AI Module 24
Propositional and Predicate logic

Introduction

From this module, we will start embarking on methods to represent knowledge. We have seen algorithms for searching, heuristics based methods, planning, game playing etc. Each of these things needs some way to represent knowledge forstoring as well as processing. In fact we have discussed a few ways of representing knowledge in our initial modules when we wrote rules to represent problems and their solutions. We are going to learn them formally now. We will begin with Formal Logic and then we will see other ways of representing knowledge using formal logic.

The module discusses formal logic on which both propositional and predicate logic is based. We will see how simple statements are represented using propositional and predicate logic and how one can infer using that representation.

Formal Logic

This module begins with one of the simplest methods of knowledge representation, propositional logic. We will also see a better and more practical method using predicate logic in this module. These methods are known as formal logic methods as they help represent the statements in a structured and formal way. For an agent to take decisions, the problem statement and rules must be defined in some standard way so the agent can reason in a systematic way. Usually some symbolic representation is chosen. We have, in previous modules, described problem statements and rules using some methods without really describing them in detail. Now we will be describing the structure formally.

In 1976, Newell and Simon gave *physical symbol system hypothesis*. According to them one needs a physical symbol system to act in an intelligent way. The physical symbol system comprises of symbol structures or expressions (for example On(A, B), OnTable(B) etc.). There are many arguments made in favour and against the usefulness of the hypothesis but it is clearly visible that such a system is really needed for taking intelligent action. For an intelligent action, such symbol structures are constructed which represent real world entities, rules, situations etc. Sometimes these symbol structures presents additional non-real-word entities needed to solve problems at hand. For example there is nothing called ‘furniture’ in strict realistic sense. We have chairs, sofas, tables and so on which we call furniture collectively. Similarly we have a concept called shape which represent a collection of circle, square, and so on. Such abstractions are sometimes needed for programming convenience and sometimes needed to handle a situation where any arbitrary element of multiple type (for example a chair or a table or a sofa) to be represented as an abstracted single type of entity.

The study offormal Logic helps two things. Representing knowledge and infer from it. For any problem to be solved, all related information is to be stored in some conceivable form and we also have to have some method for inferring from it. We have already learned a few examples of representation of knowledge and also using rules etc. to reason from them.

Formal logic begins with some facts known to be true and continuously incrementing number of true facts based on old known facts and inferring from the collection of facts. Old known-to-be-true facts are known as *premises* and the methods used to infer are known as *arguments*. The resultant additional true facts are called *conclusions*.

For example we have following premises

- Vijay is a teacher
- All teachers are educated

And we apply argument to conclude that

Vijay is educated.

We can get many similar examples to illustrate same argument. For example we might also conclude using argument for a very different set of premises

- Radha is a housewife.
- All housewives like to cook.

And we apply argument to conclude that

Radha likes to cook.

The arguments discussed above are based on inference rule known as *syllogism*; i.e. x → y and y→z than x→z. Not always such argument is valid. Let us take another example to understand. We have two premises yet again.

- Sachin is a sportsman.
- Sportsmen are found all over the world.

And we can apply the same argument to conclude that

Sachin is found all over the world[1].

Thus, even when given premises are correct and the argument applied in one case is correct, it might fail to get correct conclusion otherwise. Thus we must need a formal structure to validate this process.

Entailment in Formal Logic

The discussion that we had in the previous session clearly indicates that we need a mechanism to represent formal logic. Before we proceed, let us answer a query that might come to our mind. How do we get premises? Human get information through all five senses and learn things. The premises that we have presented are learned through one of the five senses. Logic enhances the collection of true facts by concluding further[2].

[1] Why do you think this inference is incorrect? The reason is that the attributes of elements of the class (or objects) are inherited and not of the whole class. The attributes of the class for example number of elements, the structural rules for membership and allocation and revocation of members etc. are attributes of the class as an entity and are not inherited by their elements. The class sportsmen has the attribute "distributed all over the world" but it is not the attribute inherited by the members. On the contrary, Game is an attribute (the game the sportsman plays) inherited by all members. We need special attention to handle such problems. Frames, a typical method of representing knowledge, quite similar to class definitions of languages like C++ and Java, provides the mechanism for such cases with static variables which only exists for a class and are as many as the instances of the class irrespective of number of objects.

[2] These are two different ways to learn, one from outside, using senses, and another from within, concluding, conceiving, thinking, processing the facts and so on. Richard Felder and Linda Silverman proposed a well-known model of learning styles which represents the learners preferring to learn from surrounding as Sensing Learners while those who prefer to do from within are Intuitive Learners. In fact these set of attributes of their model is based on some other model previously described by other researchers.

Another interesting question that might come to your mind may be

Given a set of facts known to be true (given the complete set of premises), can we get all facts which are true based on those premise?

The set of all facts proved to be true is sometimes known as *entailment* of the premise. Now our question is, can we get the entailment if we have the premises?

Little thinking can bring to our notice that it is possible if we decide how can we infer and try inferring from all the premises one after another. For example the syllogism inference can be done for all cases where we find some x→ y and y→ z. Interestingly, the new facts also belong to the set of true facts and we can continue applying our inference rules over them to produce yet another set of true facts.

The discussion above clearly indicates that we need a structured and elegant method to represent premises and conclude using valid arguments. One of the simplest methods to do so is known as propositional logic which we will describe in the subsequent sections. Before we begin, let us introduce the popular symbols used in propositional logic as well as predicate logic which we discuss later.

∀	Universal quantifier (for all)
∃	Existential quantifier (there exists)
→	Implication
∧	And
∨	Or
¬	Not

The universal quantifier indicates that the statement is true for all elements of the domain while existential quantifier means the statement is true for some of the elements. Both of these symbols are used with predicate logic and rest are used with both; propositional as well as predicate logic. Implication means when one statement is true, the other also is true. The later three, ∧(And), ∨(Or) and ¬(Not) have conventional meaning. We will see a few examples of the usage of these symbols in representing simple English statements.

Proportional logic

In propositional logic, a real world situation is represented using a proposition. Here are examples of real world statements.

- Sachin is a cricketer
- Saina is a badminton player
- Saniya is a tennis player
- Jay is a badminton player or a tennis player.
- If Sachin is a cricketer, he is a sportsperson

We can represent them as follows using propositional logic

- SachinCricketer
- SainaBadminton
- SaniyaTennis
- JayBadminton V JayTennis
- SachinCricketer→SachinSportsPerson

Some authors prefer to use single letter symbols like P, Q, R etc. to describe propositions rather than using more explicit SachinCricketer or SainaBadminton. That method helps representing complex cases. For example if we have following statements.

P = Jay is a cricketer

Q = Jay is a badminton Player

R = Jay is a sportsman

S = Jay lives in Ahmedabad

If we want to state that "Jay lives in Ahmedabad and he is either a cricketer or a badminton player" we can represent that as

S ∧ (P V Q)

This representation is quite appealing if one wants to prove simple facts. For example we can have following facts known to us

1. P
2. Q
3. P V Q

Now if we know that 1 is true, we can easily conclude that 3 is true.

Let us take another example

1. P
2. Q
3. P → Q

Now we know that P is true, we can conclude that Q is true.

Need for Predicate logic

Can you see the problem with this representation? If we have one more statement, "all cricketers are rich" in our kitty, can we prove Sachin to be rich? It is hard unless we will try a little different way of representing the statements. For example, we can represent them using *first order predicate logic*. This is the first example of this chapter. We call it Example 24.1. We subsequently increment the number.

Ex. 24.1

1. Cricketer (Sachin)
2. Badminton (Saina)
3. Tennis (Saniya)
4. Badminton (Jay) V Tennis (Jay)

The universal quantifiers are handy when we want to have statements like "All cricketers are rich" and use them for implication as follows.

5. $\forall x\ Cricketer\ (X)\ \rightarrow Rich(X)$

In the above statement X is known as a variable which can assume values like Jay and Sachin. Now we can combine 5 and 1 with X = Sachin and can prove Rich (Sachin). This is the power of first order predicate logic. The statements Cricketer (Sachin) are known as *predicates*. The word Cricketer is a

name of that predicate and Sachin is the *argument* of that predicate. A predicate can have multiple argument as in following cases.

Ex. 24.2

1. Mama(Bajrangi, Shahida)
2. Brother(Shan, Sagarika)
3. Father (Rajiv, Rahul)
4. Relation (Gujarat, Capital, Gandhinagar)
5. Relation (Gujarat, LargestCity, Ahmedabad)

Last two predicates can also be written in a little different form as

1. Relation (Gujarat (Capital, Gandhinagar))
2. Relation (Gujarat (LargestCity, Ahmedabad))

In that case, it becomes second order predicate logic. We are not going to explore second order predicate logic further.

Predicate logic consists of two things, facts and rules. In above Ex. 24.1, 1 to 4 are facts and 5 is an example of a rule.

Predicate Structure

The predicate, as you could see, begins with the **predicatename**. Father, Mamaetc are names of the predicate. The information within the closed parenthesis (), are known as **arguments** of the predicate. Number of arguments required by the predicate is called **arity** of a predicate. The order in which the arguments are passed is decided by the designer. For example, when we write Mama (Kans, Krishna), we assumed first argument being maternal uncle while the other argument being nephew. Somebody else may prefer to write it as Mama(Krishna, Kans) with first argument being nephew and second being the maternal uncle. That means it is the designer who decides the meaning and order of each predicate. That obviously concludes one more thing. If the designer does not remain consistent with his representation, quite unpredictable consequences occurs when somebody tries to prove something. For example in the same database if we insert the predicate Mama (Shakuni, Duryodhan), there is a likelihood of a problem. Our earlier representation uses a relation that X is Y's mama if we write Mama(Y, X). Unfortunately while entering the 2nd predicate, the order is not preserved and one might come to conclusion that Duryodhan is Shakuni's and that is why Gandhari's mama rather than the son! If we have a fact like son (Duryodhan, Gandhari) in the database, we can as well prove that Grandpa (Duryodhan, Duryodhan)! That means, Duryodhan is his own Grandpa!

An excellent tool for writing and testing predicates and rules and how one can conclude using them is a language called Prolog. We will not discuss Prolog further as it is a discipline in itself.

Another important point is that the predicate itself is not true or false. It is the argument which decides so. For example it is possible to have four statements like following.

Player (Anand)

¬Player (JagjitSingh)

¬ Singer (Anand)

Singer (JagjitSingh)

All four statements can be true at the same point of time. Thus player or singer itself is not either true or false but the value that we pass makes it so.

Another important point. When we use a variable, the truthfulness of the statement depends on the binding of the variable. For example if we have following.

Player(X)

Can we say anything about the truthfulness of the statement? Not unless we bind X to a value. If X is bound to Anand the statement is true but if it is bound to JagjitSingh, it is false.

Using Universal and Existential quantifiers

Another important thing is about using variables in different context. If we use a predicate which is true for all of the members of the domain, we can use universal quantifier while if it is only true for some, we can use an existential quantifier.

For example, consider following statements

1. All of the players are physically fit
2. Some of the players are singers

How can we represent them? You can see that representing 1 requires universal while representing 2requires existential quantifier.

1. $\forall x$ Player(X) →PhysicallyFit(X)

2. $\exists$x Player(X) → Singer(X)

In case of 1, the statement is true for any value of X, thus we can replace any value with X and the statement is true. In case of 2, it is not so.

Representing facts and rules

Let us take one more example to understand the use of predicates in representation of facts and rules.

Ex. 24.3

1. Anand is a player.
2. Anand plays chess.
3. Anand was a world champion.
4. Anand has beaten Gary.
5. If somebody has won the match, he has played that match with the opponent.

Let us represent these facts using predicate logic[3]

1. Player (Anand)
2. Plays (Anand,Chess)
3. WorldChampion (Anand, Chess)
4. Beat (Anand, Gary)
5. $\forall x \forall y$Win (X,Y) →PlayedMatch (X,Y)

[3] There are many possible representations. We are picking up one of them. Using one representation does not invalidate another. In fact some representations work better in typical cases. This is in line with the discussion that we had about representing

Let us start with first statement. The representation Player (Anand) seems quiet straight forward. Even in this simple case, we are not representing the statement as is. We are omitting the present tense indicator. If we have some other fact, Ramakrishnan was a player, we will write Player (Ramakrishnan). The difference between two statements, Anand is player NOW and Ramakrishnan was in PAST is ignored in our representation. The second predicate relates Anand and the game he plays. The third one also initiate a debate. Should we have a representation like WorldChampion as a single predicate name? Should not it be Champion (World, Anand, Chess)? One more representation may be Champion (level (World), name (Anand), game (Chess)) which is basically a second order predicate. Each detailed representation improves clarity and enable better inference but complicates things as well. The advantage of complex representation is to provide finer relationships and prove things not possible otherwise. We have preferred simplicity over ability to do complex reasoning. Fourth predicate is similar to earlier ones and thus does not need to be elaborated. Fifth predicate, a rule, requires some explanation. First, playedMatch is again a predicate with a rude representation of two things into one. The other issue is use of universal quantifier. The meaning of $\forall x \forall y$ is that the statement is true for any X and any Y belongs to some set. It says if somebody has won against somebody else, they have played a match. (Here we are ignoring the possibility of a walk over).

Interestingly, if we provide a human reader information described above, he can easily conclude that Anand and Gary played a game. Can we prove that given our predicate representation?

We will try a method known as backward chaining. We will have to prove

PlayedMatch (Anand,Gary)

We will try proving it using the available information but with a problem. Let us try and see what the problem is.

Beaten (Anand,Gary) means Win(Anand,Gary). We all know that, it is obvious. Anyway, the program does not know that. We will have to add that rule now.

6. $\forall x \forall y$Beaten (X, Y) →Win (X, Y)

Now we can start working on it using backward chaining. We will deploy backward chaining in the following fashion. We will write the item to be proved first, try to see how we can prove that by picking up predicates defined so far. Find out one RHS matching with our description. Take the LHS with variables bound to values which make that statement true. If the statement is true, we will eliminate that statement. We may continue till we exhaust and there is no item left to be proved.

playedMatch(Anand,Gary)

←Win (Anand, Gary) // statement 5 with binding X=Anand, Y=Gary

←Beaten (Anand, Gary) // statement 6 with binding X=Anand, Y=Gary

←<true> // statement 4

Great! We have proved that Anand has played a match with Gary!

We have learned a few things from this simple problem representation in predicate logic and proving from that set of statements using backward chaining. Let us summarize what we have learned.

1. The statements, many a times, are ambiguous. We must need to get right presentations out of them. For example Beaten (Anand, Gary) is about Anand winning a game and not beating Gary up.
2. There are possibly many ways to represent the facts. Some are simpler than others. Complex representations are useful for finer reasoning but are hard to design and make the representation more difficult to understand. One must learn to find how these representations to be put to use before deciding their level of representation.
3. In most cases, some obvious facts are missing in the description. For example, we need to add that when somebody is beating somebody (in a game of some sort), he is actually winning it. Such a common sense statement is hard to think of in the beginning. Unless similar common statements are added, it is not possible for anybody to reason with and prove or disprove anything.

Another question. Can we prove that Gary was defeated? Not unless we add another statement

7. ∃xBeaten (X,Y) →Defeated (Y)

What does that mean? If we find anybody who has won against Y, we consider Y defeated. Notice the use of existential quantifier and not a Universal quantifier. This is another example of a common sense reasoning.

One will have to understand the difference in following statements and therefore, their corresponding representation.

1. Every student likes all AI modules
   $\forall x \forall y$ Student(X) ⋀ Module (Y) →Likes (X,Y)
2. Every student likes an AI module
   $\forall x$∃y Student(X) ⋀ Module (Y) →Likes (X,Y)
3. All students like the AI module 2
   $\forall x$ Student(X) ⋀Module (AI2) →Likes (X,AI2)

Can you see the difference? The statements are almost identical, first two indeed are. The difference is made by the quantifiers. In case number 1 the statement is true for all students and all modules. In case number 2 it is not true for all modules. Every student likes a module but may be different for different students and thus we have to use existential quantifier. The last one is a typical module which all liked so in the last case, it is a constant value and not a variableand thus no quantification is needed.

We will continue to discuss predicate logic, representation of natural statements into predicate logic, problems and solutions, alternate methods to represent such statements in subsequent modules.

Summary

We began with need to use formal logic in representing and inferring from known facts we learn through our senses. We have looked at propositional logic in the beginning. In the propositional logic, the facts are represented by symbols. Symbols might be represented as a single character or as a phrase for improving readability. The problem with propositional logic is that it is hard to relate similar things. To handle that problem, we have introduced the predicate logic which does not only identify the predicate, but provide arguments to make sure the finer meaning is retained for processing and interring. It is possible to use both values and variables with predicate logic. Only when the variables are assigned values the predicate gets its meaning and can be ascertained to be

true or false afterwards. Both universal and existential quantifiers are useful in representing different shades of meaning for simple statements.