# Propositional logic

1. Logic is about reasoning. As rational beings (hopefully!) we are often using some process of reasoning to make decisions or draw conclusions. Logic attempts to formalise some of these processes. Supposing your friend Kim tells you that she can come to a music concert only if her mother arrives at their home on time. You find her at the show. You will probably conclude that her mother must have arrived on time. Observe that Kim's statement actually has two sentences, roughly: 1. Mother arrives on time. 2. Kim goes to the show. Her statement connects these two statement by use of "only if". Propositional logic studies statements made up of basic or atomic sentences by using *logical connectives*. Some of the most common connectives are OR, AND, IF, ONLY IF etc. There is also the single-sentence operator NOT. Formal logic is indifferent to whether the atomic sentences are true or false. Its main concern is determining the truth (or falsity) of compounded statements from the truth and falsity of the individual constituents. Let the symbol $p$ stand for the sentence 'the sun rises in the west'. This is obviously false from our experience in our planet. But it is true on Venus! In logic we are not concerned about the truth of such atomic sentences. What we consider true is something like 'the sun rises in the west or it does not rise in the west'. We write this as $p \vee \neg p$. This is declared to be always true. It is a *tautology*.

2. Start with a set of letters $p, q, r, \ldots$ sometimes with subscripts which will be called propositional variables. They stand for atomic sentences. You can think of these as statements which can be assigned a truth value **true** (`T`) or **false** (`F`). For example, a statement like 'Alice is student of the course COMP9020' is either true or false. We just have to check the truth of a single statement. But suppose I say 'Alice *and* Bob are students of the course COMP9020' then to check its truth you have to verify *two* statements: 'Alice is student of the course COMP9020' and 'Bob is student of the course COMP9020' . My statement would be true if both these atomic sentences are true. So my statement is a compound statement consisting of two atomic sentences joined by an 'and'. `AND` ($\wedge$) is an example of a logical connective. Some other commonly used connectives are $\vee$ (`OR`), $\Rightarrow$ (implies) and $\Leftrightarrow$ (implies and is implied by). These are all binary operators. They connect two propositional formulas. There is an important *unary* operator $\neg$ (`NOT`). It acts on a single formula.

3. We can now recursively define a *formula* of propositional logic.

(a) Every propositional variable $p$ is a formula. These are called atomic formulas.

(b) If $\Phi$ is formula then $\neg\Phi$ is formula. Atomic formulas $p, q, \ldots,$ and their negations $\neg p, \neg q, \ldots,$ are called literals.

(c) If $\Phi_1$ and $\Phi_2$ are formulas then so are $\Phi_1 \wedge \Phi_2$, $\Phi_1 \vee \Phi_2$, $\Phi_1 \Rightarrow \Phi_2$ and $\Phi_1 \Leftrightarrow \Phi_2$.

(d) Finally, we need parentheses to remove ambiguities. So we posit if $\Phi$ is formula then $(\Phi)$ is also a formula.

4. Study this recursive definition carefully. You will come across this type of definition frequently in computer science. The definition gives a recipe for building legitimate formulas. For example, we will reject $\vee p$ as a formula because it dos not confirm to the recipe.

   Remember what the definition is doing. Let us start with a finite set $\Sigma = \{p_1, p_2, \ldots, p_n\} \cup \{\neg, \vee, (, )\}$. $\Sigma$ is our alphabet. The set of legal formulas $\mathcal{F}$ is a subset of $\Sigma^*$. The recursive definition of a formula also defines this subset. You can use the definition to write an algorithm to check whether a given word in $\Sigma^*$ is formula or not.

   You cam also use the recursive nature of the definition to prove theorems about formulas. Suppose you want to prove that some assertion is true for all formulas. You prove it first for atomic formulas. Then you prove that if the assertion is true for $\Phi_1, \Phi_2$ it is true for $\neg\Phi_1$ and $\Phi_1 \vee \Phi_2$ and you are done! This proof trick is called structural induction. You are using induction on the structure of the formulas.

   Take a few moments to think about this. It is important!

5. You can build formulas using only two operators. Any of the pairs $(\neg, \vee)$, $(\neg, \wedge)$ or $(\neg, \Rightarrow)$ will do.

6. We actually define $\Phi \Rightarrow \Psi$ to be $\neg\Phi \vee \Psi$.

7. Atomic formulas which are same as propositional variables can be true or false. Logic is not too concerned about the truth or falsity of the atomic formulas. What is important is the truth of a compound formula. This is defined recursively (again!) as follows. First given the set of formulas a truth assignment is a map $v : \mathcal{F} \rightarrow \{\mathtt{T}, \mathtt{F}\}$. Thus $v(\Phi) = \mathtt{T}$ means that the formula is evaluated to be true.

   (a) $\neg\Phi$ is true if $\Phi$ is false and false if $\Phi$ is true.

   (b) $\Phi_1 \vee \Phi_2$ is true if at least one of them is true.

   (c) $\Phi_1 \vee \Phi_2$ is true only in case both $\Phi_1$ and $\Phi_2$ are true.

   (d) Since $\Phi \Rightarrow \Psi$ is by definition same as $\neg\Phi \vee \Psi$ it is true if $\Phi$ is false (so $\neg\Phi$ is true) or $\Psi$ is true.

   The interpretation of $\Phi \Rightarrow \Psi$ may seem strange in the beginning, especially the fact that it is true if $\Phi$ is false. It might seem counter-intuitive to our ordinary reasoning. Let us look at an example: if it rains I will stay at home'. Let $p$ stand for "it rains' and $q$ for 'I will stay at home". We can represent the full statement as $p \Rightarrow q$. When will you consider it to be false? If it rains and I do *not* stay at home. That is, $p$ is true and $q$ is false. This is the only case in which it is false. It is true otherwise.

So we can determine the truth value of a formula from the truth values of its components. Using the recursive definition we can see that the evaluation of a formula (determining whether true or false) depends only the truth value assigned to the variables (the atomic formulas). A nice way of presenting the evaluation of formulas is by using of the truth table. Let us draw few basic truth tables.

| $p$ | $\Phi = \neg p$ |
|---|---|
| F | T |
| T | F |

| $p$ | $q$ | $\Phi = p \vee q$ |
|---|---|---|
| F | F | F |
| F | T | T |
| T | F | T |
| T | T | T |

| $p$ | $q$ | $\Phi = p \wedge q$ |
|---|---|---|
| F | F | F |
| F | T | F |
| T | F | F |
| T | T | T |

| $p$ | $q$ | $\Phi = p \Rightarrow q$ |
|---|---|---|
| F | F | T |
| F | T | T |
| T | F | F |
| T | T | T |

8. The tables are almost self-explanatory. At the top row you write the variables and the final column in that row is the formula $\Phi$ we are trying to evaluate. Below the variables the different truth assignments of the variables and the final column corresponds to the valuation of the formula $\Phi$. If you have $n$ variables how many possible assignments are there? *Ans.* $2^n$. So for $n$ variables there are $2^n$ rows in the truth table.

9. A formula $\Phi$ is called a tautology if it is always true. Thus in the truth table for $\Phi$ you will only see T in the last column. Conversely, a formula is called *unsatisfiable* if it is always false. If $\Phi$ is a tautology then $\neg\Phi$ is unsatisfiable. A very important problem in computer science is checking the satisfiability of propositional logic formula. It is called **SAT**.

10. Two formulas $\Phi, \Psi$ are called equivalent if they evaluate to same truth values. We write $\Phi \equiv \Psi$. $\Phi \equiv \Psi$ if and only if $\Phi \Leftrightarrow \Psi$ is a tautology. That is, one is true or false if the other is same. So it is enough to draw the truth table of one formula only. We generally identify the two formulas on either side of $\equiv$. A simple example is $\neg\neg\Phi$ and $\Phi$. Some more examples follow.

11. Examples of some equivalent formulas.

    (a) $\Phi \wedge \Psi \equiv \neg(\neg\Phi \vee \neg\Psi)$.

    (b)
    $$\Phi_1 \Rightarrow (\Phi_2 \ldots (\Phi_n \Rightarrow \Psi) \ldots) \equiv (\Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_n) \Rightarrow \Psi \tag{1}$$

    (c) Let us try to prove the last equivalence as an exercise in induction. We will use induction on $n$, the number of implicants ($\Phi$'s). Clearly, if $n = 1$ both sides are $\Phi_1 \Rightarrow \Psi$ and there is nothing to prove. Let us see how we can prove it for $n = 2$. We have to show $\Phi_1 \Rightarrow (\Phi_2 \Rightarrow \Psi) \equiv (\Phi_1 \wedge \Phi_2) \Rightarrow \Psi$. Now draw up the truth table. You may ask the $\Phi$'s and $\Psi$ are formulas in general and so will have any number of propositional variables. So how do we evaluate them if we don't know about these variables? Actually, we don't need to know about the constituents to determine equivalence. Each $\Phi_i$ and $\Psi$ is either true or false. So if we cover all cases and show that in each case both sides of $\equiv$ have the same truth value then we are done. We are basically treating the formulas ($\Phi_i$'s and $\Psi$) as propositional variables. Whatever their actual truth value it is covered. So the table is

| $\Phi_1$ | $\Phi_2$ | $\Psi$ | $\Phi_1 \Rightarrow (\Phi_2 \Rightarrow \Psi)$ | $(\Phi_1 \wedge \Phi_2) \Rightarrow \Psi$ |
|---|---|---|---|---|
| F | F | F | T | T |
| F | F | T | T | T |
| F | T | F | T | T |
| F | T | T | T | T |
| T | F | F | T | T |
| T | F | T | T | T |
| T | T | F | F | F |
| T | T | T | T | T |

There are 3 propositional 'variables' $\Phi_1, \Phi_2$ and $\Psi$ and so 8 rows in the table. The table itself is not difficult to fill. Just remember that the formula $X \Rightarrow Y$ is false only when $X$ is true and $Y$ is false. Now convince yourself about the correctness of the entries in the table. We see that $\Phi_1 \Rightarrow (\Phi_2 \Rightarrow \Psi)$ and $(\Phi_1 \wedge \Phi_2) \Rightarrow \Psi$ have the same truth value as the last two columns are identical. So theya re equivalent.

We will use induction for the general case. So assume the statement is true for $n-1$. We will show that it is true for $n$ completing the induction. Write the formula $\Phi_n \Rightarrow \Psi$ as $\Psi'$. Then the left side of the formula (1) can be written as $\Phi_1 \Rightarrow (\Phi_2 \ldots (\Phi_{n-1} \Rightarrow \Psi') \ldots)$. Note that there are only $n-1$ implicants $(\Phi_1, \ldots, \Phi_{n-1})$. So by the induction hypotheses

$$\begin{aligned}\Phi_1 \Rightarrow (\Phi_2 \ldots (\Phi_n \Rightarrow \Psi) \ldots) &= \Phi_1 \Rightarrow (\Phi_2 \ldots (\Phi_{n-1} \Rightarrow \Psi') \ldots) \\ &\equiv (\Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_{n-1}) \Rightarrow \Psi'\end{aligned}$$

Now write the last line $(\Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_{n-1}) \Rightarrow \Psi' = \Gamma \Rightarrow \Psi'$ where $\Gamma$ stands for the formula $\Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_{n-1}$. We can now write the last line as $\Gamma \Rightarrow (\Phi_n \Rightarrow \Psi)$ recalling the definition of $\Psi'$. But this is a formula with two implicants $\Gamma$ and $\Phi_n$. We have covered this case above. Hence $\Gamma \Rightarrow (\Phi_n \Rightarrow \Psi) \equiv (\Gamma \wedge \Phi_n) \Rightarrow \Psi$. Substituting the expression for $\Gamma$ we prove the theorem.

Manas Patra