. Logical identities are not something that change over time.

Mental Events and Mental Objects

The theory we have developed so far is able to deduce facts and beliefs but it does not have any knowledge *about* facts and *about* beliefs. Knowledge and reasoning about one's own self and and other agents is useful for controlling inference. In order to tackle this issue,we needed a model of mental objects that are inside someone's head and mental processes that act on those mental objects. One way to do this is to start by defining propositional attitudes that it can have towards mental objects.These include attitudes such as *Believes*, *Knows*, *Wants*, *Intends*, *Informs*. There is however a problem : For some propositional attitudes we want **referential transperancy** ,i.e it doesn't matter what term a logic uses to refer to an object,what matters is the objecct that the term names and for others we want **referential opacity** the terms used do matter. Since this is something that first order logic cannot handle , we need a new approach.

Modal Logic

Modal Logic is designed to tackle above mentioned problem. Modal logic includes special modal operators that take sentences as arguments(rather than terms).It takes two arguments,an agent and a sentence. Modal Logic allows us to have a model in which consists of collection of **possible worlds** , different from first order logic which contains one true world. It allows us to have the logic of necessity and possibility instead of absolute truth , which makes it possible to to form complicated logics.

We can define complicated logic through modal logic as the notion of possibility more closely represent our thinking rather than absolute truth. One problem, however in modal logic approach is that it assumes **logical omniscience**- the idea that if an agent knows a set of axioms , then it also knows all the consequences of those axioms. This idea of "unlimited rationality" is on shaky ground even for somewhat abstract notion of knowledge and worse for belief .The notion of belief is not something that is derivable, it is something that is physically built into the agent. Some attempts have been made to limit the rationality of agents ,but they have met with little success.

Syntax and Semantics

In our discussion above,we defined that Modal operators used sentences instead of terms. For example: "A knows P" in modal logic represented with the notation $\mathbf{K}_A P$ which means that agent **A** has knowledge about **P**. The set of all possible worlds(universe) are connected in a graph through **accessibility** relations which controls which worlds can see each other for the sake of determining what is true. For example : A world w_0 is accessible from w_1 with respect to a modal operator $\mathbf{K}_A P$ if everything in w_1 is consistent with what **A** knows in w_0 .This is written as $Acc(\mathbf{K}_A P, w_0, w_1)$. In general , a knowledge atom $\mathbf{K}_A P$ it true in world w **iff P** is true in every world accessible from w.

[1]This might be to avoid contradictions when checking consistency of a term from KB.Ontologies keeps time indicies separate from fluents.

Reasoning Systems for Categories

This system describes system specially designed for organizing and reasoning with categories. Two systems that will be useful for such tasks include **semantic network**: providing graphical aids for visualizing category hierarchy and **description logics** for providing a formal language for combining category definitions and deciding subset/super set relation between categories

Semantic Networks

A semantic network represents semantic relationships between individual objects,categories of objects and relations among objects of the knowledge base. A typical graphical notation includes object or category names in boxes connected via labeled relation links. This notation makes it convenient to perform inheritance reasoning as we saw earlier . One important to be noted is that there should be distinction between members of a category and properties of the category as a whole. For example : We know that persons have female persons as their mother. so can we draw *HasMother* link between *Persons* and *FemalePersons* ? The answer is no because *HasMother* is a relationship between categories and categories do not have mothers,there members do. To highlight such relation, the author has introduced a new notation - the double-boxed link which asserts that(for this example):

$$\forall x \in Persons \Rightarrow [\forall y \ HasMother(x, y) \Rightarrow y \in FemalePersons]$$

Category links between categories are mostly inheritance relations. The predicate *MemberOf* can be used to link categories that form inheritance relations and the *SubsetOf* to denote subset or superset relation. These are quite useful for drawing inferences from relations and is one of the main attractions semantic networks.

Another important aspect of semantic networks is the ability to represent default values of categories . For some categories default status is naturally enforced on objects unless contradicted by more specific information. For example: Assume that John has one leg , despite the fact that John is a person and person have two legs. In a logical KB this would be a contradiction but in semantic network the assertion that all person have two legs is a default status ,that may be **overridden** by more specific values.

One drawback that is often highlighted for semantic network notation is that it can represent only binary relations whereas first order logic can form general *n-ary* relations. One way to work around this flaw is to **reifying** the proposition for appropriate event category. Reification of propositions makes it possible to represent every ground,function free atomic sentences of first order logic in semantic networks. One disadvantage of reifying propositions is that negates one of the main advantages of semantic networks and that is simplicity and transparency of the inference process.

Descriptive Logic

Description logics are notations that are designed to make it easier to describe definitions and properties of categories. The principle inference tasks from description logics are **subsumption** - checking if one category is subset of another by comparing their definitions and **classification** - checking whether an object belongs to a category or not. Some systems also include **consistency** of a category definition - whether the membership criteria are logically satisfiable. The CLASSIC language (Borgida *et al.*,1989) is an example of typical description logic language. In CLASSIC, to say that bachelors are unmarried adult males , we would write:

$$Bachelor = And(Unmarried, Adult, Male)$$

The equvivalent in first-order logic is:

$$Bachelor(x) \iff Unmarried(x) \wedge Adult(x) \wedge Male(x)$$

Notice that description logic has algebra of operation on predicates,which we cant do in first-order logic. Any description in CLASSIC can be translated into an equivalent first-order logic sentence,but some descriptions are more straightforward in description logic.

The most important aspect of description logics is their emphasis on tractability of inference. A problem instance is solved by describing it and then asking if it is subsumed in one of the possible

solution categories. In first-order logic,predicting solution time is often impossible, it is often left to the user to engineer the representation around sets of sentences. Description logic languages such as CLASSIC provides efficient subsumption testing techniques, but the worst case run time is exponential. This sounds good in principle until one realizes that only one of the two things are possible : either hard problems could not be stated at all,or they require exponentially large descriptions. Other feature that Description logics lack are *negation* and *disjunction* , which makes even the simpler sentences complex.

Reasoning with Default Information

In previous sections we defined assertions with default status which can be overridden by more specific information using the inheritance mechanism in a simple and natural way. In this section, we will discuss defaults more generally with a view towards understanding the semantics of defaults.

Circumsciption and default logic

In commonsense reasoning failures of **monotonic reasoning**[2] are evident. It often seems that humans often 'jump to conclusions'. For example when a car is moving the conclusion that the driver is present is assumed automatically, regardless of the fact that the car might be a self-driving car. There is no denying the fact that in a moving car the probability that there is a driver is high,but the possibility that the car is self-driving *does not arise unless some new evidence presents itself.* The driver conclusion is reached *by default*, in absence of any reason to doubt it. If a new evidence arrives such as (in the driver example) looking closely at the driver seat - then the conclusion can be retracted[3] This kind of reasoning is said to exhibit **nonmonotonicty**. **Nonmonotonic logics** have been devised with modified notions of truth and entailment in order to capture such behavior. Two of such logics which have been studied extensively are circumscription and default logic.

Circumscription

Circumscription can be viewed as a more powerful and precise version of closed-world assumption. The idea is to specify particular predicates as false as possible except for those which are known to be true. For example - suppose we want to assert the default rule that All Dogs are cute. We will introduce a predicate, say $Exception_1(x)$ and write that:

$$Dogs(x) \wedge \neg Exception_1(x) \Rightarrow Cute(x)$$

If we say that $Exception_1$ is to be circumscribed, a circumscribed reasoner is entitled to assume $\neg Exception_1(x)$ unless $Exception_1(x)$ is known to be true. This allows the conclusion $Cute(Bruno)$ drawn from the premise $Dog(Bruno)$, but the conclusion no longer holds if $Exception_1(Bruno)$ is asserted,which states that $Bruno$ is not $Cute$ (which I highly doubt).

Circumscription can be viewed as an example of a **modal preference** logic. In such logics, a sentence is entailed(with default status) if it is true in all *preferred* models of the KB, as opposed to the requirement of truth in *all* models of classical logic. For circumscription, one model is preferred over the another if it has fewer abnormal objects. Other forms of circumscription include **prioritized circumscription** which gives preference to one model over the other based on some belief.

Default Logic

Default logic is a formalism in which default rules can be written to generate contingent,nonmonotonic conclusions. A default rule looks like this:

$$Dog(x) : Cute(x)/Cute(x)$$

This rule means that if $Dog(x)$ is true, and it $Cute(x)$ is consistent with the knowledge base, then $Cute(x)$ could be concluded by default. In general, a default rule could be written as:

[2]Reasoning in which the conclusion remains same even when new knowledge is added to our KB

[3]This is similar to how a belief updating works.We have a general assumption and when new information is received,we update our existing KB.

$$P : J_1, J_2, ...J_n/C$$

where P is the prerequisite,C is the conclusion and J_i are justifications - if any of them can be proven false, the conclusion cannot be drawn. Any variable that appears in J_i or C must also appear in P.

To interpret what default rules mean, we can extend the default theory to be a maximal set of conclusions from the default value, *i.e* an extension S consists of original known facts and a set of conclusions that can be drawn from the default rules such that no additional conclusion can be drawn from S and the justifications are consistent with S.

Since when non-monotonic logics were first proposed,they have been a subject of interest among researchers. A great deal of progress has been made to understand their mathematical properties. There are still many unanswered questions such as: 'What is a good set of default values to have ?' and 'How can beliefs that have default status be used to make decisions ?' and 'How do we measure *trade-offs* in decisions ?' amongst many more.

Truth maintenance systems

We hhave seen that for many inferences drawn from the knowledge representation system are not absolutely certain. This might be because they were represented as events which might be true or false depending on time or that the system will have only default status,not a certain truth value. Inevitably, some of the facts have to be revised and new information has to be updated in the knowledge base. This process is called **belief revision**. Removing assertions that are wrong , is however a difficult procedure. Suppose that the Knowledge base contains a sentence P that is to be removed. The problem arises if any additional sentences were inferred from P and asserted in the knowledge base. Retracting all sentences inferred from P fails because such sentences may have other justifications besides P. **Truth maintenance systems** or **TMS** are designed to handle exactly these kinds of complications. One simple approach to truth maintenance is to keep track of order in which sentences are told to the knowledge base . The problem with this approach is that inferences are drawn again for each sentence P_i which is impractical for many large-scale applications.

Justification based Truth Maintenance Systems(JTMS)

A much more efficient approach is to use justification based truth maintenance system, or **JTMS**. In a JTMS,each sentence in the KB is annotated with a justification consisting of a set of sentences from which it was inferred. justifications make retractions efficient - the time required for retraction of P depends only on the number of sentences derived from P rather than number of sentences added since P entered the knowledge base.

The JTMS assumes that sentences that are considered once will probably be considered again, so rather than deleting the sentences from the KB directly,it merely marks the sentences as being *out* of the knowledge base. If some other assertion restores a sentence's justification, it marks the sentence as being back *in*. This is particularly helpful in event - based systems where an assertion that is true during one instance might not be true during some other instance.

Apart from handling the retraction of incorrect information, TMS are very helpful in speeding up the analysis of multiple hypothetical scenarios . While in many other logical systems a great deal of work must be done to work out logical consequences of actions, and any mistake in one term might propagate throughout the system , in TMS instead of starting again from scratch,we just need to assert new terms and TMS takes care of necessary revisions. This is one of the major advantages of using TMS.

Assumption based Truth Maintenance Systems(ATMS)

Another TMS that makes **context-switching** as described above efficiently is assumption-based truth maintenance system or **ATMS**. Unlike JTMS where maintenance of justifications by retracting only one state at a time, ATMS represents all the states that have been considered at the same time. Each sentence in ATMS has a label that consists of set of assumption sets. The sentence holds just in those

cases in which all the assumptions in one of the assumptions sets hold.

TMS also provide a mechanism for generating **explanations**[4]. This is due the fact that explanation can also include **assumptions** and an ATMS can generate a set of explanations(the assumption set) for a given assertion. In most cases we will prefer an explanation E that is minimal *i.e* there is no proper subset of E that is also an explanation.

References

[1] Russel,Stuart and Norvig,Peter.*Artificial Intelligence-A Modern Approach*(3rd ed.).Pearson Education,2010

[4] An explanation of a sentence P is a set of sentences E such that E entails P