Honors Logic, Lecture 02

Brian Weatherson

2022-08-31

Seven Symbols

Classical Models

Entailment

Negation

- Read ¬ as “It is not the case that...”.

Negation

- Read $\neg$ as “It is not the case that...”.
- This is kind of weird in English; we usually but negations in the predicate not at sentence level.

Negation

- Read ¬ as "It is not the case that...".
- This is kind of weird in English; we usually but negations in the predicate not at sentence level.
- There was a version of English occasionally spoken in the 90s that used "Not" after a sentence as a sentential negation, but this was a passing fad, and never became standard.

Conjunction

- Read $\wedge$ as “and”.

Conjunction

- Read $\wedge$ as “and”.
- Again, this is a sentential connective.

Conjunction

- Read $\wedge$ as "and".
- Again, this is a sentential connective.
- English has this, but it is probably more common to use it between predicates.

Conjunction

- Read $\wedge$ as "and".
- Again, this is a sentential connective.
- English has this, but it is probably more common to use it between predicates.
- Note that we'll use the term 'conjunction' exclusively for 'and'; it grammar books it is any term that connects two sentences.

Disjunction

- Read $\vee$ as "or".

Disjunction

- Read $\vee$ as "or".
- Once again, it is a purely sentential connective.

Disjunction

- Read $\vee$ as "or".
- Once again, it is a purely sentential connective.
- It is **inclusive** disjunction; we don't have a dedicated symbol for exclusive disjunction, though we could define one.

Disjunction

- Read $\vee$ as “or”.
- Once again, it is a purely sentential connective.
- It is **inclusive** disjunction; we don’t have a dedicated symbol for exclusive disjunction, though we could define one.
- This is a stipulative definition, but I think it’s actually the right one for natural language disjunction. Though I’ll leave that argument for class, not slides.

Conditional

- For now, read $\supset$ as “if ... then”.

Conditional

- For now, read $\supset$ as "if ... then".
- Most linguists/logicians/philosophers think it is a really bad translation of English "if", though some think it is right.

Conditional

- For now, read $\supset$ as “if … then”.
- Most linguists/logicians/philosophers think it is a really bad translation of English “if”, though some think it is right.
- Priest thinks it is so bad we’ll use the symbol $\rightarrow$ as a better “if” later.

Conditional

- For now, read $\supset$ as "if ... then".
- Most linguists/logicians/philosophers think it is a really bad translation of English "if", though some think it is right.
- Priest thinks it is so bad we'll use the symbol $\rightarrow$ as a better "if" later.
- We are about to get to what $\supset$ stipulatively means.

Biconditional

- Again for now, read $\equiv$ as “if and only if”.

Biconditional

- Again for now, read $\equiv$ as “if and only if”.
- This is sometimes shortened in writing to “iff”. The pronunciations of this shortening are dire; better to say “if-and-only-if”.

Biconditional

- Again for now, read $\equiv$ as "if and only if".
- This is sometimes shortened in writing to "iff". The pronunciations of this shortening are dire; better to say "if-and-only-if".
- Again, we're going to give it a stipulative definition.

Semantic Implication

- Read $\vDash$ as "entails".

Semantic Implication

- Read $\models$ as "entails".
- That is $\Gamma \models A$ is to be read as Γ entails A.

Semantic Implication

- Read $\models$ as "entails".
- That is $\Gamma \models A$ is to be read as Γ entails A.
- And "entails" here means that whenever all the elements of Γ are true (in a model of the salient kind), A is true as well.

Semantic Implication

- Read $\vDash$ as "entails".
- That is $\Gamma \vDash A$ is to be read as Γ entails A.
- And "entails" here means that whenever all the elements of Γ are true (in a model of the salient kind), A is true as well.
- This is sometimes called "model-theoretic entailment".

Context sensitivity

- You will sometimes see one or other subscript on $\models$.

Context sensitivity

- You will sometimes see one or other subscript on $\vDash$.
- That's to indicate which kinds of models are in play.

Context sensitivity

- You will sometimes see one or other subscript on $\models$.
- That's to indicate which kinds of models are in play.
- When there is no subscript, just be a bit careful about which model we're using.

Proof

- Read $\vdash$ as “proves”.

Proof

- Read $\vdash$ as "proves".
- That is, $\Gamma \vdash A$ means that there is a proof of A given the premises in Γ.

Proof

- Read $\vdash$ as "proves".
- That is, $\Gamma \vdash A$ means that there is a proof of A given the premises in Γ.
- Just what a proof is becomes **really** context sensitive.

Proof

- Read $\vdash$ as "proves".
- That is, $\Gamma \vdash A$ means that there is a proof of A given the premises in Γ.
- Just what a proof is becomes **really** context sensitive.
- It turns both on what logic we're talking about, and what proof system for that logic we're talking about.

Priority

- Priest says that most logicians take $\models$ to be more basic or more important than $\vdash$.

Priority

- Priest says that most logicians take $\models$ to be more basic or more important than $\vdash$.
- I'm not 100% sure of the sociological claim here.

Priority

- Priest says that most logicians take $\vDash$ to be more basic or more important than $\vdash$.
- I'm not 100% sure of the sociological claim here.
- FWIW, I'm one of the minority (or perhaps not minority) that doesn't.

Priority

- Priest says that most logicians take $\vDash$ to be more basic or more important than $\vdash$.
- I'm not 100% sure of the sociological claim here.
- FWIW, I'm one of the minority (or perhaps not minority) that doesn't.
- But in this book, we're very much starting with $\vDash$.

Seven Symbols

Classical Models

Entailment

Inputs

- An assignment function v is a function from sentences of the formal language to either 0 or 1.

Inputs

- An assignment function v is a function from sentences of the formal language to either 0 or 1.
- So the inputs are sentences in a particular language.

Sentences

- The sentences of the language are defined recursively.

Sentences

- The sentences of the language are defined recursively.
- You're possibly familiar with recursive definitions from other parts of math.

Sentences

- The sentences of the language are defined recursively.
- You're possibly familiar with recursive definitions from other parts of math.
- They have a base case, and a rule for generating more.

Sentences

- The sentences of the language are defined recursively.
- You're possibly familiar with recursive definitions from other parts of math.
- They have a base case, and a rule for generating more.
- E.g., 0 is a number (that's the base), and if n is a number, then $n+1$ is a number (that's the rule).

Basic Sentences

- These are propositional variables.

Basic Sentences

- These are propositional variables.
- We'll write them as $p_0, p_1, p_2, \ldots$.

Basic Sentences

- These are propositional variables.
- We'll write them as $p_0, p_1, p_2, \ldots$.
- We assume there are a countable infinity of them.

Basic Sentences

- These are propositional variables.
- We'll write them as $p_0, p_1, p_2, \ldots$.
- We assume there are a countable infinity of them.
- Does everyone know what "countable infinity" means?

  If not, we'll stop and go over it.

Building rule

If A and B are sentences (of arbitrary complexity), then so are:

- $\neg A$; $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$, $(A \equiv B)$.

Building rule

If A and B are sentences (of arbitrary complexity), then so are:

- $\neg A$; $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$, $(A \equiv B)$.
- When it is clear, we omit the outermost parentheses. E.g., we'll write $p_0 \wedge p_1$ as a sentence although strictly speaking it is not.

That's all clause

- Why isn't this table a number? How do we know it isn't?

That’s all clause

- Why isn’t this table a number? How do we know it isn’t?
- Answer, it isn’t generated by adding 1 to a number, and that’s the only way to generate numbers.

That's all clause

- Why isn't this table a number? How do we know it isn't?
- Answer, it isn't generated by adding 1 to a number, and that's the only way to generate numbers.
- We can do the same thing to rule out some things as sentences.

Rules

A valuation function for classical propositional logic is any function defined over these sentences (and nothing else) that satisfies the clauses on the next three slides.

- For any $i \in \mathbb{N}$, $v(p_i) \in \{0, 1\}$.

Boolean Rules

For any A, B:

- $v(\neg A) = 1$ if $v(A) = 0$, and $v(\neg A) = 0$ otherwise.

Boolean Rules

For any A, B:

- $v(\neg A) = 1$ if $v(A) = 0$, and $v(\neg A) = 0$ otherwise.
- $v(A \wedge B) = 1$ if $v(A) = 1$ and $v(B) = 1$, and $v(A \wedge B) = 0$ otherwise.

Boolean Rules

For any A, B:

- $v(\neg A) = 1$ if $v(A) = 0$, and $v(\neg A) = 0$ otherwise.
- $v(A \wedge B) = 1$ if $v(A) = 1$ and $v(B) = 1$, and $v(A \wedge B) = 0$ otherwise.
- $v(A \vee B) = 1$ if $v(A) = 1$ or $v(B) = 1$, and $v(A \vee B) = 0$ otherwise.

Conditional Rules

For any A, B:

- $v(A \supset B) = 1$ if $v(A) = 0$ or $v(B) = 1$, and $v(A \supset B) = 0$ otherwise.

Conditional Rules

For any A, B:

- $v(A \supset B) = 1$ if $v(A) = 0$ or $v(B) = 1$, and $v(A \supset B) = 0$ otherwise.
- $v(A \equiv B) = 1$ if $v(A) = v(B)$, and $v(A \equiv B) = 0$ otherwise.

Worksheet

We'll leave the slides for a bit and go over some worked examples.

Seven Symbols

Classical Models

Entailment

Definition

- $\Gamma \vDash A$ just in case whenever every sentence in Γ gets value 1, so does A.

Working Out

- Assume Γ has finitely many elements. (The infinite case turns out to be the same, but it's messier.)

Working Out

- Assume Γ has finitely many elements. (The infinite case turns out to be the same, but it's messier.)
- Then there are k different sentence letters in $\Gamma \cup \{A\}$.

Working Out

- Assume Γ has finitely many elements. (The infinite case turns out to be the same, but it's messier.)
- Then there are k different sentence letters in $\Gamma \cup \{A\}$.
- So there are 2^k different v that are seeded by the assignment of each of these letters to 0 or 1.

Working out

- So here's one way to test for validity.

Working out

- So here's one way to test for validity.
- Go through all 2^k options, and check if any of them make everything in Γ true, and A false.

Working out

- So here's one way to test for validity.
- Go through all 2^k options, and check if any of them make everything in Γ true, and A false.
- If any do, argument is invalid (in classical propositional logic).

Working out

- So here's one way to test for validity.
- Go through all 2^k options, and check if any of them make everything in Γ true, and A false.
- If any do, argument is invalid (in classical propositional logic).
- If none do, argument is valid (in classical propositional logic).

A Worked Example

$A \vDash (AsupsetB) \supset A \vDash A$

- How many sentence letters?

A Worked Example

$$A \vDash (AsupsetB) \supset A \vDash A$$

- How many sentence letters?
- How many v to check?

$v(A) = 1, v(B) = 1$

- $v(A) = 1,$

$v(A) = 1, v(B) = 1$

- $v(A) = 1$,
- $v(A \supset B) = 1$,

$v(A) = 1, v(B) = 1$

- $v(A) = 1$,
- $v(A \supset B) = 1$,
- So $v((A \supset B) \supset A) = 1)$.

$v(A) = 1, v(B) = 1$

- $v(A) = 1$,
- $v(A \supset B) = 1$,
- So $v((A \supset B) \supset A) = 1)$.
- So this is a case where Γ gets value 1, and so does A.

  No problem

$v(A) = 1, v(B) = 0$

- $v(A) = 1,$

$v(A) = 1, v(B) = 0$

- $v(A) = 1$,
- $v(A \supset B) = 0$,

$v(A) = 1, v(B) = 0$

- $v(A) = 1$,
- $v(A \supset B) = 0$,
- So $v((A \supset B) \supset A) = 1$).

$v(A) = 1, v(B) = 0$

- $v(A) = 1$,
- $v(A \supset B) = 0$,
- So $v((A \supset B) \supset A) = 1)$.
- So this is a case where Γ gets value 1, and so does A.

  No problem.

$v(A) = 0, v(B) = 1$

- $v(A) = 0$,

$v(A) = 0, v(B) = 1$

- $v(A) = 0$,
- $v(A \supset B) = 1$,

$v(A) = 0, v(B) = 1$

- $v(A) = 0$,
- $v(A \supset B) = 1$,
- So $v((A \supset B) \supset A) = 0)$.

$v(A) = 0, v(B) = 1$

- $v(A) = 0$,
- $v(A \supset B) = 1$,
- So $v((A \supset B) \supset A) = 0)$.
- So this is a case where Γ gets value 0, so it can't be a problem.

$v(A) = 0, v(B) = 0$

- $v(A) = 0$,

$v(A) = 0, v(B) = 0$

- $v(A) = 0$,
- $v(A \supset B) = 1$,

$v(A) = 0, v(B) = 0$

- $v(A) = 0$,
- $v(A \supset B) = 1$,
- So $v((A \supset B) \supset A) = 0$).

$v(A) = 0, v(B) = 0$

- $v(A) = 0$,
- $v(A \supset B) = 1$,
- So $v((A \supset B) \supset A) = 0$).
- So this is a case where Γ gets value 0, so it can't be a problem.

Summary

There is no case where all of Γ is 1 and A is false, so valid.

Another worked example

$A \vDash (A supset B) \supset B \vDash A$

- This one is not valid; can you find the v that's a problem.

The Counterexample

- $v(A) = 0, v(B) = 1.$

The Counterexample

- $v(A) = 0, v(B) = 1.$
- Then $v(A \supset B) = 1.$

The Counterexample

- $v(A) = 0, v(B) = 1.$
- Then $v(A \supset B) = 1.$
- So $v((A \supset B) \supset B) = 1.$

The Counterexample

- $v(A) = 0, v(B) = 1.$
- Then $v(A \supset B) = 1.$
- So $v((A \supset B) \supset B) = 1.$
- But $v(A) = 0.$

A Familiar Example

- $A \vee B, \neg B \vDash A$

$v(A) = 1, v(B) = 1$

- $v(A \vee B) = 1$, but $v(\neg B) = 0$, so not all of Γ is 1.

$v(A) = 1, v(B) = 0$

- $v(A \vee B) = 1$, and $v(\neg B) = 0$, so all of Γ is 1.

$v(A) = 1, v(B) = 0$

- $v(A \lor B) = 1$, and $v(\neg B) = 0$, so all of Γ is 1.
- But $v(A) = 1$, so it's not a counterexample.

$v(A) = 0, v(B) = 1$

- $v(A \vee B) = 1$, but $v(\neg B) = 0$, so not all of Γ is 1.

$v(A) = 0, v(B) = 0$

- $v(A \vee B) = 1$, so not all of Γ is 1.

Summary

There is no case where **both** premises are 1, and conclusion is 0, so it is valid.

Last Example for Today

$$A \supset B, B \vDash A$$

- Can you find the counterexample?

Counterexample

$v(A) = 0, v(B) = 1.$

- Clearly that makes second premise 1 and conclusion 0.

Counterexample

$v(A) = 0, v(B) = 1.$

- Clearly that makes second premise 1 and conclusion 0.
- And reading off the rule for $\supset$, first premise is 1 as well.

For Next Time

- We'll look at how to use tableau to do proofs, which is much quicker than this when $k > 2$.

For Next Time

- We'll look at how to use tableau to do proofs, which is much quicker than this when $k > 2$.
- And we'll talk about how these formal sentences relate to English/other natural language expressions.