Propositional logic 2

We are going to talk about propositional logic, Boolean algebras and related topics in the next fwew notes. Hopefully, some of the confusion in these abstract concepts will be cleared up.

1. The first important point is the difference between syntax and semantics. Syntax is the symbolic structure which is precisely defined and is (in principle) easily checked by a computer programme. Semantics is about assigning meaning to the syntactic structures. This is more subtle. Is the correspondence between the symbols and their assigned meaning correct, consistent, complete...? These are difficult questions. But for propositional logic the answers are known to be "yes". Let us start with a recap of the definition of a formula.

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`). We then introduce logical operators $\vee, \wedge,$ and $\neg$.

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 ϕ 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 ϕ_1 and ϕ_2 are formulas then so are $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$.

   (d) Finally, we need parentheses to remove ambiguities. So we posit if ϕ is formula then (ϕ) 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 $\Sigma = \{p_1, p_2, \ldots, p_n\} \cup \{\neg, \vee, \wedge, (,)\}$. Σ is our alphabet. The set of legal formulas $\mathcal{F}$ is a subset of Σ^*. 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 Σ^* is formula or not.

You can 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 ϕ_1, ϕ_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. The rules of building formulas tell you whether a word or string is a formula. For example, $p \vee \neg q$ is a formula but $\vee p \neg q$ is not. It is like checking the syntax in a programming language $a * b$ $(= a \times b)$ is an expression but $*ab$ is not (in most programming languages).

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

7. We know by de Morgan's laws that $\neg(\phi \vee \Psi) = \neg\phi \wedge \neg\Psi$. As words in the language of formulas they are certainly different so what is the meaning of the equality?

8. There are two possible approaches: the axiomatic approach and the semantic approach. The axiomatic approach identifies certain formulas like the ones above. It is like something you have seen for natural numbers: $a + b = b + a$. Although as expressions they are different they denote the 'same thing'.

9. In the axiomatic approach we write down some formulas as *axioms*. Then we lay down some rules deduction. Whatever new formulas you can derive from the rules is called a theorem. We will not follow this approach although variants of this approach are used in powerful tools like theorem provers.

10. In the second approach we build 'models' for the propositional variables. In these models each variable is assign a 'truth value', that is, each variable is assigned a value **true** (T or 1) or *false* (F or 0). We also give the rules for determining truth value of compound formulas from its parts. This is called a valuation. You have already seen it in the lecture notes and the previous supplementary notes. Important note: any axiom in the first approach (for example, $\phi \vee \neg\phi$) must be a tautology.

11. **Entailment.** The expression $\phi, \psi \models \theta$ means that whenever formulas ϕ and ψ evaluate to true θ has value true. Note that we only look at the rows truth table where both ϕ and ψ come out true. The other rows (where at least one of them is false) do not matter as far as entailment is concerned. That is why we say ϕ and ψ entail θ.

12. $\models \theta$ means that $\top \models \theta$. The symbol $\top$ is considered to be always true. So $\models \theta$ simply says that θ is tautology.

13. We can generalize to any number of formulas on the left side

$$\phi_1, \phi_2, \ldots, \phi_n \models \theta$$

This is equivalent to both

$$\models (\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n) \rightarrow \theta \text{ and } \models \phi_1 \rightarrow (\phi_2 \ldots (\phi_n \rightarrow \theta \ldots)$$

What this 'equivalence' means is that if any one of the entailment holds then all hold.

14. Let us look at some examples of entailment. Suppose we are given

$$\phi, \psi \models \theta_1 \vee \theta_2$$

Then we have only this much information:

ϕ	ψ	θ_1	θ_2
T	T	T	T
T	T	T	F
T	T	F	T

Can you see why? All we are given is that when both ϕ and ψ are true then $\theta_1 \vee \theta_2$ must be true. The table precisely reflects that. But it also says something that is not there. For example, we can add the row

T	F	T	T

it is an allowed valuation. What are the valuations that are not allowed? There is only one disallowed valuation:

T	T	F	F

What can you infer from the table ? Can we say that $\theta_1, \theta_2 \models \phi_1$? The simplest way to do this is to ask for what valuations does the entailment fail. The answer is :$v(\theta_1) = v(\theta_2) = T$ *and* $v(\phi_1) = F$. This is an allowed valuation for the original entailment $\phi, \psi \models \theta_1 \vee \theta_2$. So we *cannot* infer that $\theta_1, \theta_2 \models \phi$. How about $\neg\theta_1, \neg\theta_2 \models \neg\phi \vee \neg\phi_2$? Again this fails only if $v(\theta_1) = v(\theta_2) = F$ and $v(\phi_1) = v(\phi_2) = T$. This *is* disallowed as we have seen. So we can infer the last entailment.

Manas Patra