Propositional$_l ogic$

Viv Sedov — viv.sedov@hotmail.com

March 18, 2022

Contents

1 Propositional_logic

1.1 Simple Operations

When covering this : there are simple operations that you should know about :

Simple Operation Not:

p	$\neg p$
1	0
0	1

Such that in this case it is the opposite given value $\neg p$ would be the direct opposite of the given Value of P .

For example say that you have the following

p	q	$p \wedge q$
1	1	1
1	0	0
0	1	0
0	0	0

In this example, we are stating the following command : $\neg(14 > 6)$ is false , we create this truth table to prove if that is true or not .

$P \wedge Q$ is true $\iff$ p and q are true

Simple $\vee$ - OR

p	q	$p \vee q$
1	1	1
1	0	1
0	1	1
0	0	0

In this example above , for this to hold true, atleast one of teh values would have to have a one it to hold true, this is known as an *Inclusive* or as in you can say the following and it would make sense: **I will go to the shops $\vee$ i will go to the coast**

Simple Operation $\implies$ This is where if something is true the other must be true , or where you given an equivalent pointer to if p then q

p	q	$p \implies q$
1	1	1
1	0	0
0	1	1
1	1	1

With the above example this is not cause and effect , there is a reason for why this occours , and that is that there is a pointer such that it acts like an if statment, with the following code shown below :

foo = True if foo: return 1 else: return 0

The code above is rather simple , but shows that if something is true , then you would have some sort of value expression , or some pointer that would return if it is correct or not .

In the weird scenario of f and t , where if f is false it implies that q is true , that is because the q value is true , meaning that it would hold , a little trick for this one is that for what ever

the second value is , if it is true , it will hold , if both are false , then it will true , though if one is true and the other is false , it will not hold .

Logically equivalent $\iff$ this can also be seen as if and only if or iff
Here is an example of the logical truth table behind this

p	q	$p \iff q$
1	1	1
1	0	0
0	1	0
0	0	1

With this , Where if something is true then the other must be true or if its false then the other would have to be false , in this case you are seeing if these two values are the same .

Example Exersises of how this would all work :
Given that :

$$p = Logicisfunforjaneq = daviddoesnotlikecabbager = davideyesareblue$$

We can then further express all of this in a notational form .

- $\neg p \wedge q$ This would imply the following truth table

$\neg p$	p	q	$\neg p \wedge q$
0	1	1	0
1	0	1	1
0	1	0	0
1	0	0	0

- $\neg R \wedge \neg P \implies$ Both are not goint to be true such that you would get only one possible answer for this .

Multiple truth table example : show the following in a truth table :

$$p \wedge (q \implies r)$$

p	q	r	$q \implies r$	$p \wedge (q \implies r)$
1	1	1	1	1
1	1	0	0	0
1	0	1	1	1
0	1	0	0	0
0	1	1	1	0
0	0	0	1	0

Here is another example to understand how these tables would all work together

$$(p \implies q) \wedge (q \implies p)$$

p	q	$q \implies p$	$p \implies q$	$(p \implies q) \wedge (q \implies p)$
1	1	1	1	1
0	1	0	1	0
1	0	1	0	1
0	0	1	1	1

Definition 1.1 Two propositions are equal if they have teh same truth values , they are known as $P \implies Q$ this is known as logically equivalent

Definition 1.2 A proposition is tautology if it is always true an example of this is $P \vee \neg P$ this is always true

Definition 1.3 A proposition is a contradiction if it is always false for example $P \wedge \neg P$ this is always false

Definition 1.4 A proposition is **Contingent** if it is neither Always true or false

2 Disjunctive Normal Form

A given formula is said to be a *Disjunctive normal form* when it is an Or $\implies \vee$ this is known as **DNF** a function is conjunctive when it has an $\wedge$ form , within their proposition logic .

$$(p \wedge \neg q \wedge r) \vee (\neg q \wedge \neg r) \vee q$$

Everytime a given formula is built , we would folow the rules of propositional calculus , and how for each conjunctive formula there should be a disjunctive formula as well .

2.1 DNF of mini terms for truth tables

- For each row whose truth value is true , write doen for each of the prposition variables , of p_i in the formula of it self , either P_i is true in row or $\neg P_i$ if false.
- Repeate teh first pointer , for the truth table where the formula is true and write down the dijunction of the conjuctions .

What you will rsee is that those two values will equal up such that the result of teh formula in DNF is the equivalent to the original formula .

3 Conjunctive Normal Form

A formula is said to be **Conjunctive Normal Form** when its conjunction is $\wedge$ of disjunctive of $\vee$ an example of this is shown below :

$$(\neg P \vee Q \vee R \vee \neg S) \wedge (P \vee Q) \wedge \neg S \wedge (Q \vee \neg R \vee S)$$

Every expression built up according to the rules of calc , and suchj that for each conjunctive formula there is a similar or an equivelent formual that can be written in disjunctive form .

Conjunctive form , or in brackets , and on the outside
Disjunctive form, And in the brackets and or on the outside

4 Logical Equivalences

We can use this to obtain normal form , when we use the implication law to eliminate subproccess - when ever you have a doubel negation and demorgans to bring a ¬ you what this value to be on the outside : here are the sub process of how this can be done :

$$\neg\neg P \iff P$$

This rule is the double negation Law

$$\begin{aligned}(P \vee Q) &\iff (Q \vee P)\\ (P \wedge Q) &\iff (Q \wedge P)\\ (P \iff Q) &\iff (Q \iff P)\end{aligned}$$

Commutaive laws where both values would have to equal towards each other .

$$\begin{aligned}((P \vee Q) \vee R) &\iff (P \vee (Q \vee R))\\ ((P \wedge Q) \wedge R) &\iff (P \wedge (Q \wedge R))\end{aligned}$$

This is the associative laws , where it is very similar to how they work in matricies in which they can equate towards each other .

$$\begin{aligned}((P \vee Q) \wedge R) &\iff (P \vee (Q \wedge R))\\ ((P \wedge Q) \vee R) &\iff (P \wedge (Q \vee R))\end{aligned}$$

This lase is the distributive law , in which the given values would be changed within a DNF and CNF representation

$$\begin{aligned}(P \vee P) &\iff P\\ (P \wedge P) &\iff P\end{aligned}$$

Idempotent laws where the values of it self would always equal to it self no matter what .

Demorgans Law :

$$\begin{aligned}\neg(P \vee Q) &\iff (\neg P \wedge \neg Q)\\ \neg(P \wedge Q) &\iff (\neg P \vee \neg Q)\\ (P \wedge Q) &\iff \neg(\neg P \vee \neg Q)\\ (P \vee Q) &\iff \neg(\neg P \wedge \neg Q)\end{aligned}$$

Most times when you lookat demorgans law , you will notice that its very similar to teh laws that have been stated above, but the thing that you want to note is that you will see that they are equal in some sense , where an or , is a direct link with Not and And it self.

§ Contrapositive Laws

$$(P \implies Q) \iff (\neg Q \implies \neg P)$$

implication that imply towards each other are contrapositive and hence you can switch out the given details of that infomation .

where If Q is an active reciever then P must be an active pointer is teh same as stating if not p equates to Not q, in some sense you should understand how that would work .

§ Implication

$$(P \implies Q) \iff (\neg P \vee Q)$$
$$(P \implies Q) \iff \neg(P \wedge \neg Q)$$

§ Furhter implication

$$(P \vee Q) \iff (\neg P \implies Q)$$
$$(P \wedge Q) \iff \neg(\neg p \implies \neg Q)$$

Thi sone is rather annyoing , but the principle of how this works is very intriguing , if you do prove this via proof table you will see that they are truly equivalent: $P \vee Q$

p	q	$p \vee q$
1	1	1
1	0	1
0	1	1
0	0	0

this is the same as : $(\neg P \implies Q)$

P	Q	$p \implies Q$	$(\neg P \implies Q)$
1	1	1	1
0	1	1	1
1	0	0	1
0	0	1	0

If you look at the given tables above you will notice that indeed they are the same, a truth tabel may be long but they are very good at breaking down the given data that you have into something more readble .

Further Implies and equivalences

$$((P \implies R) \wedge (Q \implies R)) \iff ((P \vee Q) \implies R)$$
$$((P \implies Q) \wedge (P \implies R)) \iff ((P \implies (Q \wedge R))$$

With this law you are using the given equivelences that are shown above with the disjunctive and conjunctive views ,but within an equivelence ratio

§ Exportion Law

$$((P \wedge Q) \implies R) \iff (P \implies (Q \implies R))$$

This one is a good one , Mainly because if anything that does imply to another pointer , you can show that they are all equal towards each other .

§ Side Notes Within the compound proposition $\neg(P \vee Q)\&(\neg P \wedge \neg Q)$ they are the same , hence why when you look at the proof that is shown above you will see that they are the same .

When ever you look at equivelences you will notice that connectives $\vee\wedge$ will allways suggest that $P \vee Q \implies Q \vee P$

5 Sound and Completness without the bs

- Sound If the argument can be derived from A then it is valid $\gamma \vdash A$ then $\gamma \models A$ Such that our problem is sound, if

  $$\gamma / A$$

  What this means is that it does not have a counter example -¿ so if you have a counter example it is not sound

- Completness if
  $\gamma \models A$ then $\gamma \vdash A$ If an argument is a valid and if no counter example is given then it can in principle it can be derived.

We say something is complete iff all teh arguments that have no counter example can be derived. *What does this mean ?*
Soundness means that you cannot prove anything that is wrong
Completness means that you can prove anything that is right

in both cases we are talking about some fixed system of rules that we follow when proving some given variable we represent this proof with the following item

$$\vdash$$

Let me characterize soundness and completeness using a silly example. Suppose you are in the business of making machines which make widgets, and suppose that someone comes to you and says "I need a machine which makes red widgets which are either round or square". You go off and build a widget producing machine, and show it to your potential customer. To convert her from a potential to an actual customer, you must convince her of two things: first, that your widget machine will only produce square or round red widgets, and not blue widgets, and second, that your machine will produce both round red and square red widgets, and not only square red ones. If your machine satisfies the first requirement, then it is sound. If you machine satisfies the second requirement, then it is complete.

another example
Imagine I have a box and many crystal balls with color either black or white but not both.

- Sound : every ball inside the box are black but does not mean all the balls outside are white - sound says oh hey ive got a few negative values but i know that there is some properties that i want that exist

- Completness is Every ball outside the box are white, but does not mean all teh inside valls are black - complntness always accept black but not only.

6 logic entailment

Entailment is when you have a logical statement and you want to see if it is true or false , if it is true then you can say that it is true , if it is false then you can say that it is false , if it is unknown then you can say that it is unknown .

we can say if all models of alpha are subset of all subset in beta $\forall \alpha \in \beta$ then $\alpha \subseteq \beta$

what you need to know is that beta here can contain more things, it cna contain more information, it can have a set of the following $m(\alpha) \subset m(\ \beta)$
$\alpha \models \beta$

*text*we can say that beta can be a set of large ammount of data [false false true] but where alpha is true and is with beta has the power to contain more data that is not related to what alpha would require. Alpha is more specific than beta , and beta is more general than alpha. A good example with tables is teh following $((x \iff q) \vee r) \models (q \implies x))$

x	q	r	$q \implies x$
1	1	1	1
0	1	1	1
1	0	1	1
0	0	0	0

(1)

if x implies q and q implies x or R this models q implies x why ? because if x is true then q is true and if q is true then x is true , if x is false then q is false and if q is false then x is false , if x is unknown then q is unknown and if q is unknown then x is unknown.

6.1 Modus Ponenens

Defined through the following $\alpha \models \beta \iff \exists\gamma$ such that $\gamma \models \alpha$ and $\gamma \models \beta$ this reads as if alpha models Beta if and only if for some game where yame models alpha and game models beta.