Today’s topics:

• finishing propositional logic: proof theory
• predicate logic
• what were those Greek letters from last time?!?!?

\( \phi \) phi
\( \Phi \) capital Phi
\( \psi \) psi
\( \pi \) pi
\( \tau \) tau
\( \chi \) chi

Proof Theory

• What is logic used for? A number of things, but most importantly, it is a language for representing the properties of things.
• But also, we hope it will give us a method for establishing the properties of things.
• To see how logic may be used to establish the properties of things, it helps to look at its history.
• Logic was originally developed to make the notion of an argument precise.

(We do not mean argument as in fighting here!)

• Here is a classic argument:

  All men are mortal
  Socrates is a man
  Socrates is mortal

• This example serves to illustrate a number of features of arguments:
  – The argument has a number of premises — these are the statements that appear before the horizontal line;
  – The argument has a conclusion — this is the statement that appears after the horizontal line;
  – The argument has the form

    \[
    \text{If you accept that the premises are true then you must accept that the conclusion is true.}
    \]
In mathematics, we are concerned with when arguments are sound.

To formalise the notion of a sound argument, we need some extra terminology...

**Definition:** If $\phi \in WFF$, then:

1. if there is some interpretation $\pi$ such that
   \[ \pi \models \phi \]
   then $\phi$ is said to be **satisfiable**, otherwise $\phi$ is **unsatisfiable**.
2. if
   \[ \pi \models \phi \]
   for all interpretations $\pi$, then $\phi$ is said to be **valid**.

Valid formulae of propositional logic are called **tautologies**.

**Theorem:**
1. If $\phi$ is a valid formula, then $\neg \phi$ is unsatisfiable;
2. If $\neg \phi$ is unsatisfiable, then $\phi$ is valid.

We indicate that a formula $\phi$ is valid by writing
\[ \models \phi. \]

We can now define the logical consequence.

**Definition:** If $\{\phi_1, \ldots, \phi_n\} \subseteq WFF$
then $\phi$ is said to be a **logical consequence** of $\{\phi_1, \ldots, \phi_n\}$ iff $\phi$ is satisfied by all interpretations that satisfy
\[ \phi_1 \land \cdots \land \phi_n. \]

We indicate that $\phi$ is a logical consequence of $\phi_1, \ldots, \phi_n$ by writing
\[ \{\phi_1, \ldots, \phi_n\} \models \phi. \]

An expression like this is called a **semantic sequent**.

**Theorem:**
\[ \{\phi_1, \ldots, \phi_n\} \models \phi. \]
iff
\[ \models (\phi_1 \land \cdots \land \phi_n) \Rightarrow \phi. \]

So we have a method for determining whether $\phi$ is a logical consequence of $\phi_1, \ldots, \phi_n$: we use a truth table to see whether $\phi_1 \land \cdots \land \phi_n \Rightarrow \phi$ is a tautology. If it is, then $\phi$ is a logical consequence of $\phi_1, \ldots, \phi_n$.

Our main concern in proof theory is thus to have a technique for determining whether a given formula is valid, as this will then give us a technique for determining whether some formula is a logical consequence of some others.
EXAMPLE: Show that $p \land q \models p \lor q$.

To do this, we construct a truth-table for $(p \land q) \Rightarrow (p \lor q)$.

Here it is:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$(p \land q)$</th>
<th>$p \lor q$</th>
<th>$(p \land q) \Rightarrow (p \lor q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
</tbody>
</table>

Since $(p \land q) \Rightarrow (p \lor q)$ is true under every interpretation, we have that $p \lor q$ is a logical consequence of $p \land q$.

• The notion of logical consequence we have defined above is acceptable for a definition of a sound argument, but is not very helpful for checking whether a particular argument is sound or not.

• The problem is that we must look at all the possible interpretations of the primitive propositions. While this is acceptable for, say, 4 primitive propositions, it will clearly be unacceptable for 100 propositions, as it would mean checking $2^{100}$ interpretations. (Moreover, for first-order logic, there will be an infinite number of such interpretations.)

• What we require instead is an alternative version of logical consequence, that does not involve this kind of checking. This leads us to the idea of syntactic proof.

'Syntactic' Proof

• The idea of syntactic proof is to replace the semantic checking to determine whether a formula is valid by a procedure that involves purely syntactic manipulation.

• The kinds of techniques that we shall use are similar to those that we use when solving problems in algebra.

• The basic idea is that to show that $\phi$ is a logical consequence of $\phi_1, \ldots, \phi_n$, we use a set of rules to manipulate formulae.

If we can derive $\phi$ from $\phi_1, \ldots, \phi_n$ by using these rules, then $\phi$ is said to be proved from $\phi_1, \ldots, \phi_n$, which we indicate by writing $\phi_1, \ldots, \phi_n \vdash \phi$.

• The symbol $\vdash$ is called the syntactic turnstile.

• An expression of the form $\phi_1, \ldots, \phi_n \vdash \phi$ is called a syntactic sequent.

• A rule has the general form:

  $\vdash \phi_1, \ldots, \vdash \phi_n$ rule name

  $\vdash \phi$

Such a rule is read:

If $\phi_1, \ldots, \phi_n$ are proved then $\phi$ is proved.
---

**EXAMPLE:** Here is an example of such a rule:

\[ \Gamma \phi \vdash \psi \quad \land I \]

\[ \Gamma \phi \land \psi \]

This rule is called *and introduction*. It says that if we have proved \( \phi \), and we have also proved \( \psi \), then we can prove \( \phi \land \psi \).

**EXAMPLE:** Here is another rule:

\[ \Gamma \phi \land \psi \quad \land E \]

\[ \Gamma \phi, \Gamma \psi \]

This rule is called *and elimination*. It says that if we have proved \( \phi \land \psi \), then we can prove both \( \phi \) and \( \psi \); it allows us to eliminate the \( \land \) symbol from between them.

---

**Definition:** (Proof) If \( \{ \phi_1, \ldots, \phi_m, \phi \} \subseteq WFF \) then there is a proof of \( \phi \) from \( \phi_1, \ldots, \phi_m \) iff there exists some sequence of formulae \( \psi_1, \ldots, \psi_n \) such that \( \psi_n = \phi \), and each formula \( \psi_k \), for \( 1 \leq k < n \), is either one of the formula \( \phi_1, \ldots, \phi_m \), or else is the conclusion of a rule whose antecedents appeared earlier in the sequence.

---

**Definition:** (Soundness) A proof system \( \vdash \) is said to be sound with respect to semantics \( \models \) iff

\[ \phi_1, \ldots, \phi_n \vdash \phi \]

implies

\[ \phi_1, \ldots, \phi_n \models \phi. \]

**Definition:** (Completeness) A proof system \( \vdash \) is said to be complete with respect to semantics \( \models \) iff

\[ \phi_1, \ldots, \phi_n \vdash \phi \]

implies

\[ \phi_1, \ldots, \phi_n \models \phi. \]
A Proof System

- There are many proof systems for propositional logic; we shall look at a simple one.
- First, we have an unusual rule that allows us to introduce any tautology.
  \[ \text{TAUT} \quad \frac{}{\phi} \text{ if } \phi \text{ is a tautology} \]
- Because a tautology is true there is no problem bringing it into the proof.

- Next, rules for eliminating connectives.
  \[ \begin{align*}
  \vdash \phi \land \psi; \\
  \vdash \phi \Rightarrow \chi; \\
  \vdash \psi \Rightarrow \chi
  \end{align*} \]
  \[ \vdash \chi \]

- An alternative \( \lor \) elimination rule is:
  \[ \begin{align*}
  \vdash \phi \lor \psi; \\
  \vdash \phi \Rightarrow \chi; \\
  \vdash \psi \Rightarrow \chi
  \end{align*} \]
  \[ \vdash \chi \]

- Next, a rule called modus ponens, which lets us eliminate \( \Rightarrow \).
  \[ \begin{align*}
  \vdash \phi \Rightarrow \psi; \vdash \phi \\
  \vdash \psi
  \end{align*} \]

- We have a rule called the deduction theorem. This rule says that if we can prove \( \psi \) from \( \phi \), then we can prove that \( \phi \Rightarrow \psi \).
  \[ \begin{align*}
  \phi \vdash \psi \\
  \vdash \phi \Rightarrow \psi
  \end{align*} \]

- There are a whole range of other rules, which we shall not list here.
Proof Examples

• In this section, we give some examples of proofs in the propositional calculus.

• Example 1:
  \( p \land q \vdash q \land p \)

  1. \( p \land q \) Given
  2. \( p \) From 1 using \( \land \)-E
  3. \( q \) 1, \( \land \)-E
  4. \( q \land p \) 2, 3, \( \land \)-I

• Example 2:
  \( p \land q \vdash p \lor q \)

  1. \( p \land q \) Given
  2. \( p \) 1, \( \land \)-E
  3. \( p \lor q \) 2, \( \lor \)-I

• Example 3:
  \( p \land q, p \Rightarrow r \vdash r \)

  1. \( p \land q \) Given
  2. \( p \) 1, \( \land \)-E
  3. \( p \Rightarrow r \) Given
  4. \( r \) 2, 3, \( \Rightarrow \)-E

• Example 4:
  \( p \Rightarrow q, q \Rightarrow r \vdash p \Rightarrow r \)

  1. \( p \Rightarrow q \) Given
  2. \( q \Rightarrow r \) Given
  3. \( p \) Assumption
  4. \( q \) 1, 3, \( \Rightarrow \)-E
  5. \( r \) 2, 4, \( \Rightarrow \)-E
  6. \( p \Rightarrow r \) 3, 5, \( \Rightarrow \)-I
• Example 5:
\((p \land q) \Rightarrow r \vdash p \Rightarrow (q \Rightarrow r)\)

1. \((p \land q) \Rightarrow r\) Given
2. \(p\) Assumption |
3. \(q\) Assumption ||
4. \(p \land q\) 2, 3, \&-I ||
5. \(r\) 1, 4, \&\(-I ||
6. \(q \Rightarrow r\) 3–5, \&\(-I ||
7. \(p \Rightarrow (q \Rightarrow r)\) 2–6, \&\(-I

• Example 6:
\(p \Rightarrow (q \Rightarrow r) \vdash (p \land q) \Rightarrow r\)

1. \(p \Rightarrow (q \Rightarrow r)\) Given
2. \(p \land q\) Assumption |
3. \(p\) 2, \&-E |
4. \(q\) 2, \&-E |
5. \(q \Rightarrow r\) 1, 3, \&\(-E |
6. \(r\) 4, 5, \&\(-E |
7. \((p \land q) \Rightarrow r\) 2–6, \&\(-I

• Example 7:
\(p \Rightarrow q, \neg q \vdash \neg p\)

1. \(p \Rightarrow q\) Given
2. \(\neg q\) Given
3. \(p\) Assumption |
4. \(q\) 1, 3, \&\(-E |
5. \(q \land \neg q\) 2, 4, \&\(-I |
6. \(\neg p\) 3, 5, \neg\(-I

• Example 8:
\(p \Rightarrow q \vdash \neg(p \land \neg q)\)

1. \(p \Rightarrow q\) Given
2. \(p \land \neg q\) Assumption |
3. \(p\) 2, \&-E |
4. \(\neg q\) 2, \&-E |
5. \(q\) 1, 3, \&\(-E |
6. \(q \land \neg q\) 4, 5, \&-E |
7. \(\neg(p \land \neg q)\) 6, \neg\(-I
Example 9:

Jim will party all night and pass AI? That must be wrong. If he works hard he won’t have time to party. If he doesn’t work hard he’s not going to pass AI.

Let:

\[ p \] Jim will party all night
\[ q \] Jim will pass AI
\[ r \] Jim works hard

Formalisation of argument:

\[ r \implies \neg p, \neg r \implies \neg q \vdash \neg (p \land q) \]

Predicate Logic

- And now, we introduce first-order predicate logic.
- More expressive than propositional logic.
- Consider the following argument:
  - all monitors are ready;
  - \( X12 \) is a monitor;
  - therefore \( X12 \) is ready.
- Sense of this argument cannot be captured in propositional logic.
- Propositional logic is too coarse grained to allow us to represent and reason about this kind of statement.

Syntax

- We shall now introduce a generalisation of propositional logic called first-order logic (FOL). This new logic affords us much greater expressive power.
- Definition: The alphabet of FOPL contains:
  1. a set of constants;
  2. a set of variables;
  3. a set of function symbols;
  4. a set of predicates symbols;
  5. the connectives \( \lor, \neg \);
  6. the quantifiers \( \forall, \exists, \exists! \);
  7. the punctuation symbols \( (),. \)
Terms

- The basic components of FOL are called terms.
- Essentially, a term is an object that denotes some object other than \( \top \) or \( \bot \).
- The simplest kind of term is a constant.
- A value such as 8 is a constant.
- The denotation of this term is the number 8.
- Note that a constant and the number it denotes are different!
- Aliens don’t write “8” for the number 8, and nor did the Romans.

Functions

- We can now introduce a more complex class of terms — functions.
- The idea of functional terms in logic is similar to the idea of a function in programming: recall that in programming, a function is a procedure that takes some arguments, and returns a value.
  In C:
  ```c
  T myfunction( T1 a1, ..., Tn an ) {
    ...
  }
  ```
  this function takes \( n \) arguments; the first is of type \( T1 \), the second is of type \( T2 \), and so on. The function returns a value of type \( T \).
- In FOL, we have a set of function symbols; each symbol corresponds to a particular function. (It denotes some function.)

- The second simplest kind of term is a variable.
- A variable can stand for anything in the domain of discourse.
- The domain of discourse (usually abbreviated to domain) is the set of all objects under consideration.
- Sometimes, we assume the set contains “everything”.
- Sometimes, we explicitly give the set, and state what variables/constants can stand for.

- Each function symbol is associated with a number called its arity. This is just the number of arguments it takes.
- A functional term is built up by applying a function symbol to the appropriate number of terms.
- Formally …
  **Definition:** Let \( f \) be an arbitrary function symbol of arity \( n \). Also, let \( \tau_1, \ldots, \tau_n \) be terms. Then
  \[
  f(\tau_1, \ldots, \tau_n)
  \]
  is a functional term.
• All this sounds complicated, but isn’t. Consider a function `plus`, which takes just two arguments, each of which is a number, and returns the first number added to the second. Then:
  – `plus(2, 3)` is an acceptable functional term;
  – `plus(0, 1)` is acceptable;
  – `plus(plus(1, 2), 4)` is acceptable;
  – `plus(plus(plus(0, 1), 2), 4)` is acceptable;

• In maths, we have many functions; the obvious ones are:
  \[ + \ - \ * \ \sqrt{\cdots} \sin \ \cos \ \cdots \]

• The fact that we write
  \[ 2 + 3 \]
  instead of something like
  \[ \text{plus}(2, 3) \]
  is just convention, and is not relevant from the point of view of logic; all these are functions in exactly the way we have defined.

• Using functions, constants, and variables, we can build up expressions, e.g.:
  \[(x + 3) \times \sin 90\]
  (which might just as well be written
  \[\times(\plus(x, 3), \sin(90))\]
  for all it matters.)

Predicates

• In addition to having terms, FOL has relational operators, which capture relationships between objects.
• The language of FOL contains predicate symbols.
• These symbols stand for relationships between objects.
• Each predicate symbol has an associated arity (number of arguments).
• Definition: Let \( P \) be a predicate symbol of arity \( n \), and \( \tau_1, \ldots, \tau_n \) are terms. Then
  \[ P(\tau_1, \ldots, \tau_n) \]
  is a predicate, which will either be \( T \) or \( \bot \) under some interpretation.
EXAMPLE. Let $gt$ be a predicate symbol with the intended interpretation 'greater than'. It takes two arguments, each of which is a natural number. Then:
- $gt(4, 3)$ is a predicate, which evaluates to $\top$;
- $gt(3, 4)$ is a predicate, which evaluates to $\bot$.

The following are standard mathematical predicate symbols:
- $> \leq \geq \neq \ldots$

The fact that we are normally write $x > y$ instead of $gt(x, y)$ is just convention.

We can build up more complex predicates using the connectives of propositional logic:

$(2 > 3) \land (6 = 7) \lor (\sqrt{4} = 2)$

So a predicate just expresses a relationship between some values.

What happens if a predicate contains variables: can we tell if it is true or false? Not usually; we need to know an interpretation for the variables.

A predicate that contains no variables is a proposition.

- We now come to the central part of first order logic: quantification.
- Consider trying to represent the following statements:
  - all men have a mother;
  - every positive integer has a prime factor.
- We can’t represent these using the apparatus we’ve got so far; we need quantifiers.
We use three quantifiers:

- \(\forall\) — the universal quantifier; is read 'for all…'
- \(\exists\) — the existential quantifier; is read 'there exists…'
- \(\exists_1\) — the unique quantifier; is read 'there exists a unique…'

The simplest form of quantified formula is as follows:

\[
\text{quantifier variable} \cdot \text{predicate}
\]

where

- quantifier is one of \(\forall, \exists, \exists_1\);
- variable is a variable;
- and predicate is a predicate.

Examples

- \(\forall x \cdot \text{Man}(x) \Rightarrow \text{Mortal}(x)\)
  'For all \(x\), if \(x\) is a man, then \(x\) is mortal.'
  (i.e. all men are mortal)
- \(\forall x \cdot \text{Man}(x) \Rightarrow \exists y \cdot \text{Woman}(y) \land \text{MotherOf}(x,y)\)
  'For all \(x\), if \(x\) is a man, then there exists exactly one \(y\) such that \(y\) is a woman and the mother of \(x\) is \(y\).'
  (i.e., every man has exactly one mother).
- \(\exists m \cdot \text{Monitor}(m) \land \text{MonitorState}(m, \text{ready})\)
  'There exists a monitor that is in a ready state.'
- \(\forall r \cdot \text{Reactor}(r) \Rightarrow \exists t \cdot (100 \leq t \leq 1000) \land \text{temp}(r) = t\)
  'Every reactor will have a temperature in the range 100 to 1000.'
• \( \exists n \cdot \text{posInt}(n) \land n = (n \times n) \)
  “Some positive integer is equal to its own square.”
• \( \exists c \cdot \text{EUcountry}(c) \land \text{Borders}(c, \text{Albania}) \)
  “Some EU country borders Albania.”
• \( \forall m, n \cdot \text{Person}(m) \land \text{Person}(n) \Rightarrow \neg \text{Superior}(m, n) \)
  “No person is superior to another.”
• \( \forall m \cdot \text{Person}(m) \Rightarrow \neg \exists n \cdot \text{Person}(n) \land \text{Superior}(m, n) \)
  (same as previous)

---

• Suppose our intended interpretation is the positive integers. Suppose \( >, +, \times, \ldots \) have the usual mathematical interpretation.
• Is this formula satisfiable under this interpretation?
  \( \exists n \cdot n = (n \times n) \)
• Now suppose that our domain is all living people, and that \( * \) means “is the child of”.
• Is the formula satisfiable under this interpretation?

---

• Domains & Interpretations
  
  • Suppose we have a formula \( \forall x \cdot P(x) \).
  What does \( x \) range over?
  Physical objects, numbers, people, times, \ldots?
  • Depends on the domain that we intend.
  • Often, we name a domain to make our intended interpretation clear.

---

• Comments
  
  • Note that universal quantification is similar to conjunction.
  Suppose the domain is the numbers \( \{2, 4, 6\} \). Then
  \[ \forall n \cdot \text{Even}(n) \]
  is the same as
  \[ \text{Even}(2) \land \text{Even}(4) \land \text{Even}(6) \].
• Existential quantification is similar to disjunction. Thus with the same domain,
  \[ \exists n \cdot \text{Even}(n) \]
  is the same as
  \[ \text{Even}(2) \lor \text{Even}(4) \lor \text{Even}(6) \].
The universal and existential quantifiers are in fact duals of each other:
\[ \forall x \cdot P(x) \iff \neg \exists x \cdot \neg P(x) \]
Saying that everything has some property is the same as saying that there is nothing that does not have the property.
\[ \exists x \cdot P(x) \iff \neg \forall x \cdot \neg P(x) \]
Saying that there is something that has the property is the same as saying that its not the case that everything doesn’t have the property.

Decidability

- In propositional logic, we saw that some formulae were tautologies — they had the property of being true under all interpretations.
- We also saw that there was a procedure which could be used to tell whether any formula was a tautology — this procedure was the truth-table method.
- A formula of FOL that is true under all interpretations is said to be valid.
- So in theory we could check for validity by writing down all the possible interpretations and looking to see whether the formula is true or not.

Unfortunately in general we can’t use this method.
- Consider the formula:
\[ \forall n \cdot \text{Even}(n) \Rightarrow \neg \text{Odd}(n) \]
- There are an infinite number of interpretations.
- Is there any other procedure that we can use, that will be guaranteed to tell us, in a finite amount of time, whether a FOL formula is, or is not, valid?
- The answer is no.
- FOL is for this reason said to be undecidable.

Proof in FOL

- Proof in FOL is similar to propositional logic (PL); we just need an extra set of rules, to deal with the quantifiers.
- FOL inherits all the rules of PL.
- To understand FOL proof rules, need to understand substitution.
- The most obvious rule, for \( \forall \)-E.

\[ \vdash \forall x \cdot \phi(x) \quad \forall \text{-E} \quad \vdash \phi(a) \]

Tells us that if everything in the domain has some property, then we can infer that any particular individual has the property.

Going from general to specific.
Example 1.
Let’s use \( \forall\)-E to get the Socrates example out of the way.

\[
\text{Man}(s) ; \forall x \cdot \text{Man}(x) \Rightarrow \text{Mortal}(x)
\]

\[
\vdash \text{Mortal}(s)
\]

1. \( \text{Man}(s) \)  
   Given
2. \( \forall x \cdot \text{Man}(x) \Rightarrow \text{Mortal}(x) \)  
   Given
3. \( \text{Man}(s) \Rightarrow \text{Mortal}(s) \)  
   2, \( \forall\)-E
4. \( \text{Mortal}(s) \)  
   1, 3, \( \Rightarrow\)-E

Existential Introduction Rule 1 (\( \exists\)-I(1)).

- We can also go from the general to the slightly less specific!

\[
\vdash \forall x \cdot \phi(x) ; \exists\-I(1) \text{ if domain not empty}
\]

Note the side condition.
The \( \exists \) quantifier asserts the existence of at least one object.
The \( \forall \) quantifier does not.

- Existential Introduction Rule 2 (\( \exists\)-I(2)).
- We can also go from the very specific to less specific.

\[
\vdash \phi(a) ; \exists\-I(2)
\]

\[
\vdash \exists x \cdot \phi(x)
\]

- In other words once we have a concrete example, we can infer there exists something with the property of that example.

- We often informally make use of arguments along the lines...
  1. We know somebody is the murderer.
  2. Call this person \( a \).
  3. . . .

(Here, \( a \) is called a Skolem constant.)
- We have a rule which allows this, but we have to be careful how we use it!

\[
\vdash \exists x \cdot \phi(x) ; \exists\-E\ a \text{ doesn’t occur elsewhere}
\]

\[
\vdash \phi(a)
\]
• Here is an invalid use of this rule:
  1. $\exists x \cdot Boring(x)$ Given
  2. Lecture(AI) Given
  3. Boring(AI) 1, $\exists$E

  (The conclusion may be true, the argument isn’t sound.)

• Example 2:
  1. Everybody is either happy or rich.
  2. Simon is not rich.
  3. Therefore, Simon is happy.

  Predicates:
  - $H(x)$ means $x$ is happy;
  - $R(x)$ means $x$ is rich.

  Formalisation:
  \[ \forall x. H(x) \lor R(x); \neg R(Simon) \vdash H(Simon) \]

Summary
• This lecture looked at predicate (or first order) logic.
• Predicate logic is a generalisation of propositional logic.
• The generalisation requires the use of quantifiers, and these need special rules for handling them when doing inference.
• We looked at how the proof rules for propositional logic need to be extended to handle quantifiers.