Logic

Aryan

March 12, 2023

Contents

Chapter 1

First order logic

1.1 Syntax - The formal language

There are 3 main ingredients which make up the syntax

1. Propositional Atoms
2. Boolean Connectives
3. Punctuation

Propositional atoms

These have an associated truth value
We normally use values such as p, p', r, r' etc

Boolean Connectives

These link together the propositional atoms
and : written as $\wedge$
not : written as $\neg$
or : written as $\vee$
if-then / implies : written as $\implies$
iff : written as $\iff$
Then there is also truth and falsity (also known as bottom) which are written as $\top$ and $\bot$

Punctuation

This is the brackets, which are a pain
The main thing to remember is:

- if we have $\neg p$ this is written as $(\neg p)$, notice how the brackets are not after then not, bit of a pain

The rest of the things we will come on to in a few sections, but this was important

1.1.1 Propositional Formulas

There are 4 rules for a string of symbols to be a propositional formula.
ANYTHING ELSE IS NOT ONE!

- Any propositional atom is a formula eg (p, r, q)
- both $\top$ and $\bot$ are formulas
- if ϕ [1] is a formula so is $(\neg\phi)$ [2]
- if ϕ and ψ are formulas so are things like $(\phi \wedge \psi)$

1.1.2 Abbreviations

Take the formula

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

This is a pain to write and a pain to read
BUT THIS IS A ACTUAL FORMULA!
When we omit the brackets this is then a abbreviation, it is no longer the genuine formula.

1.1.3 Binding Conventions

The binding conventions go as follows from strongest to weakest

$$\neg \wedge \vee \implies \iff$$

Also remember we are left associative so we read from the left if we have the same binding conventions.

[1] This called a meta-variable, they are used to represent formulas
[2] Look at the brackets very important

1.1.4 Drawing formation trees

Say we had the formula [3]

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

This is the formation tree for this formula notice how the connectives on the top are the ones of the least binding conventions [4].

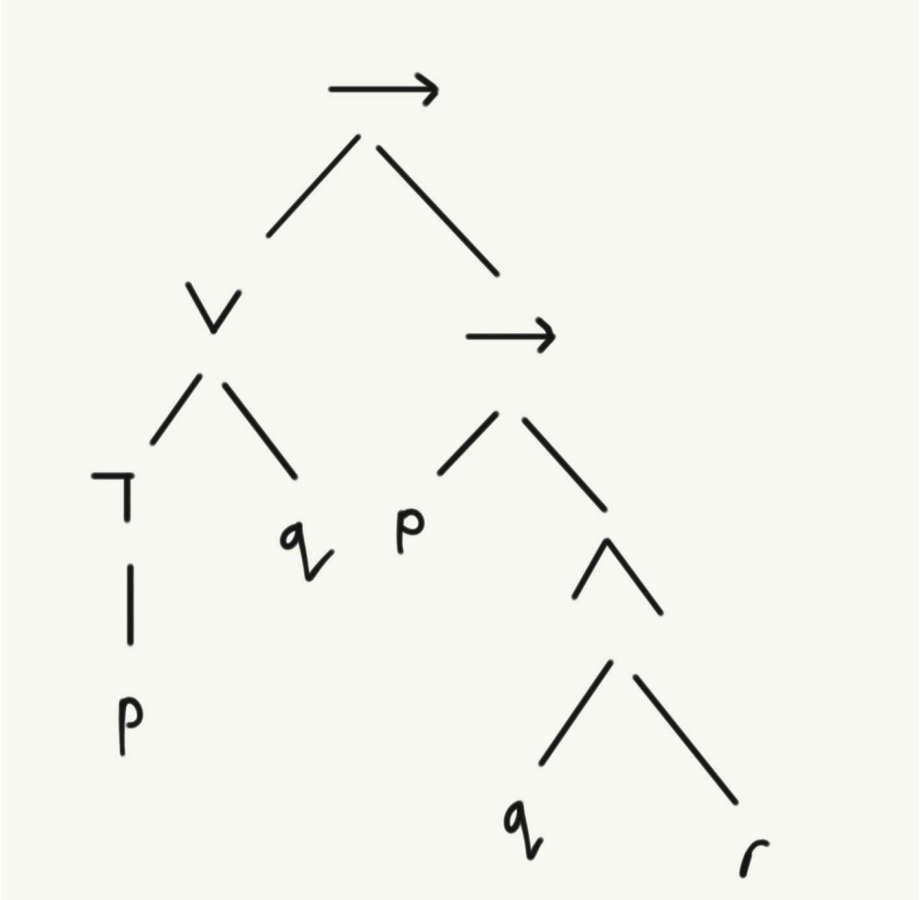


Figure 1.1: A tree.

[3] Notice how I did not add the $q \wedge r$ in brackets, that's cos of binding conventions on the last page

[4] This is called the principle connective

1.1.5 Overall Logical Form

Say we have a formula

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

This is a bit of a pain to look at
So we can then split this up into:

$$\neg p \vee q$$

$$p \implies q \wedge r$$

And we know that the principle connective [5] is

$$\implies$$

So now we can set:

$$\phi = \neg p \vee q$$

$$\psi = p \implies q \wedge r$$

and then:

$$\phi \implies \psi$$

This is the OVERALL logical form [6]

BTW - FOR SOME GOD FORSAKEN REASON WE CAN HAVE MULTIPLE LOGICAL FORMS:

So say we have the formula :

$$\neg(p \implies \neg q)$$

This can have the logical forms

- $\neg\phi$
- $\neg(\phi \implies \psi)$
- ϕ

So if you are asked for the overall logical form this is determined by the principle connective

The other versions are just the logical forms

[5] This is the weakest one
[6] This is determined by the principle connective

1.1.6 Sub formulas

Say we had the formula:

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

and we wanted to get out all of the sub formulas

One way of doing it is to draw out the tree and then to write ever element of the tree out [7]

Instead the better way to do this is:

Start by splitting the formula up via the principle connective:

$$\neg p \vee q$$

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

Oh look you now have 3 sub formulas (the original and the two you just made)

Now splitting those up further we get

$$\neg p$$

$$q$$

$$p$$

$$q \wedge r$$

And finally splitting the ones up which are not yet in the their simplest form

$$p$$

$$q$$

$$r$$

Notice how we have repeats, this is fine.

[7] But this is long and in exams not worth it

1.1.7 Special words we need to know

Atomic - Any formula of the sort p, $\top$ or $\bot$ is atomic
Negated formula - Any formula of the sort $\neg\phi$
Negated atomic formula - Any formula of the sort $\neg p$, $\neg\top$ or $\neg\bot$
Conjunction - Any formula of the sort $\phi \wedge \psi$ is a conjunction
Disjunction - Any formula of the sort $\phi \vee \psi$ is a conjunction
Implication - Any formula of the sort $\phi \implies \psi$ is a implication. In these formulas the ϕ is the antecedent and the ψ is the consequent
Bidirectional Implication - Any formula of the sort $\phi \iff \psi$ is a bidirectional implication [8]

1.1.8 Literals And Clauses

This is stupid BTW:

Literal - a formula which is atomic or negated atomic is called a literal
That makes sense right?
Now we have the definition of a clause
Clause - this is the disjunction of ONE OR MORE literals

This means that say we had the atomic formula p this is BOTH a literal and a clause, cos "by definition" that is what holds. [9]

[8] Obviously it is a implication which goes both way therefore "bidirectional"

[9] Some people also allow the empty clause cos that is the disjunction of zero literals which they take to mean that same as $\bot$

1.2 The Semantics

1.2.1 Situations + Evaluation Formulas

A situation - this is a something which will determine the truth-values of each propositional atom.

The truth values can either be (tt) or (ff) or 1 or 0

NOTICE HOW T AND F or t or f ARE NOT THERE - THERE'S IS A REASON FOR THAT, BUT WE DON'T CARE ABOUT THAT, JUST USE THE CONVENTIONS THEY HAVE GIVEN US.

The way we work out the truth-value of the propositional atom is with the use of a atomic evaluation formula.

$$v : A \to \{tt, ff\}$$

The formula above will take in a input from the A domain and will then return the truth value of that.

1.2.2 Evaluation Functions

This is a massive pain BUT it is also really intuitive and important

REMEMBER:
When your doing questions like this make sure to wrap everything your doing in:

$$|...|_v$$

Because logic is very specific and it does make sense, its just were evaluating ... with respect to the function v

$$\text{If } \phi \text{ is an atom then } |p|_v = \text{tt iff v(p)} = \text{tt}$$

This is obviously true, p applied to the function v is true iff when you apply p to the function you get true

MAKE SURE YOU LEARN THIS SHIT!!!!!!

$$|\neg\phi|_v = \text{tt iff } |\phi|_v = \text{ff}$$
$$|\phi \wedge \psi|_v = \text{tt iff } |\phi|_v = \text{tt and } |\psi|_v = \text{tt}$$
$$|\phi \vee \psi|_v = \text{tt iff } |\phi|_v = \text{tt or } |\psi|_v = \text{tt}$$
$$|\phi \implies \psi|_v = \text{tt iff } |\phi|_v = \text{ff or } |\psi|_v = \text{tt}$$
$$|\phi \iff \psi|_v = \text{tt iff } |\phi|_v = |\psi|_v$$
$$|\top|_v = \text{tt}$$
$$|\bot|_v = \text{ff}$$

1.2.3 Truth Functional

The truth value of the whole formula is determined by the truth values of the connected sub formulas.

1.2.4 Constructions of n-ary connectives

So if we have say a 0-ary Boolean connective this means we can have a possible 2 possible functions we can make with it (unique)

So if we have say a 1-ary Boolean connective this means we can get 4 possible functions out of them.

In general there are

$$2^{2^n}$$

Different n-ary Boolean connectives

1.2.5 Functional Completeness

If we have C which is a set of Boolean connectives, then C is functionally complete if any connective of any arity can be defined only from the elements in C

For example:

$$\{\wedge, \neg\}$$

If functionally complete as we can make any connecitve of any arity out of these two (think back to A-Level)

1.3 Truth Tables

This is the single greatest thing in logic
Free marks and very easy to do
How it works is that you are given a formula, and you evaluate every possible combination of inputs and check the outputs

p	q	$p \wedge q$
tt	tt	tt
tt	ff	ff
ff	tt	ff
ff	ff	ff

The sheffer stroke:

p	q	$p \uparrow q$
tt	tt	ff
tt	ff	tt
ff	tt	tt
ff	ff	tt

1.4 English Correspondence

This sections a pain, so just learn the rules and really hope they show up in the exam:

First lets look at the things we can't do, We can't translate:

- Commands
- Questions
- Exclamations
- Invitations

So if you see that in exams, write you can't do it
For example:
"Who let the dogs out"
Notice how this is a exclamation therefore we can not translate it.

Learn this table:

But / Yet / Although / Through	And
Unless	Or
Strong Unless	Iff
If / Only if / It is necessary for / it is sufficient for	Implies

For some reason these people like to write

$$\neg p$$

As "it is not the case p"
So maybe we should also do the same

1.4.1 Modalities of English Correspondence

There are issues which come with the translation to English as well:

Time:
Say we had the sentence "I will be rich and famous"
This line will still hold if say I get rich when I'm 20, lose all my money by 30 then get famous at 40. Which is not what we want to say logically

Permission:
Say we had the sentence "You can have chicken or fish"
If you make this into say you can have chicken $\vee$ you can have fish, then there's an issue as you can then have both which is not good. Therefore the direct translation is an issue

Obligation:
Say we had the sentence "I must do topics or a foreign language"
If you make this into say I must do topics $\vee$ I must do a foreign language, then there's the issue as you can then only do one and the statement will hold which is not true

1.5 Arguments and Validity

1.5.1 Valid Arguments

In every case where the inputs are all true, the outputs must also be all true
So in symbols

$$\phi_1, \phi_2, \phi_3, ...\phi_n \vDash \psi$$

So for all of the value between 1 - n, if ϕ_x is true then the formula is true
So some examples:

$$\phi \vDash \phi$$

This is true as whenever ϕ is true the RHS is also true

$$\phi \wedge \psi \vDash \phi$$

This is valid, and we can do a basic proof why:

$|\phi \wedge \psi|_v = tt$ iff $|\phi|_v = tt$ and $|\psi|_v = tt$

Therefore ever situation in which the LHS is true the RHS is also true as

$$|\phi|_v = tt$$

Therefore the argument is valid by the semantics of $\wedge$

Another two important examples are modus ponens and modus tollens:

$$\phi, \phi \implies \psi \vDash \psi$$

$$\phi \implies \psi, \neg\psi \vDash \neg\phi$$

1.5.2 Valid, Satisfiable, Contradiction and Equivalent

Valid - A formula is valid if it is true in every situation (these are also called tautologies)

Satisfiable - A formula is satisfiable if it is true in at least one situation

Contradiction - A formula is a contradiction if it is true in at false in all situations

Equivalent - ϕ and ψ are equivalent if they are true in exactly the same situations

YOU MAY GET A COURSEWORK OR SOMETHING AND YOU MAY WANT TO WRITE DOWN IF A FORMULA IS VALID THEN IT IS SATISFIABLE AS TECHNICALLY IT FULFILS THE DEFINITIONS, BUT DON'T DO THAT, IN THE NOTES IT LITERALLY SAYS IF YOU HAVE VALID U GET THE OTHER FOR FREE

1.6 Corresponding implication formula

So if we have something in the form

$$\phi_1, \phi_2, \phi_3, ...\phi_n \vDash \psi$$

then the corresponding implication formula is

$$\phi_1 \wedge \phi_2 \wedge \phi_3 \wedge ... \wedge \phi_n \implies \psi$$

From this we can now write the relationship between arguments and formulas

$$\phi_1, \phi_2, \phi_3, ...\phi_n \vDash \psi$$

Is a valid argument if and only if

$$\phi_1 \wedge \phi_2 \wedge \phi_3 \wedge ... \wedge \phi_n \implies \psi$$

Is a valid formula

1.7 Checking arguments are prepositionally valid

THIS NEXT SECTIONS A PAIN, BUT SO IS ALL OF LOGIC SO ITS OK:

There are 4 ways we will check an argument is valid:

- Truth Tables
- Direct argument
- Equivalences
- Various proof systems

1.7.1 Using truth tables to check validity

This should be common sense from the definition of valid.

Say we had the question

$$p \implies \neg q, q \vDash \neg p$$

So if we want to show this is valid we need to look at every situation where

$$p \implies \neg q \wedge q$$

IS true and then check if

$$\neg p$$

is true

p	q	$\neg q$	$p \implies \neg q$	$(p \implies \neg q) \wedge q$	$\neg p$
tt	tt	ff	ff	ff	ff
tt	ff	tt	tt	ff	ff
ff	tt	ff	tt	tt	tt
ff	ff	tt	tt	ff	tt

So this formula is valid as in every situation the LHS is true so is the RHS

A continuation from the question above is if we just compute the truth table for

$$p \implies \neg q \wedge q \implies \neg p$$

And it should show up true in all situations

In the same idea if we have

$$p \implies \neg q \wedge q \iff \neg p$$

The if we compute the truth table for that formula then it should come up to be true in all situations.

This tells us that the formula is valid.

1.7.2 Using direct argument

To do these types of proofs we have to construct arguments by taking arbitrary situations and then arguing to get to a conclusion.

When doing these kinds of questions remember what we want to prove, in every situation where the LHS is true, the RHS is also true.

So consider the formula:

$$p \implies (p \vee q)$$

So to prove this by direct argument we can do:

Take an arbitrary situation
To be a valid formula we need if p is true then $p \vee q$ is true
By the semantics of $\vee$ we know that if p is true then $p \vee q$ is also true
So we are done

Now looking at a harder one:

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

Take an arbitrary situation. To be a valid formula it must be true in this situation.

For this formula to be true either $p \wedge (p \implies q)$ = false or q = true (By the semantics of $\implies$)

For $p \wedge (p \implies q)$ to be true we must have p = true and $(p \implies q)$ = true (By the semantics of $\wedge$)

For $(p \implies q)$ = true we must have p = false or p = true and q = true

If p is false then $p \wedge (p \implies q)$ is false (by the semantics of $\wedge$)

and therefore $p \wedge (p \implies q) \implies q$ by the semantics of $\implies$

But if p is true, then for $(p \implies q)$ to be true q must also be true

Since in any situation where $p \wedge (p \implies q)$ is true q is also true then $p \wedge (p \implies q) \implies q$ is true in that situation

Now looking at a weird example
Say we wanted to show

$$(\phi \wedge \psi) \wedge \delta \equiv \phi \wedge (\psi \wedge \delta)$$

So take an arbitrary situation
For $(\phi \wedge \psi) \wedge \delta$ to be true we have
$(\phi \wedge \psi) = \text{tt} \wedge \delta = \text{tt}$ (By the semantics of $\wedge$)

For $(\phi \wedge \psi)$ to be true we need to have
$\phi = \text{tt} \wedge \psi = \text{tt}$ (By the semantics of $\wedge$)

Therefore for $(\phi \wedge \psi) \wedge \delta$ to be true we must have
$\phi = tt$ and $\psi = tt$ and $\delta = tt$

Here's the weird bit. We can now build from this answer to get the RHS:
$\phi = tt$ and $\psi = tt$ and $\delta = tt$

Holds if and only if
$\phi = tt$ and $(\psi = tt \wedge \delta = tt)$

Which holds if and only if
$\phi \wedge (\psi \wedge \delta)$

Which is what we wanted to get on the RHS so we are done.

1.7.3 Using equivalences to check validity

The way we do this is, take the implication formula and then simplify it to $\top$
Or just look at what your are given what you want to work to get to and then apply equivalences until you get there.

The 27 equivalences

The ones with and:

$\phi \wedge \psi$ is logically equivalent to $\psi \wedge \phi$ (Commutativity of $\wedge$)

$\phi \wedge \phi$ is logically equivalent to ϕ (idempotence of $\wedge$)

$\phi \wedge \top$ and $\top \wedge \phi$ is logically equivalent to ϕ

$\phi \wedge \bot$, $\bot \wedge \phi$, $\phi \wedge \neg\phi$ and $\neg\phi \wedge \phi$ are logically equivalent to $\bot$

$(\phi \wedge \psi) \wedge \delta$ is logically equivalent to $\phi \wedge (\psi \wedge \delta)$

The ones with or

$\phi \vee \psi$ is logically equivalent to $\psi \vee \phi$ (Commutativity of $\vee$)

$\phi \vee \phi$ is logically equivalent to ϕ (idempotence of $\vee$)

$\phi \vee \top$, $\top \vee \phi$, $\phi \vee \neg\phi$ and $\neg\phi \vee \phi$ are logically equivalent to $\top$

$\phi \vee \bot$, $\bot \vee \phi$ is logically equivalent to ϕ

$(\phi \vee \psi) \vee \delta$ is logically equivalent to $\phi \vee (\psi \vee \delta)$

The other ones

$\neg\top$ is equivalent to $\bot$

$\neg\bot$ is equivalent to $\top$

$\neg\neg\phi$ is equivalent to ϕ

$\phi \implies \phi$ is equivalent to $\top$

$\top \implies \phi$ is equivalent to ϕ

$\phi \implies \top$ is equivalent to $\top$

$\bot \implies \phi$ is equivalent to $\top$

$\phi \implies \bot$ is equivalent to $\neg\phi$

$\phi \implies \psi$ is equivalent to $\neg\phi \vee \psi$

$\neg(\phi \implies \psi)$ is equivalent to $\phi \wedge \neg\psi$

$\phi \iff \psi$ is equivalent to

$((\phi \implies \psi) \wedge (\psi \implies \phi))or(\neg\phi \iff \neg\psi)or((\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi))$

$\neg(\phi \iff \psi)$ is equivalent to

$(\phi \implies \neg\psi)or(\neg\phi \iff \psi)or((\phi \wedge \neg\psi) \vee (\neg\phi \wedge \psi))$

De-Morgans Laws:

$\neg(\phi \wedge \psi)$ is equivalent to $\neg\phi \vee \neg\psi$

$\neg(\phi \vee \psi)$ is equivalent to $\neg\phi \wedge \neg\psi$

Distributively of $\wedge$ and $\vee$

$\phi \wedge (\psi \vee \delta)$ is equivalent to $(\phi \wedge \psi) \vee (\phi \wedge \delta))$

$(\phi \wedge \psi) \vee \delta$ is equivalent to $(\phi \vee \delta) \wedge (\psi \vee \delta))$

Absorption:

$\phi \wedge (\phi \vee \psi)$ is equivalent to ϕ

1.7.4 Normal forms

We can use the skills learnt in the last few pages to re-write formulas into disjunctive and conjunctive normal form.

Disjunctive normal form

This holds when the formula is a disjunction of conjunctions of literals and we can not further simplify without leaving this form [10]

A formula in DNF is unsatisfiable, if and only if each of its conjunctions contains some literal and its negation.

Conjunctive normal form

This holds when we have a conjunction of disjunctions of literals and we can not further simplify without leaving this form.

A formula in CNF is valid, if and only if each of its disjunctions contains some literal and its negation.

Putting a formula into normal form:

1. Remove all of the occurrences of $\implies$ and $\iff$ by
- replacing all sub formulas $\phi \implies \psi$ by $\neg\phi \vee \psi$
- replacing all sub formulas $\phi \iff \psi$ by $(\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi)$

2. Use the De Morgan laws to push the negations down next to atoms. Delete all double negations (replace $\neg\neg\phi by \phi$).
3. Rearrange using distributively to get the desired normal from.
4. Simplify:
- replacing subformulas $p \wedge \neg p$ by $\bot$, and $p \wedge \neg p$ by $\top$
- replacing subformulas $\top \vee p$ by $\top$, $\top \wedge p$ by p , $\bot \vee p$ by p and $\bot \wedge p$ by $\bot$.
- absorption is often useful too. - repeat till no further progress.

Remember when doing equivalences

- Remember to write what you are using
- U may put consecutive applications of the same law into one step
- Reference the law next to the RESULT
- make sure to use the correct version of the laws
- remember to quote associativity and commutativity

[10] What this means is that you may be able to make the formula simpler but if in doing so you leave disjunctive normal form, then don't do it.

DNF from truth tables

To write a formula in DNF from a truth tables:
We first draw out the truth table for the formula
We then look at all of the points where it evaluates to tt
And we take the disjunction of the conjunction of these formulas

CNF from truth tables

To write a formula in CNF from a truth tables:
We first draw out the truth table for the formula
We then look at all of the points where it evaluates to ff
And we take the conjunction of the disjunctions of the negation of all of the atoms in these formulas

1.7.5 Natural deduction

So think of natural deduction as a game:

We have some givens and a goal and we need to get from the given to the goals.
Once we have done this we can write

$$\phi_1, \phi_2, \phi_3, ...\phi_n \vDash \psi$$

Where
$\phi_1, \phi_2, \phi_3, ...\phi_n$ - is called the premise
ψ - is called the conclusion
$\phi_1, \phi_2, \phi_3, ...\phi_n \vDash \psi$ - is called the sequent

BOXES ARE A PAIN, BUT THEY ARE USEFUL, SO THEY GET A PASS

How they work is you open a box when you want to assume and once you close that box you are no longer allowed to use that assumption.

One important thing about natural deduction is you have to remember each step of the proof must be a valid argument.

$\wedge$ introduction

If we have
1. ϕ
2. ψ
3. $\phi \wedge \psi \wedge I(1,2)$

This is and introduction if we have two things proven, we can take the conjunction of both of them

$\wedge$ elimination

If we have
1. $\phi \wedge \psi$
3. $\psi \wedge E(1)$

If we have a and formula, we can use and elimination of get either the LHS or the RHS

$\vee$ introduction

Say we have ϕ proven
We can just do $\phi \vee \psi \vee I(Linenumber)$
And this can be done for any formula ψ

$\vee$ elimination

Say we have the formula $\phi \vee \psi$

And we want to say get some formula ρ

The we have the ability to argue by cases:

ϕ assume	ψ assume
.	.
.	.
.	.
ρ	ρ

$\rho \vee E(line1, line2)$

BTW REMEMBER TO ADD IN THE LINE NUMBERS, LATEX IS ANNOYING SO I HAVE NOT

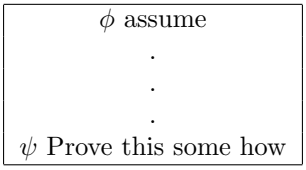
$\rightarrow$ elimination

If we have say $\phi \rightarrow \psi$
And we have also prove ϕ
Then we can do $\rightarrow$ elimination to get ψ

(This may seems familiar, this is cos it is literally modus pones.)

$\rightarrow$ introduction

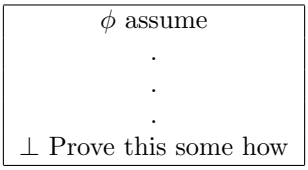
Say we want to have $\phi \rightarrow \psi$
What we do is we make a box and we assume



$\phi \to \psi \to$ I(line number)

$\neg$ introduction

Say we want to have $\neg\phi$
What we can do is assume ϕ and prove that is bullshit:



$\neg\phi\neg$ I(line number)

$\neg$ elimination

Say we have $\neg\phi$ and ϕ proven then we have get $\perp$

Double negation introduction

Say we have ϕ we can get to $\neg\neg\phi$

Double negation elimination

Say we have $\neg\neg\phi$ we can get to ϕ

Deriving anything from false

If we have shown false, then we can get anything, as we can derive anything from false.

$\perp$ introduction

Say we have ϕ and we have $\not\phi$ we can get to $\perp$

$\perp$ introduction

Say we have ϕ and we have $\not\phi$ we can get to $\perp$

$\perp$ elimination

Say we have $\perp$ we can get to ϕ
as we can assume anything from $\perp$

Law of excluded middles

At any given time in a proof you have the ability to just write
$\not\phi \vee \phi$
Which technically means $\perp$
This means that if we have ϕ
then we can get to $\not\phi$ by elimination

Deductions with lemmas

This is a proof which helps you prove what you actually what to prove.

Natural deduction proofs

Say we have $\phi_1, ..., \phi_n \vdash \psi$

Then this means we have a natural deduction proof which starts with $\phi_1, ..., \phi_n$ and proves ψ

Difference between $\vdash$ and $\vDash$

$\vDash$ - this is syntactic and involves proofs

$\vDash$ - this is semantic and involves situations

1.7.6 Soundness, Completeness And Other Final Points

Theorem

A meta-variable ψ is said to be a theorem if it can be proven with no premise

Soundness

A proof system is sound if every theorem is valid

So basically:
If

$$\phi_1, \phi_2, \phi_3, ...\phi_n \vdash \psi$$

then

$$\phi_1, \phi_2, \phi_3, ...\phi_n \vDash \psi$$

Completeness

A proof system is complete if every valid formula is a theorem
If

$$\phi_1, \phi_2, \phi_3, ...\phi_n \vDash \psi$$

then

$$\phi_1, \phi_2, \phi_3, ...\phi_n \vdash \psi$$

Consistency

A system is said to be consistent if $\nvdash \neg\phi$

So in words this means that if the negation of the meta variable is not valid

Now we can extent this to say

A formula is consistent iff it is satisfiable

Provable and Semantic Equivalence

Two formulas are provably equivalent if and only if

$\phi \vdash \psi$ and $\psi \vdash \phi$

And they are semantically equivalent if they are true in the same situations.

Therefore we now have the theorem:

Two formulas are provably equivalent iff they are semantically equivalent

Chapter 2

First-Order Predicate Logic

You though the first bit was a bitch, oh get ready.

2.0.1 Syntax - the formal language

Predicate logic is the one which is more expressive, so expressive in fact we now have 6 more features:

- Predicates (Things which take arguments)

  Predicate Symbols - these describe the properties and relations between objects: Student: Takes in 1 argument : unary Friend : Takes in 2 arguments : binary

- Constants (Things like Bob etc)

  Constants - these give names to objects For example things like Arron and Russell

- Variables (x, y, z)

  Variables - these can be used to represent elements in a set when we don't know exactly what the values are;

  For example ∀x student(x) literally means that 'For all x, x is a student'

- Quantifiers (∀ and ∃) Quantifiers - There's two new ones we care about
  ∀ - for all
  ∃ - there exists

- Functions

  Functions - A function symbol will refer to an object in terms of other objects (WHICH IS JUST THE SAME AS MATHS, A FUNCTION TAKES IN SOMETHING AND RETURNS SOMETHING ELSE)

- Equality

  Equality - This is used to check for the same object, so if we have x = y, this means that x and y are the same object

Signatures

I'm gonna try and write this nicer then the PowerPoint's, cos the one there's a bit confusing so:

A signature is a triple which is made up of ¡K, F, P¿

- K
  This is a (possibly empty) set of constants
- F
  This is a set of possibly empty sets of function symbols
- P
  This is a set of predicate symbols.

So to give this a bit of explanation this is like a language which we can use. We are now given a set of "things" and we can do things with those things.

For example say we could have a L signature which has[1]

- constants eg Frank, Susan, Tony etc
- unary predicate symbols eg PC, human
- binary predicate symbols eg bought

THE THINGS ABOVE ARE JUST SYMBOLS - SYNTAX, THEY THEMSELVES DON'T HAVE ANY MEANING! TO GIVE THEM MEANING WE NEED TO WORK OUT WHAT A SITUATION SHOULD BE.

Terms

To write formulas we need terms to name objects: [2]

So for a given signature L:

- Any constant in L is a L-Term
- Any variable is a L-term
- If f is an n-ary function symbol in L and $t_1 ... t_n$ are L-terms then $f(t_1, ..., t_n)$ is a L-term.

A closed term is one which does not involve a variable

[1] These are the things we call signatures

[2] These are not formulas they will not be true or false

Formulas

Say we have been given a signature L

- If R is an n-ary predicate symbol in L and t1 ... tn are L-terms then R(t1, t2 ... tn) is a atomic L-Formula
- if we have t and t' which are L-terms then t = t' is a L-formula
- $\top$ and $\bot$ are L-Formulas
- if ϕ and ψ are L-Formulas then any boolean connective added to them will also be a L-Formula (make sure to remember brackets)
- if ϕ is a L-Formula so and x is a variable then $(\forall x\phi)$ and $(\exists x\phi)$ are also L-formulas

Atomic Formulas, Literals and Sentences

An atomic formula is a predicate symbol which has been filled in with terms

A literal is a atomic formula or it's negation

A sentence is a formula with all variables in the scope of a quantifier:

$$\forall x \forall y bought(x, y)$$

2.0.2 Semantics - the meaning

To give a situation we now need to write

- what the situation is
- how to evaluate the predicate logic formulas for any given situation

So to do this shit, we first need to give meaning to all of the symbols in a given signature. Then we need to define the notation of valuation of terms and evaluation of atomic formulas and quantifiers.

↑ I have just typed this out and I have no clue what it means, so this is my attempt to understand it:

Ohhh, what this means is that we now need to come up with some way to be able to make a situation. So we have to traverse through the concept of a signature, and at each stage give more meaning to what we have. Until we have a way to be able to evaluate things like the atomic formulas etc.

Functions and relations over the domain of discourse

Basically the domain of discourse ($\mathbb{D}$) is a none empty set of objects.

For a positive whole number n, the n-ary Cartesian power set $\mathbb{D}$ written $\mathbb{D}^n$ is the set of all n-tuples which can be made from its members.

Ok and then a relation of arity n is a subset of $\mathbb{D}^n$

AND THAT MAKES SENSE, COS THE RELATION WILL TAKE IN A SUB SET OF ALL OF THE POSSIBLE COMBINATIONS

And finally a many to one function is one which will send diffrent n-tuples to the same object in $\mathbb{D}^n$

L-Structure

Say we have a signature (K, F, P). An L-Structure, M is a pair M = ($\mathbb{D}$, $\mathbb{I}$) where

$\mathbb{D}$ is the domain of discourse, this is all the shit M knows exists and can use.

$\mathbb{I}$ is the interpretation, this takes all of the symbols in L and gives them meaning in terms of $\mathbb{D}$.
So for example

- for each constant c in K $\mathbb{I}(c) = c_m \in \mathbb{D}$

- for each n ary function symbol f in F $\mathbb{I}(f) = f_m : \mathbb{D}^n \to \mathbb{D}$
- for each n ary predicate symbol f in F $\mathbb{I}(P) = P_m \subseteq \mathbb{D}^n$

Now using all of this new knowledge we have the ability to be able to draw the L-Structure as a diagram:

To do this we have to do

- draw the collection of objects
- naming all of the named constants
- denoting everything which fulfils the unary predicate symbols
- drawing the arrows between the objects which satisfy the binary predicate symbols

In these diagrams if we wanna say something is true we do:

$$M \vDash function(argument)$$

Which means that M says that function(argument)

In a similar way

$$M \nvDash function(argument)$$

Means that it is false

2.1 Free and Bound Variables + Trees

Bound Variables - these are ones which lie in the scope of their quantifiers
Free Variable - a variable is free if it is not bound

For example consider the formula

$$\forall x(R(x,y) \wedge R(y,z) \rightarrow \exists z(S(x,z) \wedge z = y))$$

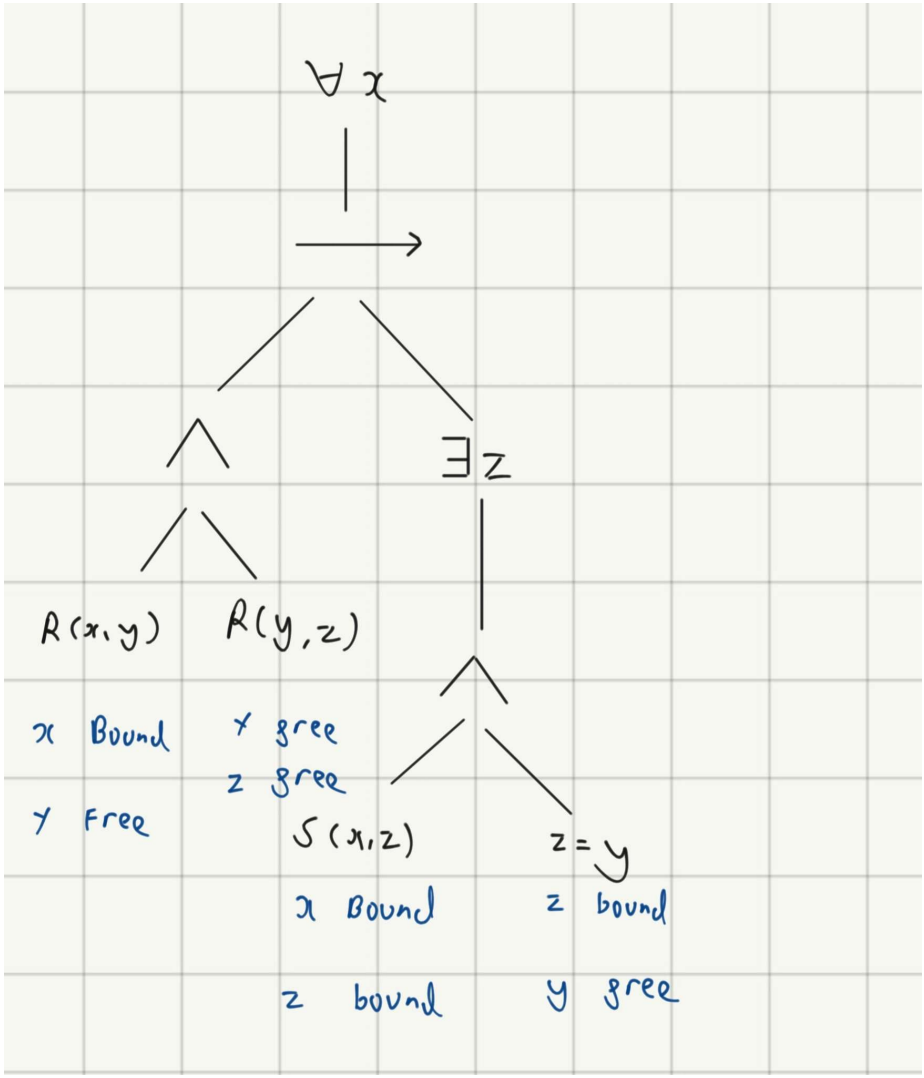


Figure 2.1: A tree.

So there are some weird constructions when it comes to building up the trees.
First of all we do not separate out the quantifiers
Also when we have a equal this is a predicate so we do not split it up further

(same for the R(x,y)s)

And for all and there exists have the same strength as not.

POINT - When we have a situation, we need to handle the values of the free variables before getting a true or false value. Even if we have things like x = x, where this will always be true, we have to assign what x is before evaluating.

Assigning Variables

Say we have a structure M. The a assignment into M is a function which takes objects in the dom(M) and assigns them to each variable.

$$h : V \to dom(M)$$

if an assignment and V is a set of variables.
For an assignment h and a variable x, we write h(x) to denote the object in dom(M) assigned to x by h.

Evaluating terms

We have a complete situation if we have an L-structure M plus and assignment into M. [3]

After we have this we can:

- Any L-term into an object dom(M)
- Any L-Formula to be true or false

Actually evaluating L-Formulas

So to be able to evaluate a term what we have to do is:

Say we have L as a signature, and then M = $(\mathbb{D}, \mathbb{I})$, Is a L-structure and then h is a assignment into M.
Then we know there are 3 types of terms and this is how to evaluate all of them:

- if we have a constant t : $v_m^h(t) = \mathbb{I}(\text{t})$
- if we have a variable t : $v_m^h(t) = \text{h(t)}$
- if we have a function t : $v_m^h(t) = f_m(v_m^h(t_1)), ...v_m^h(t_n)$

AND NOW THIS SHIT MAKES SENSE, IN A COMPLETE SYSTEM WE CAN FIGURE OUT WHAT EVERYTHING IS, AND NOW WITH THE APPLICATION STUFF WE CAN ACTUALLY DO THAT.

[3] Before we go forwards, I wanna lokey try and understand literally everything on this page. So an assignment into M is a function. That makes sense. This function takes each variable and assigns some object which is in the domain to that variable. Fucking obviously now that I've written it down, it is literally called an assignment and what it does is it assigns variables shit in the domain which means we can now actually use all of it, as before remeber when it said we can't do certain things if we have a varibale.

Evaluating formulas

The next part only works if we don't have quantifiers!

If we have a structure M and a assignment h, we can write

$$M, h \vDash A$$

If it is true under M and

$$M, h \nvDash A$$

If it is false

The next bit is a lot of symbols:

If we have R as a predicate symbol in L, and $t_1, t_2, ..., t_n$ be L-terms. The if we have $v_m^h(t_i) = a_i$ for all of i = 1, .., n. Then:
$M, h \vDash R(t_1, t_2, ..., t_n)$ if $(a_1, a_2, ..., a_n) \in R_m$ and false if not.

SO IN WORDS THIS MEANS THAT R IS TRUE UNDER M, IF WHEN APPLY H TO ALL OF THE TERMS WE GET BACK SOMETHING WHICH IS IN R_m

If we have t, t' as terms then:

$M, h \vDash t = t'$ holds if $v_m^h(t) = v_m^h(t')$

This means that if t and t' get mapped to the same value in the domain then they hold under M, else they do not.

$$M, h \vDash \top and M, h \nvDash \bot$$

This means that $\top$ will always hold under M, and $\bot$ will not hold.

$M, h \vDash A \wedge B$ holds if $M, h \vDash A$ and $M, h \vDash B$

Which again makes sense, $A \wedge B$ holds if both A and B hold.

And all other Boolean connectives follow the same overall structure.

Now its more interesting:
We have looked at when it is unbounded. Now when its bounded by the quantifiers we know if we have :

$\forall$ then we have to be true in all situations $\exists$ then we have to be true in some situations

[Variable]-equivalent

Two variable assignments are [Variable]-equivalent if they differ at, at most one assignment at the variable [Variable].

So say we have a structure M and we have two assignment's g and h, we say they are x equivalent

$$g =_x h$$

If they differ at most at the assignment of x.

NOTICE HOW IT SAYS AT MOST, THIS MEANS THAT IT CAN BE NONE AS WELL! SO THIS MEANS THAT A ASSIGNMENT IS ALWAYS [VARIABLE]-EQUIVALENT TO ITSELF.

The semantics of quantified formulas

Say we have M being a L-Structure and h be any assignment into M.

If we know how to evaluate ϕ in M under any assignment, say we have variable x. Then:

$M, h \vDash \exists x\phi$, will work if there is some g into M such that $M, g \vDash \phi$ when g $=_x$ h. Else we say $M, h \nvDash \exists x\phi$

Before moving onto the next thing, I wanna try and write out what this means in words

So we are saying that $\exists x\phi$ hold in M, if there is some x equivalent function g to h such that $M, g \vDash \phi$ holds. Which means that the exists statement holds if there is some other x equivalent assignment such that that version does hold.

$M, h \vDash \forall x\phi$, will work if for every g into M such that $M, g \vDash \phi$ when g $=_x$ h. Else we say $M, h \nvDash \exists x\phi$

Ohhhhhh and now it makes sense. The last one was that we needed at least one other assignment such that it holds for the exists statement to hold. For the for all we need all of the other assignment to hold for the statement to hold.

So now look at the diagram on slide 16 cos i cba to draw it out. Say we had the question

$$Q, h_4 \vDash \forall x \exists y, bought(x, y)$$

we have a $\forall$ and then a $\exists$ which is nested in it.

This means from the diagram on slide 16, we will need to look at all of the x's which are [x]-equivalent, and then all of the y's which are [y]-equivalent.

And then find orderings which work for all of them.

Given a formula ϕ, whether or not $M, h \vDash \phi$ depends of $\phi(x)$ for the variables which life free in ϕ

OH SHIT, IT MAKES SENSE, So what you do is say we have a formula

$$\phi(x_1, x_2, ..., x_n)$$

and we also have

$$\phi(h(x_1) - a_1, h(x_2) = a_2, ..., h(x_n)) = a_n$$

Wherever we have something which is free, we can replace it for example this becomes:

$$M \vDash \phi(a_1, a_2, ..., a_n)$$

Notice on the RHS we no longer have $M, h \vDash$
Cos we have replaced every instance in which we could have had h with it's value.

In the same way if we have

$$M, h \vDash S$$

and this does not depend on h then we can write $M \vDash S$

REMEMBER THIS ONLY WORKS ON THE FREE VARIABLES

Sentences

A sentence is a formula with no free variables.

WEIRD THING WHICH COULD SHOW UP IN AN EXAM:

$$\forall x(bought(Tony, x)) \rightarrow PC(x)$$

is a sentence

BUT IT'S SUB FORMULA ARE NOT, Because say:

$$bought(Tony, x)$$

Is not a sentence as it is not bounded.

2.2 English translation

There are rought patterns which we should learn to spot, things like

$$\forall(A \to B)$$

Translates to every A is a B

IN THIS SECTION USE COMMON SENSE. NORMAL SPEAKING ENGLISH PEOPLE DO NOT USE VARIABLES IN SPEECH, NEITHER SHOULD YOU!

So say you have variables which you can not get around omitting, try replacing the variable with the word someone.

Also this is quite weird, say we have $\forall x$ then we do not say things like anyone (if we are keeping it general), we try and stick to anything to generalize it as much as possible.

The patterns to look out for in English translation are:

$\forall x(A \to B)$ and $\exists x(A \wedge B)$ are very common.

If we have $\exists x(A \to B)$ that is very rare, if we get it check if we have everything right.

Chapter 3

Many-sorted logic

In logic we have things called sorts. These are things of diffrent types.

We can fix a collection s, s', s" or sorts. We can then not use these to be able to make new sorts.

If we want to add more elements to the sorts, we need to explicitly add them to the list of or original sorts (BTW we need to have a list original sorts.)

Each variable and constant has its own sort s. To say what sort we write $x : s$ and $c : s$. In each sort there are infinitely many variables which will come into the sort.

Each n-ary function will come with a template:

$$f : (s_1, ..., s_n) \to s$$

Each of the $t_1, ..., t_n$ if t_i has sort s_i then $f : (s_1, ..., s_n)$ in terms of the sort s.

So basically each relation symbol R will come with a template $R(s_1, ..., s_n)$ where $s_1, ...ms_n$ are sorts.
So this means that if we have say
$R(t_1, t_2, ..., t_n)$ then term t_n must have sort s_n else it is pointless.

So say we have the formula t = t', then the terms t and t' have the same sort, else its stupid to write this.

3.1 L-Structures in many sorted logic

Say we have L as a many-sorted signature (same as a L signature but also with all of the sorts added)

We need to in the diagram explicitly state what each sort it by writing the word sort before it.

3.2 Lists in many sorted logic

This is quite intuitive but in the slides they made a big deal about them, so I'll add it in here.

What we have available to us is:

- +, -, *
- []
- (cons) :
- ++
- head
- tail
-
- !!

And we can use these to write things like say:

For all lists which aren't empty, the head is given by the element at position 0.

$$\forall xs(xs \neq [] \rightarrow head(xs) = xs!!0)$$

3.3 Arguments and validity

We know what a valid argument is:

Say we have a signatures L and then $A_1, ..., A_n$ and B as L-formulas, then $A_1, ..., A_n$ therefore B will be valid if M, h $\vDash A_1$, $M, h \vDash A_2$ ... $M, h \vDash A_n$ (which means that they are all true under M and h), then M, h $\vDash$ B. In that case we can write $A_1, ..., A_n \vDash B$

THIS MAKES SENSE, WHAT IT SAYS IS THAT UNDER M AND h, IF ALL OF THE SHIT ON THE LHS IS TRUE, THEN ALL OF THE SHIT ON

THE RHS IS ALSO TRUE, WHICH MAKES SENSE.

We say a L-formula is valid if for every M and h, $M, h \vDash A$ holds.

We say an L-formula is satisfiable if for some M $M, h \vDash A$

Now to check if two formulas A and B are equivalent we need to have

$$M, h \vDash A \iff M, h \vDash B$$

3.4 Ways of checking validity

There are 3 mains ways to check if a argument is valid:

- direct reasoning
- equivalences
- proof systems

3.4.1 Direct reasoning

I quite liked this one so I'll put is here:

If we want to show:
$\forall(human(x) \to lecturer(x)), \forall(PC(x) \to lecturuer(x), \forall(human(x) \vee PC(x))) \vDash \forall lecturer(x)$

This is actually quite cool, when proving shit is valid, start by taking an arbitrary a in x.

Now lets to case analysis on the third thing we were give

$\forall(human(x) \vee PC(x)))$

We know that either

$M \vDash human(a) or M \vDash PC(a)$

Either way, we can cancel out using 1 or 2, and we end up with lecturer(a)

3.4.2 Equivalences! Again!

We have all of the old ones, And:

$$\forall x \forall y A \equiv \forall y \forall x A$$
$$\exists x \exists y A \equiv \exists y \exists x A$$
$$\nexists x A \equiv \exists x \not A$$
$$\not\exists x A \equiv \forall x \not A$$
$$\forall x (A \wedge B) \equiv (\forall x A \wedge \forall x B)$$
$$\exists x (A \vee B) \equiv (\exists x A \vee \exists x B)$$

NOW REMEMBER FOR THE NEXT FEW THESE ONLY WORK IF THE VARIABLE WE ARE TAKING OUT IS FREE IN THE IN THE OTHER FORMULAS.

$\forall x A$ and $\exists x A$ are logically equivalent to $\exists x P(x)$

$$\exists x (A \wedge B) \equiv A \wedge \exists x B$$
$$\forall x (A \wedge B) \equiv A \wedge \forall x B$$
$$\forall x (A \to B) \equiv A \to \forall x B$$
$$\exists x (A \to B) \equiv A \to \exists x B$$
$$\forall x (A \to B) \equiv \exists x A \to B$$
$$\exists x (A \to B) \equiv \forall x A \to B$$

Renaming variables

Lets say we have formulas, then if we rename the variables in the formula then that formula is still equivalent to the original.

Leibnizs' principle

If A is a formula and there are no y's in it, and B is got from A by replacing on of more free occurrences of x by y then:

$$x = y \to (A \iff B)$$

is valid.

Which kinda makes sense. This says that x equals y (cos we replaced the x with the y), then A holds iff B holds which obviously is true. also if x does not equal y, then it does not necessarily have to be true, which also holds from this formula.

3.4.3 Natural deduction

$\exists$ introduction

If we have a formula A(t/x), this is the formula got from A by replacing all free occurrence of x in A by t.

So say we have A(t/x) we can write this as

$\exists x$ A

A(c/x) assume
the proof
B

B

$\forall$ introduction

To make this work we want to take some arbitrary value c, and then we want to prove that A(c/x) holds for all of those situations.

c $\forall$I const
the proof
A(c/x)

$\forall x A$

$\forall$ elimination

If we have $\forall$ x A then we can write A(t/x) for some arbitrary value

$\forall x A$
$A(t/x)$

And this makes sense cos if we for all of x, A holds, then we should be able to take some random x and it will still hold.

Rules for equality

There are two main rules
refl - this is reflexive so at any give time we can introduce t = t
=sym - this is symmetric, we have x = y we can write y = x
=sub - If we have say A(t/x), and t = u, then we can derive A(u/x)

3.5 Soundness and completeness

Soundness

If we have $A_1, ..., A_n, B$ by first order sentences.
If $A_1, A_2, ..., A_n \vdash B$ then $A_1, A_2, ..., A_n \vDash B$

Completeness

If we have $A_1, ..., A_n, B$ by first order sentences.
If $A_1, A_2, ..., A_n \vDash B$ then $A_1, A_2, ..., A_n \vdash B$