18.014 Week 0 : Introduction to Mathematical Logic and Proof Writing

MIT OCW Discord

Lecture 0.1 : Mathematical Logic and Axiomatic Systems

0.1 Propositional Logic

Definition .1. A proposition p is a variable[a] that can take the values true (T or 1) or false (F or 0) and no others

[a]meaning just a formal expression without any structure

That's all a proposition is, it's something that can be either true or false. Here goes some more terminology that's related:

Definition .2. A proposition which is always true is called a *tautology*, while one which is always false is called a *contradiction*.

0.2 Unary Operators

Definition .3. One can build new propositions from given ones by using *logical operators*. The kind of logical operators that give you one proposition from one proposition are called *unary operators* and the ones that take two propositions to give one new proposition are called *binary operators*. There are in total 4 unary operators:

- The $\neg$ *negation* operator. E.g P: I am mad has $\neg$P as: I am *not* mad.
- The id *identity* operator. E.g. P: I am mad has id(P) as: I am mad. And, P': I am not mad has id(P') as: I am not mad. Stays

the same.

- The $\top$ *tautology* operation. E.g P: I am mad has $\top_P$ as: I am *always* mad. Me being mad is always true.
- The $\perp$ *contradiction* operator. E.g P: I am mad has $\perp_P$ as: I am *always not* mad. Me being *not* mad is always true, contradicting P.

These are all the possible ways you can have one proposition from another one.

Exercise .3.1. As an exercise try to fill the entries in this table yourself from your understanding of the unary operators:

P	$\neg P$	id(P)	$\top_P$	$\perp_P$
F	-	-	-	-
T	-	-	-	-

Also, as a way of practice, write down what each of the unary operations mean in English language while you fill in the above entries. This will help you get used to the concepts of negation, tautology and others while holding your intuition.

0.3 Binary Logical Operators

As defined before, a binary logical operators needs two proposition and then gives you a new proposition by "operating" on them. And as one can guess by trivial counting, there are 16 such possible operators, we have 4 combinations of the truth values of two propositions and as the binary operator is going to assign two possible truth values to each of those, you'll have 16 in total. We're not going to go into all 16 of them, but just mention a few that are important. Here are those:

- The $\wedge$ *and* operator.
- The $\vee$ *or* operator.
- The $\implies$ *implies* operator.
- The $\iff$ *equivalence* operator.

Exercise .3.2. Let's play with these operators for a while and put them in a table. Consider the following two propositions:

1. p: I read books all day.
2. q: I am mad.

Now lets see how the truth tables for each of these binary operators is going to look like for the first two propositions above. I've filled some of the entries, the rest are for you to exercise your muscles of logic.

0	p	q	$p \wedge q$	$p \vee q$
1	F	F	F	F
2	F	T	-	-
3	T	F	F	-
4	T	T	-	T

Here is an explanation in English words for how I filled some of the entries in the above table.

Explanation:

For the first row, what I have is, p and q are both not true. Thus, q:I am *not* mad and p: I *don't* read books all day. Now to put an entry in the 3rd row of 3rd column, you have to answer the following question: "*What truth value does p **and** q together have when p is not true and q is also not true?*" In other words, if me reading books all day is not true and I am not mad, can it be true that I am both mad *and* I read books all day? Notice that the second "and" in the last question is equivalent to the $\wedge$ operator on p and q. Now, as you can guess the answer to both the questions is NO. If p and q are not true, then p *and* q can't be true, either.
Now for the next column in the first row, we again have p and q both as false. But this time we'll have to look at $p \vee q$. You ask a similar question like before but replace the *and* with *or*. And you'll find the answer is the same. If p and q are not true, then $p \vee q$ can't be true.

Easy...heh? Use the same argument and you'll be able to fill all the entries in the table. Now let's get on to the more complicated binary operators, $\implies$ and $\iff$. We consider the same propositions p and q and make another truth table:

0	p	q	$p \implies q$	$p \iff q$
1	F	F	T	T
2	F	T	T	-
3	T	F	-	-
4	T	T	T	T

Explanation:

For the first and last row, the entry for $p \iff q$ should be obvious, if you have any of them with different truth values then they're not going to be equivalent. In other words, "I read books all day" is *equivalent to* "I am mad" only when both the statements are true or false at the same time. If one of them is true and the other false, you'll lose equivalence. If I read books all day but somehow it turned out that I am not actually mad, then me reading books all day can't be *equivalent* to me being mad. Makes sense? Good, that implies you're not mad!

Next for $p \implies q$ the last row should be trivial. The first row says that if both p and q are not true then what's going to be the truth value for $p \implies q$? You make a similar argument, "I don't read books all day" (p is false) and "I am not mad." (q is false) then one can say that "I read books all day" *implies* "I am mad". But the interesting thing is even if you make q true keeping p false (2nd row), this doesn't invalidate the implication. This can be surprising at first, that if p is false, no matter what q is, you can have a valid implication that is true. This is also known as "Ex falso quad libet" meaning "from a false premise (assumption) anything follows". Think a bit more on this while referring to the table if it isn't immediately clear to you. Try filling the rest of the entries in the table, by making similar arguments but without resorting to self-referential tautology.

And now I'll put in formal terms what we were trying to do in the lecture. Namely the important use of *contrapositive*. It's going to be obvious after the following theorem and it's explanatory theorem.

Theorem .4. *Let p and q be propositions. Then $(p \implies q) \iff (\neg q \implies \neg p)$.*

1. *Proof.* Let's try to understand what the theorem's statement means in the context of our original propositions. It's saying that if p being true implies q being true then that's *equivalent* to saying q being false *implies* p being false. So if "I read books all day" implies "I am mad" then it's the same as saying "I am not mad thus, I don't read books all day."[a]

Now let's try to draw the truth table again, we're going to copy our previous table of $p \implies q$.

p	q	$\neg q$	$\neg p$	$p \implies q$	$(\neg q) \implies (\neg p)$
F	F	T	T	T	T
F	T	F	T	T	-
T	F	T	F	F	F
T	T	F	F	T	-

I'll explain the first and third entries of the last column, but I'd like you to try the others, would be a good exercise of implications among negatives. $\square$

Explanation:

The first entry (of last column) is quite trivial, if both of your propositions–which here are negations–are true, then you can have a valid implication that holds. Translating things again we get, $\neg q$: I am not mad and $\neg p$: I don't read books all day, then it's true that me not being mad implies me not reading books all day ($\neg q \implies \neg p$), thus making the contrapositive hold. For the third entry, you have $\neg q$ as true, thus "I am not mad" and you have $\neg p$ is false, thus "I am reading books all day". And you have to answer the question, what is the truth value for $\neg q \implies \neg p$ or the statement "I am not mad implies I am reading books all day", which as you can see can't be true, me reading books all day isn't *necessary* for me being mad[b].

When you're done with filling all the entries for last column you'd realise that it's exactly the same as the previous column, and that was entire goal of the proof. Now both the columns and thus the statements are *equivalent*, an implication is equivalent to it's contrapositive. You'd see yourself using this simple-looking statement more than you can think now!

[a]This also works in the other direction $((\neg q \implies \neg p) \implies (p \implies q))$, and in proofs its called the "if and only if" condition. And when you're faced with such a statement you'll have to prove the statements in both the direction.

[b]There have been lunatics who haven't read a word!

Remark .5. One can semantically use the $\implies$ in various ways, we mention some of them here, all of them are equivalent to each other:

- If p, then q.
- q if p.

- p implies q.
- p only if q.
- p is sufficient for q.
- q is necessary for p.

1 Predicate Logic

Definition .6. A *predicate* is (informally) a proposition-valued function of some variable or variables. In particular, a predicate of two variables is called a *relation*.

Definition .7. Let $P(x)$ be a predicate. Then:

$$\forall x : P(x), .$$

is a proposition, which we read as "for all x, P of x (is true)", and it is defined to be true if $P(x)$ is true independently of x, false otherwise. The symbol $\forall$ is called *universal quantifier*.

Definition .8. Let $P(x)$ be a predicate. Then we define:

$$\exists x : P(x) :\iff \neg\left(\exists x : \neg P(x)\right).$$

Example .9. Let $P(x, y)$ be a predicate. Then, for fixed y, $P(x, y)$ is a prediacte of one variable and we define:

$$Q(y) :\iff \forall x : P(x, y).$$

Hence we may have the following:

$$\exists y : \forall x : P(x, y) :\iff \exists y : Q(y).$$

Definition .10. Let $P(x)$ be a predicate. We define the *unique existen-*

tial quantifier ∃ !by :

$$\exists ! x : P(x) :\iff (\exists x : P(x)) \wedge \forall y : \forall z : (P(y) \wedge P(z) y = z) .$$

2 Axiomatic Systems

3 Appendix : Introduction to Functions

3.1 Functions

Definition .11. A function (or map) $\phi : A \to B$ is a relation such that for each $a \in A$ there exists only one $b \in B$ such that $\phi(a) = b$. Here A is the **domain** of ϕ, B is the **codomain** and $\phi(A) = \{f(a) : \forall a \in A\}$ is the **image** of ϕ.

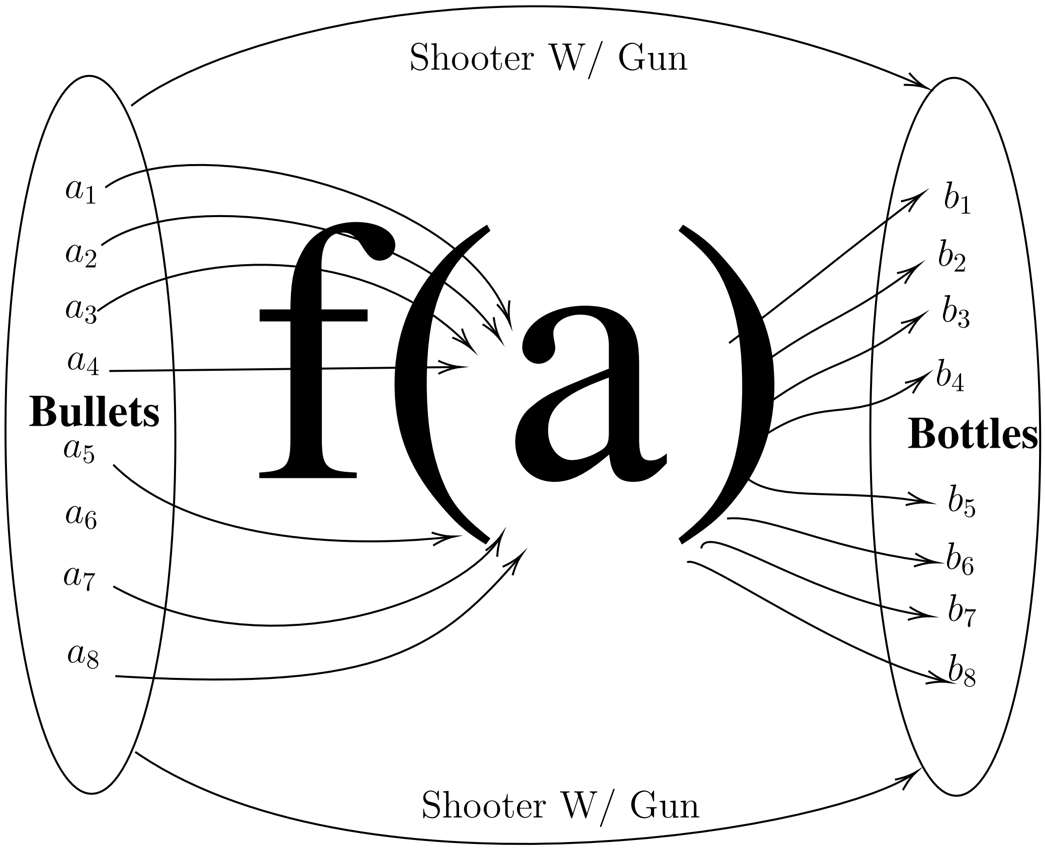

Figure of bullets (arguments) getting into the gun (function) and then shot by shooter
bottles (images).

3.2 Explaining Injectivty and Surjectivity

Consider a function to be like a gun (more like a pistol), the bullets in the gun to be elements in the domain, and coke bottles as elements in the codomain. So what our gun does is it hits a coke bottle with one bullet, the set of all coke bottles that got hit by all the bullets we fired is the image of our function. It's everything that got hit.

Now lets use this analogy to define injectivity and surjectivity. Consider that while shooting, we make it entirely sure that one bullet that gets fired from my gun hits *only one* bottle. So if I found bottles b_1 and b_2 were hit by the same gun, then I'm sure that those two bottles are one and the same.[1]Such a gun is defined as ***injective***, if two bottles got hit, then there *must* be two bullets that got fired. Do you see the problem with such a gun shooter? Its that if he follows this, he might not be able to shoot all bottles we have in the codomain. As, if the number of bullets is less than the number of bottles and he hits exactly one bottle with one bullet, then obviously he's going to miss a bunch of bottles.

Now consider another case, you have another shooter who uses the gun in such a manner that he *always* hits every bottle in the codomain. After a session of shooting, you're going to find that every bottle was hit and is broken. This is the case of a ***surjective function***, no matter what bottle we consider we're always going to find *at least* one bullet that hit it.[2] Can you see the problem with this shooter? He surely is able to hit every single bottle, but he probably has tried more than one bullet on some bottles out of which some got missed and only of them hit the bottle. In this case you can actually imagine the shooter having a double-barrel gun or something that can shoot more than one bullets at a time because we need *at least* one bullet to hit, but we can go for more! Thus this shooter is wasting those precious bullets!

So who's the *perfect* shooter? As some of you might have guessed, it'd be the shooter who has the ability to hit a bottle *exactly* with one bullet and has enough bullets to hit *each and every* bottle in the codomain. It's trivial to point out that this is a "special" shooter, he's actually the *nicest* of all shooters we can get. In mathematics we call such shooters (more like, functions!) ***bijective***, every bullet hits only one bottle and all bottles get hit[3]As you might've already guessed, this special shooter has the skills of both the other two shooters we hired, and thus a bijective function is one

[1]Mathematically it means $f(x) = f(x') \implies x = x'$.

[2]Mathematically, it means the whole image of the function is the codomain itself. So $\text{im}_\phi(A) = B$

[3]Mathematically you say it as a one-to-one correspondence, you have exactly one unique ordered pair of $(x, f(x))$ in each case.

which is both injective and surjective.

Graphically, you imagine a function to intersect (or more like, hit!) the graph at precisely one point, for it to be injective. And similarly for surjective you need the function's image to be the graph itself ("everything gets hit").