Lecture: Basics of Logic Programming Md.
Shahriar Karim

Logic Programming [Incomplete]

Imperative programming languages such as, C, C++, Java etc. were developed following the von Neumann architechure, where the instructions ....

- Programming that uses symbolic logic as the programming language is known as logic programming. This type of programming uses a database full of facts (known as proposition in logic) and rules that demonstrate relationship between the given facts. An inference approach then decides the possibility of new propositions, assuming the given set of facts and rules as true.
- The concept of logic programming is significantly different than the imperative and functional programming languages.
- Logic programming language is also called as delcarative language– a notion later we see that coincides with definition of proposition, known as the fundamenalt building block of logic.

Definition: [Syllogistic form of Logic]: Aristotle provided the syllogistic form, in which a conclusion is drawn from the two stated premises; one of the premises is known as the major premise, and the other one is a minor premise. Example of a syllogistic form of logic would be as follows:

(S1)
Every man except Socrates is musician
Socrates is a man.

Socrates is not a musician $\therefore$

Another example of syllogistic form:

(S2)
Man is mortal
Socrates is a man.

Socrates will die $\therefore$

One important aspect of the stated premises is that, they delcare a definitive truth values (either True or False) in each sentences. So, Aristotle, in the Syllogistic form, gave us the notion of a type of declarative sentence with definitive true or false value. This delclarative sentence is known as **proposition**.

Basics of Propositional Logic

Definition: Propostition: Proposition is the basic building block of the logic and is defined as– a setence that is either true or false, but not both. The following declarative sentences are propositions:

- Dhaka is the capital of Bangladesh. [Because, it declares a **true** value]
- $7 + 19 = 33$ [It declares a **false** value]
- $4 + 7 = 11$ [It declares a **true** value]

The above discussion reveals that a proposition must declare either a true or a false value. Any sentence with inconclusive **T** or **F** value is not a propostion. A few such examples are as follows:

- Where are you going? [Indefinite/Uncertain]
- $x + 19 = 33$ [x is unknown]
- $x + y = 2z$ [x, y, z are unknown]

Notations: Propositions are represented using Propositional variables– conventionally, p, q, r, s, t are a few variables that are often used as propositional variables. For the Truth value and False value of a proposition **T** and **F** are used respectively.

Negation: Let p be a proposition, the negation (symbol $\neg$) of p is denoted as $\neg p$. Say, p stands for p : Today is Friday, then negation of p is: *It is not the case that today is Friday.* This also can be stated as: *Today is not Friday.* So,

- If the proposition p has a truth value T, negation of p will have F as the truth value. The negation ($\neg p$) of p actually means the denial of p.
- Truth Table for negation is as follows:

p	$\neg p$
T	F
F	T

Table 1: The truth table for the negation of a propostion

Definition: Compound Proposition: Propositions can be formed by combining multiple propositions. The new propositions are known as compound propositions. Each compound proposition has its definitive truth value, and the value directly depends on the consituent propositions.

- Some of the phrases, such as *and*, *or*, *but*, *If*, *when*, *necessary*, *sufficient* are a few words that are often used to combine multuple propostions together.
- Consider two propositions– i) p: *It is raining today*, ii) q: *I am driving my car*. The two propositions can be combined using an *and* as: *It is raining today and I am driving my car*. Here, *and* works as the connective between p, q, and is also known as logical operator.
- A few logical operators often used in logic are- i) Conjunction, ii) Disjunction, iii) Exclusive or, iv) Implication, v) Bidirectional.

Definition: Conjunction: Let p and q be propositions. The *conjunction* (also, known as *and*) of p and q is denoted as $p \wedge q$. Truth value of the compound propostion depends on the truth values of both p, q and is listed out in the below table.

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Table 2: The truth table for *conjuction* of two propostions.

As evident from the above table, the compound proposition $p \wedge q$ is true when both p, q are true (T). It is false when either of p, q is false (F).

Definition: Disjunction: Let p and q be propositions. The *disjunction* (also, known as *or*) of p, q is denoted as $p \vee q$. Truth value of the compound proposition depends on the truth values of both p, q and is listed out in the below table.

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Table 3: The truth table for *disjunction* of two propositions.

As evident from the bewlo truth table, the compound proposition $p \vee q$ is true when at least on of p, q is true (T). It is false only when both p, q are false. Consider two propositions– i) p: *It is raining today*, ii) q: *I am driving my car*. Disjunction of p, q would be: *It is raining today and I am driving my car*

Definition: Exclusive or: Let p and q be propositions. The *exclusive or* of p, q is denoted as $p \oplus q$, is the proposition that is true (T) when exactly on of p, q is true and is false otherwise. Application of *exclusive or* is often seen in buffet offer in restaurant!

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Table 4: The truth table for *exclusive or* of two propositions.

Definition: Conditional Statements: Let p and q be propositions. The *conditional statement*, denoted as $p \rightarrow q$, is the proposition *If p, then q*. Here, p is called *hypothesis (or premise or antecedent)* and q is *conclusion (or, consequence).* A few examples of $p \rightarrow q$ are as follows:

- If *you study hard*, then *you do well in exam.* Here, p: *You study hard* and q : *You do well in exam*
- Assume p:*Maria learns discrete mathematics*, q: *Maria will find a good job*, the $p \rightarrow q$ form of conditional statement becomes *If Maria learns discrete mathematics, then Maria will find a good job.*
- Similar meaning can be conveyed through *Maria will find a good job when she learns discrete mathematics*
- Similar meaning can be conveyed through *For maria to get a good job, it is sufficient for her to learn discrete mathematics*

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Table 5: The truth table for *conditional statement* of two propositions.

Definition: Biconditional Statements: Let p and q be propositions. The *biconditional statement*, denoted as $p \leftrightarrow q$, is the proposition *p if and only if q*. The biconditional statement $p \leftrightarrow q$ is true (T) when p and q have same truth values, and is false otherwise. The truth table is as follows:

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Table 6: The truth table for *biconditional statement* of two propositions.

There are a few other alternative expressions for biconditional statement, and those are as follows:

- p is necessary and sufficient for q
- If p then q, and conversely.
- p iff q
- $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
- An example: p: *You can take the flight*, q : *You buy a ticket*. So, the $p \leftrightarrow q$ form would be *You can take the flight if and only if you buy a ticket*

Practice 1: Construct the truth table of the compound proposition $(p \vee \neg q) \rightarrow (p \wedge q)$

Practice 2: Translate the english sentence into a logical expression: *You can access the Internet from campus only if you are computer science major or you are not a fresman.*
Answer to 2: a : *You can access the Internet from campus*, c : *You are a computer science major*, and f : *You are freshman*. So, finally the logical expression becomes $a \rightarrow (c \vee f)$

Logical Equivalence [Ongoing]

It is an important concept that provides transformed version of a given logical expression while keeping the truth values of the given logical expression unchanged. That is, for a given set of inputs, a logical expression and its logically equivalent form must have the similar outputs. The concept is explained further using a few examples as:

- Consider the two expressions $p \rightarrow q$ and $\neg p \vee q$. For a given set of inputs, both the expressions have similar outcomes as evident from the below truth tables:

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

p	$\neg p$	q	$\neg p \vee q$
T	F	T	T
T	F	F	F
F	T	T	T
F	T	F	T

- The symbol $\equiv$ is used to show the equivalence between two logical expression. As $p \rightarrow q$ is equivalent to $\neg p \vee q$, it can be represented as $p \rightarrow q \equiv \neg p \vee q$
- **Practice:** By using truth table show that a logically equivalent form of $p \vee (q \wedge r)$ is $(p \vee q) \wedge (p \vee q)$.

De Morgan's Law

These laws allow us to take the negation of compound propositions formed using conjunction and disjunction. Precisely, the laws provide equivalent form of $\neg(p \wedge q)$ and $\neg(p \vee q)$. Assume that p, q are propositions, compound propositions formed using p, q and their equivalent forms are as follows:

Name	Symbol	Example	Meaning
negation	$\neg$	$\neg p$	not p
conjunction	$\wedge$	$p \wedge q$	p and q
disjunction	$\vee$	$p \vee q$	p or q
equivalence	$\equiv$	$p \equiv q$	p is equivalent to q
implication	$\rightarrow$	$p \rightarrow q$	p implies q
biconditional	$\leftrightarrow$	$p \leftrightarrow q$	p if and only if q

Table 7: Detials of the logical connectors.

- $\neg(p \wedge q) \equiv \neg p \vee \neg q$
- $\neg(p \vee q) \equiv \neg p \wedge \neg q$
- Let us use De Morgan's law to express the negation of: *I shall go to the concert or I shall go to the theatre*. Assume that p*: I shall go to the concert*, $q : Ishallgotothetheatre$. The given statement becomes: $p \vee q$. Using De Morgan's law we obtain: $\neg(p \vee q) \equiv \neg p \wedge \neg q$. That is, *I shall not go to the concert and I shall not go to the theatre*.
- *Practice:* Use truth table approach show that the above laws are equivalent as stated.

Predicate Logic

Consider a situation– suppose, you are working as the network administrator at the computer center of your university and you know that every computer that is in the university network is working perfectly. Given that, it is impossible to express the below statements using the standard propositional logic.

- CSE131 computer in the network is working perfectly.
- CSE 111 computer is under attack by an evesdropper/intruder.

Also, suppose your instructor wants to convey that *All students in his class is good*, and to do so, he needs to use separate proposition for each student. This is cliche, and a more efficient approach is warranted. A powerful concept that works perfectly in the above cases is the **predicate logic.**

Definition: Predicate: Consider mathematical statements, such as $x > 3$, $x + 7 = y$, $x + y = z$.

- The statements are not propositions, as they do not possess definitive truth values (True or False). However, above statements are neither true nor false; instead, their truth values are context dependent. This means, for instance in $x > 3$, if x is replaced by 4, the statement is True.

- $x > 3$ reads as *x is greater than 3*. Here, x is the subject, and the part *is greater than 3* is syaing about the subject x. The part *is greater than 3* is known as the Predicate, and the statement can be represented as $P(x) : x$ is greater than 3.
- When x is fixed at a certain value, $P(x)$ generates a T/F value. Pecisely, if $x = 7$, then *7 is greater than 3* becomes **T** and the statement *7 is greater than 3* becomes a proposition. So, $P(x)$ works a propositional function, and it generates a propostion when value of x is fixed at a certain value.
- Anter alternative way to fix the value of x is known as **Quantification**; it quantifies x over a range of values of x.

Quantification:

This process produces a proposition from a propositional function. Precisely, quantification process decides the extent to which a given predicate is true. Let's consider $P(x)$ is a propostional function that stands for: *x is greater than 3*. Here, x is the variable that are fixed to different values from the allowable set of values. In the quantifiaction process, x is known as the bound variable, and the set of possible values of x is called the **domain of discourse**, or shortly as domain.

- The English words all, some, a few, many, more, each, every are used in the quantification process. Two main types of quantification are: i) universal quantification, ii) existential quantification.
- ***Universal quantification:*** This quantification process suggests that the predicate in a propositional function is true for all the possible values of the variables being considered. Given, a propositional function $P(x)$, universal quantification decides if $P(X)$ generates **T** for all the values of $x \in Domain$.
  - $\forall x$ is used as the notation for the phrase *for all x*. So, $\forall x P(x) \equiv$ *For all values of x in the domain* $P(x)$
  - It works as a series of conjunction among the set of propositions obtained through fixing x at its different values obtained from the domain: $P(x_1) \wedge P(x_2) \wedge P(x_3) \ldots P(x_n)$

Quantifier	**Statement**	**When True?**	**When False?**
Universal	$\forall x P(x)$	P(x) is T for each x	For at least single x, P(x) is F
Existential	$\exists x P(x)$	There is at least an x such that P(x) is T	P(x) is F for each x

Table 8: The two Quantifiers

Rules of Inference	Name
p $p \to q$ $\therefore \quad q$	Modus Ponens
$\neg q$ $p \to q$ $\therefore \quad \neg p$	Modus Tollens
$p \to q$ $q \to r$ $\therefore \quad p \to r$	Hypothetical Syllogism
$p \wedge p$ $\therefore \quad p$	Simplification

Rules of Inference	Name
p $\therefore \quad p \vee q$	Addition
$\neg p$ $p \vee q$ $\therefore \quad p$	Disjunctive Syllogism
p q $\therefore \quad p \wedge q$	Conjunction
$p \vee q$ $\neg p \vee r$ $\therefore \quad q \vee r$	Resolution

Table 9: Rules of Inference

Rules of Inference

Propositional Logic

Example 1

It is not sunny this afternoon and it is colder than yesterday. We will go swimming only if it is sunny. If we do not go swimming, then we will take a canoe trip. If we take a canoe trip, then we will be home by sunset.
Conclusion: We will be home by sunset.

p: It is sunny this afternoon q: It is colder than yesterday r: We will go swimming s: We will take a canoe trip t: We will be home by sunset

Let's apply rules of inference to see if the conclusion can be reached.
1. $\neg p \wedge q \quad (Premise)$
2. $r \to p \quad (Premise)$
3. $\neg r \to s \quad (Premise)$
4. $s \to t \quad (Premise)$
5. $\neg p \quad (Simplification\ rule\ on\ 1)$
6. $\neg r \quad (Moduls\ Tollens\ between\ 1\ and\ 5)$
7. $s \quad (Modus\ Ponens\ between\ 3\ and\ 6)$
8. $t \quad (Modus\ Ponens\ between\ 4\ and\ 7)$

Predicate Logic

Example 1

A student in this class has not read the book. Everyone in this class passed the first exam. Therefore, someone who passed the first exam has not read the book.
Assuming All NSU students as the domain.
$C(x)$: x is student in this class.
$B(x)$: x has read the book.
$P(x)$: x has passed the exam.
By applying rules of inference,

1. $\exists x(C(x) \wedge \neg B(x))$
2. $\forall x(C(x) \rightarrow P(x))$
3. $C(a) \wedge \neg B(x)$ $(Existential\ instantiation\ of\ 1)$
4. $C(a)$ $(Simplification\ of\ 3)$
5. $C(a) \rightarrow P(a)$ $(Universal\ instantiation\ of\ 2)$
6. $P(a)$ $(Modus\ Ponens\ of\ 4\ and\ 5)$
7. $\neg B(a)$ $(Simplification\ of\ 3)$
8. $P(a) \wedge \neg B(a)(Conjunction\ of\ 6\ and\ 7)$
9. $\exists x(P(a) \wedge \neg B(a))(Existential\ generalization\ of\ 8)$

Resolution Principle

Resolution principle, discovered by Alan Robinson in 1965, is used in the most widely used logical programming, namely the Prolog. In fact, resolution principle is the primary activity of a Prolog interpreter. The resolution method is also used to check the validity of an argument. Precisely, it is an inference rule that allows to devise propositions from a given set of propositions. This approach is widely used

Connection between resolution principle and Prolog

Consider a family relation among three persons Serena (**S**), Mike (**M**), Gloria (**G**). Their relations are stated as per the below facts. A logic rule is defined to decide if x is the aunt of y as in Rule 1. With the defined facts and the rule as stated below, for a query in Prolog to know if Serena is the aunt of Mike Prolog shows YES/NO. How a Prolog interpreter uses resolutions principle to finally resolves the query is demonstrated here.

Facts (F):

Female (S)	[***S** is Female*: F1]
Sister (S, G)	[***G** is a sister of **S***: F2]
Parent (G, M)	[***G** is a parent of **M***: F3]

Rules: $\text{Aunt}(x, y) :- Female(x), Sister(x, z), Parent(z, y)$ [*Rule 1*]

Rule 1 can be written with proper logical notation (such as,) as:

$\forall x \forall y \forall z [Female(x) \wedge Sister(x, z) \wedge Parent(z, y) \rightarrow Aunt(x, y)]$

Using the given facts $F1, F2, F3$, we can instantiate (that is, fixing specific value for the variables x, y, z) the Rule 1 as:

$[Female(S) \wedge Sister(S, z) \wedge Parent(z, y) \rightarrow Aunt(x, y)]$
$\equiv \neg(Female(S) \wedge Sister(S, G) \wedge Parent(G, M)) \vee Aunt(S, M)$ [Definition of implication]
$\equiv \neg Female(S) \vee \neg Sister(S, G) \vee \neg Parent(G, M) \vee Aunt(S, M)$ Rule 1A [De Morgan's Law]

Conclusion to be sought: $Aunt(S, M)$.

Considering the stated facts $F1, F2, F3$, negated conclusion ($\neg Aunt(S, M)$), and the instantiated rule as in $Rule1A$ as clauses ($\mathbf{C}$), resolution principle can be used to see if the empty clause ($\square$) is achievable.

$C_1 : Female(S)$
$C_2 : Sister(S, G)$
$C_3 : Parent(G, M)$
$C_4 : \neg Female(S) \vee \neg Sister(S, G) \vee \neg Parent(G, M) \vee Aunt(S, M)$
$C_5 : \neg Aunt(S, M)$
$C_6 : \neg Sister(S, G) \vee \neg Parent(G, M) \vee Aunt(S, M)$ [Resolution between C_1, C_4]
$C_7 : \neg Parent(G, M) \vee Aunt(S, M)$ [Resolution between C_2, C_6]
$C_8 : Aunt(S, M)$ [Resolution between C_3, C_7]
$C_9 : \square$ (empty clause) [Resolution between C_5, C_8]

In Prolog, the command syntax *? Aunt(S, M)* results YES. Here, Prolog tries to find rules and facts and using the appropriate instantiation process it tries to see if the goal is satisfied. This process is known as **backward chaining.**