Introduction to Logic 1

- What is Logic?
- Why Study Logic?
- Object-language/Meta-language.
- Propositions, Beliefs and Contradictions.
- Formalisation

What is Logic?

Logic is . . .

- the study of the . . . principles used to distinguish good (correct) from bad (incorrect) reasoning (Copi);
- the study, by symbolic means, of the exact conditions under which patterns of argument are valid or invalid (Lemmon);
- the study of formal (that is symbolic) systems of reasoning and of methods of attaching meaning to them (Reeves & Clarke).

– Abstract principle or patterns
– Distinguishing valid/invalid patterns
– Formalisation/abstraction

Patterns of Reasoning

e.g.

If Logic is fun, then Bill is happy.
Logic is fun.
(It follows that) Bill is happy.

If mungs are flit, then mizzles are gloggy
Mungs are flit.
(It follows that) mizzles are gloggy.

Question: *Do the conclusions follow in the above?*

Questions: *What is noticeable about these two patterns?*

Abstraction

if A then B	$A \supset B$	Premiss
A	A	Premiss
————	———	————
B	B	Conclusion

Question: *What about the following pattern? Does it fit; is it good?*

If Logic is fun, then Bill is happy.
Programming is fun.
(It follows that) Bill is happy.

Why Study Logic?

A little history:

Aristotle (384–322 BC): first systematic study of "patterns of reasoning"; development of syllogistic reasoning.

Boole (1815–1864): develops algebraic system of symbol manipulation now regarded as the basis of propositional logic and computer hardware.

Frege (1848–1925): studies the foundations of mathematics with the objective of deriving all mathematics from logical principles; introduces a new notation and language which provides the basis of modern logic (the first order predicate calculus).

But why study logic as computer scientists?

Logic and Computer Science

- Foundational issues:

  – there are intimate links between computation and logic

- Analytic tool:

  – use of logic as a tool for formalizing/studying properties of programs.

- Hardware Design:

  – conventional computer hardware is is based on electronic devices called *logic gates*.

- Automated reasoning:

  – mechanical generation of proofs: automated theorem proving, artificial intelligence

  – logic as a programming language (e.g. Prolog)

Some Terminology

We will, in this course, be looking at *mathematical* (formal/symbolic) logic.

In fact, we are using mathematical techniques to study a branch of mathematics called 'logic'

Question: does this make sense?

- It does, as long as we are careful.
- We use different languages to separate the object of our study (i.e. logic) from the means of our study.
- The former is called the *object language*; the latter is called the *meta-language*.

Example

- "The expression 'Karl ist krank' is a well-formed sentence of German"

  Here, the object language is German, while the meta-language is English.

- "$x = 0$; is equivalent to $x = 1; x = x - 1$; in *Java*"

  In this case, the object language is presumably *Java*, while the meta-language is English again.

- "The sentence 'John likes Mary' is true"

  Note that is this case the object language is the same as the meta-language (i.e. English).

Propositions

Logic is is concerned with objects called *propositions* and the relationships between them, but what are propositions?

- Language can be used to *express* propositions:

  Bill teaches Logic
  Logic is taught by Bill
  I teach Logic

  – propositions communicate judgements or beliefs about the world.

- Note that not all sentences express propositions:

  Who teaches Logic?
  Teach Logic!!

  – **declarative sentences** express propositions

Simple test for declarative sentences:

"It is true that SENTENCE "

Beliefs and Contradictions

- Our beliefs provide our 'world view' or picture

  They allow us to reason about the world, to construct hypotheses and to draw conclusions

- Is there any restriction on the beliefs that we may hold or entertain?

  We cannot knowingly entertain, simultaneously, contradictory beliefs:

  "Bill teaches Logic/Bill does not teach Logic"

A fundamental task of logic is to be able to decide whether or not a set of beliefs (propositions) is contradictory.

Formalization

We are interested in a *formal* approach to the study of logic. Actually, there are two different senses of 'formalization' here:

1. The process of constructing an object language and the rules needed to manipulate sentences.
2. The provision of a means of manipulating objects according to their *form* rather than their *content*.

   i.e. we can work without understanding exactly what we are doing (there's no need to know what individual propositions mean or what the manipulations achieve)!

Summary

- Logic is concerned with abstract principles of reasoning and the notion of truth;
- Mathematical logic began at the turn of the last century (Frege);
- Logic is an important area of study for computer scientists;
- Logic deals with propositions, which express beliefs;
- Sets of beliefs may be contradictory (we would like to know when this is so);
- Formalization allows us to 'mechanize' the process of reasoning.

Introduction to Logic 2

Last time:

- What is Logic and why study it?
- Object-language/Meta-language.
- Propositions, Beliefs and Contradictions.
- Formalization

This time:

- The Propositional Calculus.
- The Language of Propositional Logic.
- What does it all mean?
- Arguments

Introduction to the Propositional Calculus

• The *Propositional Calculus* (PC) is a simple language for expressing certain kinds of propositions.

• Essentially, it allows us to write down Boolean combinations of simple declarative sentences.

For example:

- Logic is fun
- Logic is *not* fun
- Bill teaches Logic *and* Logic is fun
- Logic is fun *or* Bill is happy
- *if* Logic is fun, *then* Bill is happy

Connectives

- Statements are combined with words such as *not*, *and*, *or* and *if ..., then* to build more complex statements
  - The words will be called the **connectives**
  - The connectives are **truth-functional**

  The statement

  "It is raining and it is snowing"

  is **true** if
  - the statement "it is raining" is **true**; and
  - the statement "it is snowing" is **true**

  otherwise it is **false**.
- The truth-value of the whole can be calculated once the truth-values of the parts are known.

The Language of PC

- Rather than using the English connectives, we will use the following symbols:

Symbol	English Equivalent
$\neg$	not
$\wedge$	and
$\vee$	or
$\rightarrow$	implies (if ... then ...)
$\leftrightarrow$	... if and only if ...

- We will also require some further symbols:
  - left and right parentheses: ‘(‘ and ‘)‘
  - a stock of *propositional variables* : $p, q, r, s, \ldots$

  These symbols (connectives, propositional variables, parentheses) form the *alphabet* of our language.

The *well-formed formulas* (wffs) of the language are strings (i.e. sequences) of symbols from the alphabet.

Definition: *(The language of the PC)*

1. *Any propositional variable is a wff;*
2. *If A and B are wffs, then so are:*
   - $(\neg A)$
   - $(A \wedge B)$
   - $(A \vee B)$
   - $(A \rightarrow B)$
   - $(A \leftrightarrow B)$
3. *Nothing is a wff except in virtue of 1 and 2 above.*

NB: I may also refer to a wff as a *sentence* or a *statement*.

Which of these are wffs?

p

$(p \wedge q)$

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

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

$(p \rightarrow \wedge q)$

$p \vee q$

$(A \wedge B)$

A Note about Brackets

In practice, we adopt conventions that permit parentheses to be dropped where no confusion can arise from doing so

Operator Precedence: ¬ takes precedence over ∧ and ∨, so:

$$(((\neg p) \wedge q) \rightarrow r)$$
becomes
$$((\neg p \wedge q) \rightarrow r)$$

and ∧ and ∨ in turn take precedence over → and ↔ so:

$$((\neg p \wedge q) \rightarrow r)$$
becomes
$$(\neg p \wedge q \rightarrow r)$$

Outermost Parentheses: these can always be dropped, so:

$$(\neg p \wedge q \rightarrow r)$$
becomes
$$\neg p \wedge q \rightarrow r$$

What does it all mean?

- We have provided a definition of the wffs of the PC.
  - This tells us what *form* they take (their *syntax*)
  - It does not tell us about their meaning (their *semantics*)
- **Questions:**
  - How do we assign meaning to the sentences of the PC?
  - What do we mean by ‘meaning’ anyway?

Meaning and Truth

- Sentences express propositions, and these may be either **true** or **false**.
  - we say that sentences denote *truth-values*
- We have already noted that the connectives are *truth-functional*
  - they allow us to calculate the meaning (i.e. truth-value) of complex sentences as a function of the meaning (i.e. truth-values) of their parts.

Arguments

- Part of our motivation for introducing Propositional Logic is to formalize and study ‘patterns of reasoning’ or *arguments*.
- We have seen some examples of arguments, both ‘good’ and ‘bad’:

  e.g.

  If Logic is fun, then Bill is happy

  Logic is fun

  Therefore, Bill is happy

  If Logic is fun, then Bill is happy

  Programming is fun

  Therefore, Bill is happy

- We can now ‘formalize’ these arguments (and many others) in the sense of providing an abstract representation of their *structure*.
- Consider the first argument given on the last overhead. Let

p	stand for	‘Logic is fun’
q	stand for	’Bill is happy’

  then, we can write:

  $$\begin{array}{l} (p \rightarrow q) \\ p \\ \hline q \end{array}$$

- Or perhaps:

$$((p \rightarrow q) \wedge p) \rightarrow q$$

Summary

- The Propositional Calculus (PC) is a simple language for expressing propositions
- Sentences of the PC are built up from propositional variables, connectives and parentheses.
- Sentences of the PC are either **true** or **false**
- Connectives are truth-functional
- The PC allows us to formalize the structure of arguments
- The formal representations are *abstract*.

Introduction to Logic 3

Last time:

- Introduction to the PC
- The Language of the PC
- Giving the language meaning
- Arguments

This time:

- Truth-values and the connectives
- Truth tables
- Tautologies and Inconsistencies
- Arguments revisited

Truth Tables and the Connectives

- We said last time that sentences of the PC express propositions and may be either **true** or **false**. What exactly, are we assuming?
  - there are only two truth values: **true** and **false**
  - sentences cannot be *both* **true** and **false** simultaneously.
  - sentences cannot be ‘undefined’ (i.e. there are not truth-value ‘gaps’)
- These are fundamental assumptions of ‘classical’ logic

Questions: *Could you have a non-classical logic? What might that be like?*

- The simplest sentence-types of our language are the propositional variables:

$$p, q, r, s, \ldots$$

- These are combined with the connectives to build ‘complex’ or ‘compound’ sentences:

$$((p \to q) \wedge p) \to q$$

- Clearly, the truth-value of a compound sentence depends on:
  1. the truth-values of the propositional variables that it contains;
  2. the meaning (i.e. truth-functions) of the connectives $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$.

- For classical logic, this is *all* we need to know.
  - We don’t need to know *how* the variables got their values;
  - We don’t need to know anything about the meaning of sentences beyond their truth-values.
- **Analogy:** (arithmetic)
  - Suppose that variable x has value 3, y has value 2 and z has value 5.
  - Given that you know the meaning of $+$ and $-$, you can calculate the value of the expression:

$$(x + y) - z$$

  - Thus: $(3 + 2) - 5 = 5 - 5 = 0$

Truth Tables

- The connectives of our language are truth-functional
- The truth-functions that they correspond to can be expressed conveniently in the form of matrices:

$\neg$	
t	f
f	t

$\wedge$	t	f
t	t	f
f	f	f

$\vee$	t	f
t	t	t
f	t	f

$\rightarrow$	t	f
t	t	f
f	t	t

$\leftrightarrow$	t	f
t	t	f
f	f	t

Example

- Suppose that we want to determine the truth-value of the sentence

$$p \rightarrow q$$

given that $p =$ t and $q =$ f.

- We know the truth-function for $\rightarrow$:

$\rightarrow$	t	$q =$ f
$p =$ t	t	$p \rightarrow q =$ f
f	t	t

- So, in this case $p \rightarrow q$ is **false** .
- Here's the case when $p =$ t and $q =$ t:

$\rightarrow$	$q =$ t	f
$p =$ t	$p \rightarrow q =$ t	f
f	t	t

- Note that the truth-value of a compound sentence can vary according to (as a function of!) the truth-values of its parts.
- Given a sentence of the PC, we can display all of the different possible cases in the form of a matrix or **truth table**:

  e.g. $p \rightarrow q$

p	q	$p \rightarrow q$
t	t	t
t	f	f
f	t	t
f	f	t

- Thus we see that $p \rightarrow q$ is alway **true** *except* in case that p is **true** and q is **false** .

- Here's a more complicated example:

$$((p \rightarrow q) \wedge p) \rightarrow q$$

p	q	$(p \rightarrow q)$	$(p \rightarrow q) \wedge p$	$((p \rightarrow q) \wedge p) \rightarrow q$
t	t	t	t	t
t	f	f	f	t
f	t	t	f	t
f	f	t	f	t

- We calculate the truth-value of the whole expression 'inside-out'.
- Each line of the table corresponds to one way of assigning truth-values to the propositional variables in the sentence
- a *function* from propositional variables to truth-values is called a *valuation*.

Tautologies, Inconsistencies and Equivalences

- A **tautology** is a sentence that is **true** in all possible valuations.
- Consider the sentence $p \vee \neg p$:

p	$\neg p$	$p \vee \neg p$
t	f	t
f	t	t

- We've already seen an example of a sentence that is *not* a tautology:

p	q	$p \to q$
t	t	t
t	f	f
f	t	t
f	f	t

- An **inconsistency** is a sentence that is **false** in all possible valuations:
- Consider the sentence $p \wedge \neg p$:

p	$\neg p$	$p \wedge \neg p$
t	f	f
f	t	f

  Clearly, $p \wedge \neg p$ is inconsistent.

- A sentence is **contingent** if it is *neither* tautologous *nor* inconsistent.

p	q	$p \to q$
t	t	t
t	f	f
f	t	t
f	f	t

  So, $p \to q$ is contingent.

- Two sentences A and B are said to be **equivalent** if, for any given valuation, they have exactly the same truth-value.
- Consider $\neg p \vee q$ and $p \to q$:

p	q	$\neg p$	$\neg p \vee q$	$p \to q$
t	t	f	t	t
t	f	f	f	f
f	t	t	t	t
f	f	t	t	t

- Note that the last two columns are identical, row-by-row. So, $\neg p \vee q$ is equivalent to $p \to q$
- **Question** *Suppose that A and B are equivalent. What can you say about $(A \leftrightarrow B)$?*

Arguments Revisited

- We now have a way of distinguishing between 'good' (i.e. valid) and 'bad' (i.e. invalid) arguments.
- Intuitively, an argument is valid if whenever all of its premisses $P_1, P_2, \ldots, P_k$ are **true** , then its conclusion C is also **true** .
- In other words, the sentence:

$$(P_1 \wedge P_2 \wedge \ldots P_k) \to C$$

  is a tautology.

- So, to check whether an argument is valid we can
  - formalize the argument as a sentence of the PC
  - check whether the resulting sentence is a tautology

Example

Lets return to the argument that we formalized in the last lecture.

$$\begin{aligned} P_1 &= \text{If Logic is fun, then Bill is happy} \\ P_2 &= \text{Logic is fun} \\ C &= \text{Therefore, Bill is happy} \end{aligned}$$

So:

$$\begin{array}{ccccc} P_1 & \wedge & P_2 & \rightarrow & C \\ ((p \rightarrow q) & \wedge & p) & \rightarrow & q \end{array}$$

p	q	$(p \rightarrow q)$	$(p \rightarrow q) \wedge p$	$((p \rightarrow q) \wedge p) \rightarrow q$
t	t	t	t	t
t	f	f	f	t
f	t	t	f	t
f	f	t	f	t

Note that the final column contains only t. This means that the sentence is a tautology, and hence the argument is valid.

Summary

- Sentences of the PC can be either **true** or **false** (but not both and they cannot be undefined or have some other value).
- The connectives correspond to *truth-functions*
- Truth tables allow us to set out, systematically, the way the truth-value of a compound sentence varies according to the truth-values of its simpler parts.
- We can distinguish between sentences that are **true** in all valuations (tautologies), and **false** in all valuations (inconsistencies).
- Two sentences are *logically equivalent* if they have the same truth values in all possible valuations
- We can test the validity of arguments by formalizing them as sentences of the PC and then testing to see if they are tautologous.

Introduction to Logic 4

Last time:

- The meaning of the connectives
- Truth tables
- Tautologies, Inconsistencies and Equivalences
- Arguments and validity

This time:

- Functional completeness
- The Sheffer Stroke
- Logical Equivalences
- Limitations of Truth Tables

Functional Completeness

- Our definition of the PC includes the connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$.
- Our motivation for introducing these connectives came from considering sentences of English
- Perhaps surprisingly, it turns out that it is not necessary, strictly speaking, to have so many connectives.

Proposition: *We can express all of the connectives in terms of negation ($\neg$) and conjunction ($\wedge$).*

Proof (sketch):

We can show that for any sentences A and B:

$$\begin{aligned} A \vee B &\equiv \neg(\neg A \wedge \neg B) \\ A \rightarrow B &\equiv \neg(A \wedge \neg B) \\ A \leftrightarrow B &\equiv \neg(A \wedge \neg B) \wedge \neg(B \wedge \neg A) \end{aligned}$$

Proof continued:

- *We can show logical equivalence using the method of truth tables.*

  Consider: $A \vee B \equiv \neg(\neg A \wedge \neg B)$*:*

A	B	$\neg A$	$\neg B$	$A \vee B$	$\neg A \wedge \neg B$	$\neg(\neg A \wedge \neg B)$
t	*t*	*f*	*f*	*t*	*f*	*t*
t	*f*	*f*	*t*	*t*	*f*	*t*
f	*t*	*t*	*f*	*t*	*f*	*t*
f	*f*	*t*	*t*	*f*	*t*	*f*

- *We can provide similar demonstrations for the other connectives.*

□

- There are other combinations of connectives that are **functionally complete in this sense** (e.g. ¬ and ∨).
- It is even possible to find a *single* connective that is functionally complete!

The Sheffer Stroke

- The Sheffer Stroke is a logical connective written as | with the following meaning (i.e. truth function):

|	t	f
t	f	t
f	t	t

Proposition: *All of the connectives introduced as part of the language of the PC can be expressed in terms of the single connective* |.

Proof: *We must show that any sentence of the PC can be re-written as an equivalent sentence involving involving only the connective* |.

Note: we can simplify the proof by noting that we have already shown (in sketch) that the connectives can be expressed in terms of ¬ and ∧.

Proof continued:

There are two cases to consider:

1. $\neg A$ *is equivalent to* $A|A$

A	$\neg A$	$A\|A$
t	f	f
f	t	t

2. $A \wedge B$ *is equivalent to* $(A|B)|(A|B)$

A	B	$A \wedge B$	$A\|B$	$(A\|B)\|(A\|B)$
t	t	t	f	t
t	f	f	t	f
f	t	f	t	f
f	f	f	t	f

The Sheffer Stroke is functionally complete. □

So What?

- It is interesting to observe that logically speaking, the full set of connectives is not necessary.

  Question: Why is it that natural languages such as English have so many connectives when they could be more economical?

- The observation is also helpful when we want to prove things about the PC itself.
  - Note that this assumption helped to shorten the proof that the Sheffer Stroke was functionally complete.
  - Many other facts about the PC can be proved more easily using the same trick.

More on Logical Equivalence

- We have already noted that different sentences of the PC can be logically equivalent.
- For example, for any sentences A and B we showed (last lecture) that

  $$A \to B \equiv \neg A \vee B$$
- Here are some further examples:

  $$\begin{aligned} A \wedge B &\equiv B \wedge A \\ A \vee B &\equiv B \vee A \end{aligned}$$

  These are rather obvious. They show that $\wedge$ and $\vee$ are **commutative** operators.

  $$\begin{aligned} A \wedge (B \wedge C) &\equiv (A \wedge B) \wedge C \\ A \vee (B \vee C) &\equiv (A \vee B) \vee C \end{aligned}$$

  These equivalences show that $\wedge$ and $\vee$ are **associative** operators.

- More interesting are the following 'laws' of **distributivity**:

  $$\begin{aligned} A \wedge (B \vee C) &\equiv (A \wedge B) \vee (A \wedge C) \\ A \vee (B \wedge C) &\equiv (A \vee B) \wedge (A \vee C) \end{aligned}$$
- The following equivalences are known as **De Morgan's laws**:

  $$\begin{aligned} \neg(A \wedge B) &\equiv \neg A \vee \neg B \\ \neg(A \vee B) &\equiv \neg A \wedge \neg B \end{aligned}$$
- **Aside:** De Morgan (1806–1871) was a pioneer of the algebraic approach to logic.

- Let us introduce explicit symbols to represent *inconsistency* and *tautology*.
  - $\bot$ represents a proposition that is always **false** (inconsistency).
  - $\top$ represents a proposition that is always **true** (tautology).
- It is not difficult to see that:

  $$\begin{aligned} A \wedge \bot &\equiv \bot \\ A \wedge \top &\equiv A \\ A \vee \bot &\equiv A \\ A \vee \top &\equiv \top \\ A \wedge \neg A &\equiv \bot \\ A \vee \neg A &\equiv \top \end{aligned}$$
- There are many other equivalences (See: Kelly p.12 for a summary of some of the most important)

- The various logical equivalences provide a means of *simplifying* expressions.
- Consider for example:

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

  We can simplify this as follows:

  $$\begin{array}{lll} & (p \wedge \neg q) \vee (r \wedge (p \wedge q)) & \\ \equiv & (p \wedge \neg q) \vee ((p \wedge q) \wedge r) & \text{Commutativity} \\ \equiv & (p \wedge \neg q) \vee (p \wedge (q \wedge r)) & \text{Associativity} \\ \equiv & p \wedge (\neg q \vee (q \wedge r)) & \text{Distributivity} \\ \equiv & p \wedge ((\neg q \vee q) \wedge (\neg q \vee r)) & \text{Distributivity} \\ \equiv & p \wedge (\top \wedge (\neg q \vee r)) & \text{Tautology} \\ \equiv & p \wedge (\neg q \vee r) & \text{Identity of } \wedge \end{array}$$
- Note that the final line is also equivalent to:

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

Limitations of the Method of Truth Tables

- In principle, the method of truth tables can be applied to answer questions about:
  - the validity of arguments
  - consistency and inconsistency
  - logical equivalence
- There are *practical* limitations to this method however.

You may have noticed the following:

- for a sentence with 1 propositional variable, the truth table has two rows.
- for a sentence with 2 (distinct) propositional variables, the truth table has four rows.

Question: *In general, for a sentence with n distinct propositional variables, how many rows does its truth table have?*

- To appreciate what this means in practice, suppose that the sentence has 10 distinct propositional variables.
- In this case, the truth table has $2^{10} = 1024$ rows. (That's rather a lot for a person to work out, but we've got fast computers, right?)
- For a sentence with 50 distinct propositional variables, the number of rows is:

$$2^{50} = 1,125,899,906,842,624$$

- Calculating the truth table at the rate of one million rows per second, would still require **approximately 36 years** to complete the table.

Summary

- The stock of connectives we used to define the language of the PC are not strictly necessary.
- It is possible to find smaller sets of connectives that are functionally complete.
- The Sheffer Stroke is a single functionally complete connective.
- Logical equivalences can be used to simplify sentences of the PC.
- The method of truth tables has practical limitations which restrict its usefulness.

Introduction to Logic 5

Last time:

- Functional completeness
- The Sheffer Stroke
- Logical Equivalences and simplification
- Practical limitations of truth tables.

This time:

- Valuations
- Consistency/Inconsistency
- The Entailment relation
- Some Facts about entailment

Valuations

- A valuation is really just a function that assigns truth values to propositional variables.
  - If we use $\{t, f\}$ to model truth values; and
  - *Prop* is the set of propositional variables, then
  - $V : Prop \rightarrow \{t, f\}$ is a valuation.
- It is useful to extend the notion of a valuation to arbitrary sentences of the PC.
- Given a valuation V, we extend V to a new function V^* that assigns truth-values to all sentences of the PC (not just the propositional variables).
- $V^* : PC \rightarrow \{t, f\}$

  Note: The function V^* is also called a valuation (and confusingly, we may sometimes just write it as V).

Consistency and Inconsistency

- The language of PC can be used to represent sets of propositions.
- We may be interested in determining whether it is possible for every proposition in a given set to be true at the same time.

  Consider for example the following set G:

  $$G \quad = \{p, (\neg p \vee \neg q), (q \rightarrow p)\}$$

  Is there a valuation which makes every sentence in G true?

Definition: *A set $G = \{A_1, A_2, \ldots, A_k\}$ of sentences of the PC is said to be* **consistent** *if there exists some valuation V such that $V^*(A_i) = t$ for each sentences $A_i \in G$ $(1 \leq i \leq k)$. Otherwise G is said to be* **inconsistent**.

Testing Consistency

- We can use the method of truth tables to test whether a set of sentences is consistent.

  Consider: $\{p, (\neg p \vee \neg q), (q \rightarrow p)\}$

p	q	$\neg p$	$\neg q$	$(\neg p \vee \neg q)$	$(q \rightarrow p)$	
t	t	f	f	f	t	
t	f	f	t	**t**	**t**	⇐
f	t	t	f	t	f	
f	f	t	t	t	t	

- Note that the second row of the truth table has **t** in each column corresponding to one of the sentences in the set
- The set of sentences is *consistent* for any valuation V such that $V(p) = t$ and $V(q) = f$

- Consider the set of sentences:

  $$G = \{p, (p \rightarrow q), \neg q\}$$

p	q	$\neg q$	$(p \rightarrow q)$
t	t	f	t
t	f	t	f
f	t	f	t
f	f	t	t

- There is no row of the truth-table for which each sentence in G has the value **t**.
- The set of sentences G is *inconsistent*

Entailment

- *Entailment* is a relation that holds between a set of sentences G and a sentence A.
- Entailment is a *semantic* relation:

  i.e. it is defined with reference to the meaning of the sentences involved.
- Entailment captures a notion of *logical consequence*.

Definition: *A set of sentences G* **semantically entails** *a sentence A if and only if there is no valuation that makes all of the sentences in G* **true** , *but makes A* **false**

– i.e. assuming the truth of all the sentences in G has the consequence that A is **true** *as well.*

- We will introduce some special notation to stand for the entailment relation, and write:

$$G \models A$$

  to mean "G semantically entails A".
- We can think of $G \models A$ as formalizing the notion that given the **assumptions** in G, then the **conclusion** A is **true** , or A follows from the assumptions.

Note:

- The symbol $\models$ does **not** belong to the language of the PC.
- It belongs to our **meta-language** for talking about a relation between sentences and sets of sentences in our **object language** (the PC).

Example

- Consider the set of sentences:

$$G = \{p, (\neg p \vee \neg q), (q \rightarrow p)\}$$

  Then we have:

$$G \models \neg q$$

- To see this, note that (as we showed a little earlier by the method of truth tables) any valuation V which makes each sentence in G true is such that:

$$V(p) = \mathrm{t}$$
$$V(q) = \mathrm{f}$$

- But if $V(q) = \mathrm{f}$, then $V^*(\neg q) = \mathrm{t}$.
- So, assuming the truth of all the sentences in G has the consequence that $\neg q$ is **true** as well.

Some Facts about Entailment

Fact 1:

For any set of sentences G, if $A \in G$, then it must be the case that:

$$G \models A$$

e.g. if $G = \{(p \wedge q), \neg p\}$, then

$$\begin{aligned} G &\models (p \wedge q) \\ G &\models \neg p \end{aligned}$$

But note that $G \models A$ does *not* imply that $A \in G$.

Consider the previous example:

$$G = \{p, (\neg p \vee \neg q), (q \rightarrow p)\}$$

and

$$G \models \neg q$$

Fact 2: An inconsistency entails everything!

Consider a set of sentences G such that G is **inconsistent**. It follows that:

$$G \models A$$

for any sentence A

Proof: *Let G be an inconsistent set of sentences and A an arbitrary sentence. Suppose that A is* **not** *entailed by G. From the definition of entailment, there must exist a valuation that makes every sentence in G* **true** *, but which makes A* **false** *. But G is inconsistent, so no such evaluation can exist. It follows that $G \models A$.* □

Fact 3: Anything entails a tautology

Consider a **tautology** A. From the definition of entailment it follows that

$$G \models A$$

for any set of sentences G.

Proof: *Let A be a tautology and G an arbitrary set of sentences. Suppose that G does not entail A. From the definition of entailment, it follows that there must be a valuation which makes every sentence in G* **true** *, but that makes A* **false** *. But A is a tautology, so no such valuation can exist. It follows that $G \models A$.* □

Fact 4: Only a tautology follows from the empty set

Consider the case when G is the **empty** set of sentences $\{\}$. From the definition of entailment it must be that:

$$\text{if } \{\} \models A \textit{ then } A \textit{ is a tautology}$$

Proof: If G is the empty set, then there can be no valuation that makes a sentence in G **false** . In other words, every valuation makes all of the sentences in G **true** . So, if $G \models A$, then from the definition of entailment, every valuation must make A **true** as well. It follows that A is a tautology. □

We write $\models A$ to mean $\{\} \models A$.

Summary

- A valuation is a function from propositional variables to truth-values.
- A set of sentences is consistent if there exists a valuation which makes each sentence in the set **true**
- We can use the method of truth tables to establish the consistency or inconsistency of sets of sentences.
- Entailment is a semantic relation that holds between sentences and sets of sentences.
- The entailment relation captures a notion of logical consequence

Introduction to Logic 6

Last time:

- Valuations
- Consistency/Inconsistency
- The Entailment relation
- Some Facts about entailment

This time:

- Meaning and form
- Formal systems
- PC as a formal system
- Proof and Theorems
- Soundness and Completeness
- Decidability

Meaning and Form

- We have introduced a simple language for expressing propositions and sets of propositions.
- We have studied this language from the point of view of its meaning

  i.e.

  - Sentences are taken to denote truth-values
  - The connectives are truth-functions
  - We have looked at how the truth-value of a compound sentence is calculated from the meaning of its parts

- By investigating the meaning of our language we have found ways to:
  - classify sentences as tautologous, contingent or inconsistent
  - decide whether simple arguments are valid or not
  - decide whether two sentences are logically equivalent
  - determine the consistency/inconsistency of sets of sentences
  - formalize a notion of logical consequence between sentences and sets of sentences (entailment)
  - etc. etc. ...

- Studying a language from the point of view of its meaning seems natural.
- It is not the only way to proceed however.
  - We can examine the **form** of the sentences in our language rather than the **content**
  - We can provide rules for manipulating sentences in a purely formal (i.e. symbolic, syntactical) way.
  - We can devise techniques for determining consistency, inconsistency, equivalence, validity, etc., etc., that *do not depend on meaning or truth*.

- This all raises a couple of questions:

Question 1: Why study logic in this purely formal way?

Answer: Formal techniques are often more convenient, both for people and computers

- recall the limitations of the method of truth tables that we uncovered

Question 2: If it's all a matter of symbol manipulation, without regard to meaning at all, how do we know that it makes any sense?

Answer: Good question!

- Ultimately we have to *demonstrate* that the formal rules are sensible (and this *does* require reference to meaning and truth).

Logic as a Formal System

In general, a formal system is made up of

1. A language of some kind for making statements (expressing propositions)
2. A designated set of sentences called **axioms**
3. A set of rules for generating new sentences from old — the **rules of inference**.

In studying logic as a formal system we are interested in the notion of formal deduction or **proof**

- The **axioms** are sentences that we hold to be true in virtue of their form.
- The rules of inference allow us to prove **theorems**
- idea is that the formal notion of a **theorem** should coincide exactly with our previous semantic notion of a **tautology**.

Axiomatic Propositional Logic

- We can now view propositional logic as a formal system.
- One way is the following:

1. *The language of propositional logic*
2. *The following* **axiom schemas**:

   A1 $(A \to (B \to A))$

   A2 $((A \to (B \to C)) \to ((A \to B) \to (A \to C)))$

   A3 $(((\neg A) \to (\neg B)) \to (B \to A))$
3. **Modus Ponens**:

   from A and $(A \to B)$ infer B

Note:

- We have specified the axioms of the system using three **schemas**
  - Each schema must be *instantiated* to provide an axiom
  - There is an infinite number of instantiations of each schema!
- The axioms (axiom schemas) do not appear very natural.
  - it is hard to see where they come from
  - it is also hard to see how they might be used
- On the other hand, there is a single, and reasonably intuitive rule of inference

Proofs and Theorems

- Our intention is to provide a formal definition of our informal notion of **proof**.
- What is a proof?
  - Informally, we might say that a proof is a demonstration that some statement follows from some set of statements
  - A connected sequence of statements that go together to establish a conclusion
- Informal forms of proof often leave much of the structure or working *implicit*.

  Note that this includes mathematical proofs. While these are precise, many obvious steps (obvious to mathematicians!) are typically left out.
- To provide a *formal* notion of proof, we must make everything *explicit*.

Soundness and Completeness

- Of course, in the end we must show that our formal notion of proof makes sense.
- The formal notion of proof must be related back to our notion of logical consequence (the semantic relation $\models$)

Soundness: If there is a proof of a statement A (i.e. A is a theorem), then $\models A$ (i.e. A is a tautology).

Completeness: If $\models A$ (A is a tautology), then it must be possible to prove that A (i.e. A is a theorem).

- Only if our formal system is both **sound** and **complete** can we regard it as adequate.

Note:

- Soundness and completeness is something that we must *prove* about a formal system of logic. We cannot just take it for granted.
  - Proving that a formal system is sound is generally quite straightforward.
  - Proving completeness can be very tricky.
- Having said this, we will not actually attempt to prove soundness and completeness for axiomatic propositional logic.
- In fact, the system is both sound and complete (see e.g. Kelly chapter 4, section 5 for proofs).

Decidability

- A further property of a formal system of logic of interest to us is **decidability**.
- A formal system of logic is **decidable** if there exists an **effective procedure** for determining whether or not an arbitrary statement A is a theorem; i.e.:
  - If A is a theorem the procedure should halt and answer **yes**
  - If A is not a theorem the procedure should halt and answer **no**

Proposition: *Axiomatic propositional logic is decidable.*

Proof: *(Sketch) A is a theorem if and only if it is a tautology (soundness and completeness).*

We can check whether A is a tautology in a purely mechanical way (e.g. by constructing its truth table). □

Summary

- We can study logic according to the meaning or content of statement.
- An alternative is to examine the form of statements.
- A formal system of logic has axioms and inference rules.
- The aim is to formalize a notion of *proof*.
- To be adequate, a formal system of logic must be both *sound* and *complete* – axiomatic propositional logic is adequate in this sense.
- A useful property of a formal system of logic is *decidability* — axiomatic propositional logic is decidable.

Introduction to Logic 7

Last time:

- Meaning and Form
- Formal Systems
- PC as a Formal System
- Proof and Truth
- Decidability

This time:

- PC as an Axiomatic System
- Formal Proofs
- The Deduction Relation
- Deduction and Entailment

Propositional Logic as an Axiomatic System

- *The language of Propositional Logic*
- *The following axiom schemas:*

S1: $(A \to (B \to A))$

S2: $((A \to (B \to C)) \to ((A \to B) \to (A \to C)))$

S3: $(((\neg A) \to (\neg B)) \to (B \to A))$

- *The following rule of inference:*

From A and $(A \to B)$ deduce B

- B is called a *direct consequence* of A and $(A \to B)$
- The rule is known as **Modus Ponens** (**MP** for short)

- Note once again that the number of axioms is infinite.
- There is an infinite number of *instances* of the axiom schemas S1, S2 and S3.

Intuitively, and instance of an axiom is a sentence of propositional logic formed by *instantiating* the meta-language variables in a schema.

Example:

$(A \to (B \to A))$ – (schema S1)

can be instantiated as:

$((p \to q) \to ((\neg q) \to (p \to q))$

where:

A is instantiated as $(p \to q)$

B is instantiated as $(\neg q)$

- Each instantiation of S1, S2 or S3 is an axiom of the formal system of Propositional Logic.

e.g. The following sentences are all axioms:

Inst S1: $(p \rightarrow (q \rightarrow p))$

Inst S2: $((p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$

Inst S3: $((\neg p) \rightarrow (\neg q)) \rightarrow (q \rightarrow p))$

Inst S1: $((p \wedge q) \rightarrow (r \rightarrow (p \wedge q)))$

Inst S3: $((\neg(p \vee q) \rightarrow \neg r) \rightarrow (r \rightarrow (p \vee q)))$

etc., etc. …

Formal Proofs and Theorems

- We are interested in formalizing the notion of **proof**

- We can now define a notion of *proof within a formal system* as follows:

Defintion: (Proof) A proof in a formal system is a sequence of sentences

$$A_1, A_2, \ldots, A_n$$

where each A_i $(1 \leq i \leq n)$ is either:

1. an *axiom*; or
2. a *direct consequence* of two earlier sentences A_j and A_k $(j, k < i)$

Definition: (Theorem) If a sequence of sentences $A_1, A_2, \ldots, A_n$ is a proof in a formal system, then the sentence A_n is called a *theorem* of that system.

Example:

- Proof that $(p \rightarrow p)$ is a theorem of the formal system of propositional logic.

(1) $((p \rightarrow ((p \rightarrow p) \rightarrow p)) \rightarrow ((p \rightarrow (p \rightarrow p)) \rightarrow (p \rightarrow p)))$ – Inst S2

(2) $(p \rightarrow ((p \rightarrow p) \rightarrow p))$ – Inst S1

(3) $((p \rightarrow (p \rightarrow p)) \rightarrow (p \rightarrow p))$ – MP on (1) & (2)

(4) $(p \rightarrow (p \rightarrow p)$ – Inst S1

(5) $(p \rightarrow p)$ – MP on (3) & (4)

- So $(p \rightarrow p)$ is a theorem.

Note:

- Proofs give us a way of generating new theorems from a given stock of 'old' theorems (i.e. the axioms).
- In general, if

$$A_1, A_2, \ldots A_{n-1}, A_n$$

is a proof, then so is

$$A_1, A_2, \ldots A_{n-1}$$

Question: Why and what does this imply about A_{n-1}?

Deduction

- We may be interested in finding out what follows from an arbitrary stock of sentences (i.e. not just from the axioms).
- We formalize a notion of a **deduction** (in a formal system) as follows:

Definition: (Deduction) Let G be an arbitrary set of sentences. A sequence of sentences

$$A_1, A_2, \ldots, A_n$$

is a **deduction from** G if each sentence A_i $(1 \leq i \leq n)$ is either:

1. an *axiom*; or
2. a *sentence in* G; or
3. a *direct consequence* from two earlier members of the sequence

Note:

1. a deduction from a set G is just like a proof, except that the members of the sequence $A_1, A_2, \ldots, A_n$ can also be drawn from G.
   - The elements of G are like temporary axioms.
2. Also, if a sequence of sentences:

$$A_1, A_2, \ldots, A_n$$

is a deduction from a set G, then the sentence A_n will *not*, in general, be a theorem.

- We say that A_n is **deducible from** G and this is written:

$$G \vdash A_n$$

Question: What can we say about A_n if G is the empty set?

Example

- We shall show that:

$$\{p, (q \rightarrow (p \rightarrow r))\} \vdash (q \rightarrow r)$$

(1) p	– Assumption
(2) $(q \rightarrow (p \rightarrow r))$	– Assumption
(3) $(p \rightarrow (q \rightarrow p))$	– Inst S1
(4) $(q \rightarrow p)$	MP on (1) & (3)
(5) $((q \rightarrow (p \rightarrow r)) \rightarrow ((q \rightarrow p) \rightarrow (q \rightarrow r)))$	– Inst S2
(6) $((q \rightarrow p) \rightarrow (q \rightarrow r))$	– MP on (2) & (5)
(7) $(q \rightarrow r)$	– MP on (4) & (6)

So: $(q \rightarrow r)$ is deducible from $\{p, (q \rightarrow (p \rightarrow r))\}$

Deduction and Entailment

- You may have observed some similarities between notion of the *deduction relation* ($\vdash$) and the notion of *entailment* ($\models$).
  - both relations are defined to hold between a set of sentences G and a sentence A;
  - both attempt to capture a notion of ‘consequence’
- We may suspect that the two relations will actually turn out to be identical.
  i.e.

$$G \models A \text{ if and only if } G \vdash A$$

Summary

- Propositional logic can be formalized as an axiomatic system.
- We can define a notion of formal proof within such a system.
- Proofs establish that certain sentences are theorems of the system.
- More generally, we have the notion of a deduction from a set of statements or assumptions.
- Deduction captures the idea of a statement being consequent on some set of assumptions.
- Deduction and entailment have striking similarities, even though they are defined in very different ways.

Introduction to Logic 8

Last time:

- PC as an Axiomatic System
- Formal Proofs
- The Deduction Relation
- Deduction and Entailment

This time:

- Consistency and Inconsistency
- Semantic Tableau
- The Tableaux Technique
- Tableaux Derivation Rules

Consistency and Inconsistency

- Recall that a set of sentences G is *consistent* if there is at least one valuation that makes every sentence in G true (and otherwise G is *inconsistent*).
- We can test consistency/inconsistency using the method of truth tables:

  e.g.

$$G = \{(p \wedge q), (p \rightarrow \neg q)\}$$

p	q	$(p \wedge q)$	$\neg q$	$(p \rightarrow \neg q)$
t	t	t	f	f
t	f	f	t	t
f	t	f	f	t
f	f	f	t	t

Thus G is inconsistent!

Semantic Tableaux

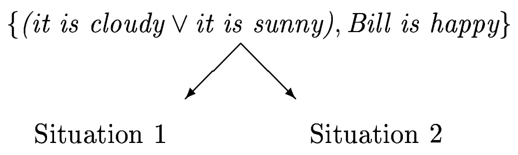
- There are more effective ways of testing for consistency/inconsistency.
- The method of semantic tableaux provides a means of testing *inconsistency* of sets of sentences.
- Semantic tableaux are more expressive and in some ways easier to use than truth tables
- Can also be used to test *entailement* :

  Is it the case that $G \models A$?
- Based on the idea of generating *descriptions* of situations.

The Tableaux Technique

- Consider a set of sentences G.
  - We can think of G as describing different possible situations....
  - ....those situations which make every sentence of G true.

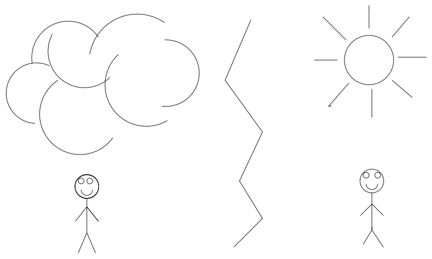
i.e.



{*(it is cloudy* ∨ *it is sunny)*, *Bill is happy*}

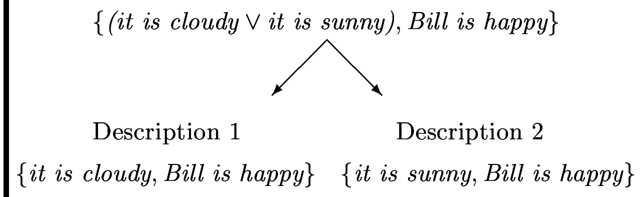
Situation 1

Situation 2



- Semantic tableaux provide a systematic method for finding what possible situations are described by a set of sentences G.
  - We use G to produce new descriptions of the situations....
  - ....the new descriptions are obtained by *simplifying* the complex sentences in G.

e.g.



{*(it is cloudy* ∨ *it is sunny)*, *Bill is happy*}

Description 1

{*it is cloudy*, *Bill is happy*}

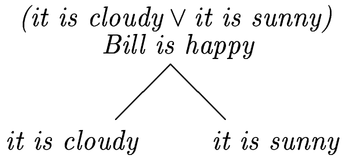
Description 2

{*it is sunny*, *Bill is happy*}

Tableaux and Inconsistency

- The tableaux method has various **tableaux derivation rules** that allow us to construct a ‘picture’ of all the different possible descriptions.
- This picture is a tree diagram (the **tableau**).

  e.g.



(it is cloudy ∨ *it is sunny)*

Bill is happy

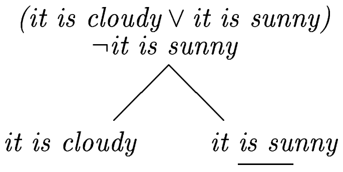
it is cloudy

it is sunny

- This tableau has two branches, where each branch represents a situation.

- Sometimes, branches *fail* to represent a possible situation.

  e.g.



(it is cloudy ∨ *it is sunny)*

¬*it is sunny*

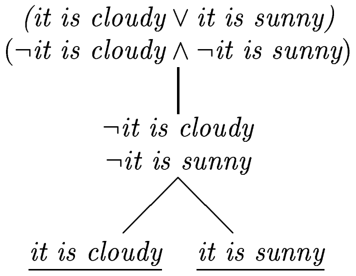
it is cloudy

it is sunny

- This tableau has two branches.
  - One branch fails to describe a situation – it contains inconsistent information.
  - The branch is said to be **closed**
- We write a line under the branch to show that it is closed.

- In general, whenever a branch contains a statement A and a statement $\neg A$, then it contains inconsistent information, and is said to be *closed*.
- Closed branches are not extended further.
- If *all* the branches of a tableau are closed, then we have shown that the set of statements we started from is inconsistent.

e.g.



$(\textit{it is cloudy} \vee \textit{it is sunny})$

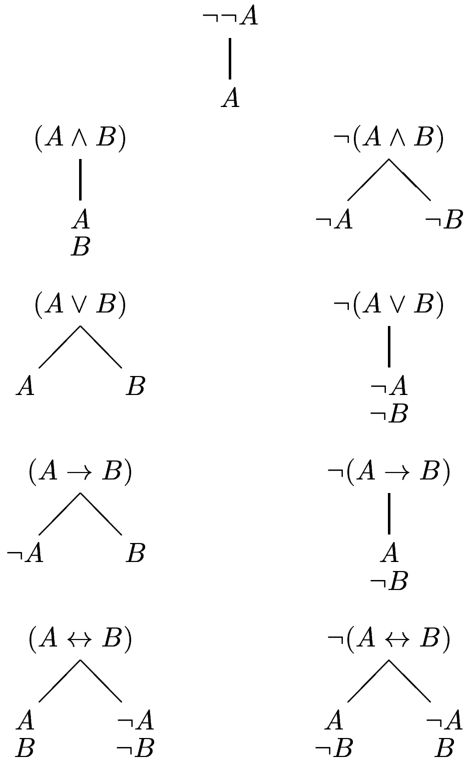
$(\neg \textit{it is cloudy} \wedge \neg \textit{it is sunny})$

$\neg \textit{it is cloudy}$

$\neg \textit{it is sunny}$

$\underline{\textit{it is cloudy}}$ $\underline{\textit{it is sunny}}$

Tableaux Rules



$\neg\neg A$ / A

$(A \wedge B)$ / A, B

$\neg(A \wedge B)$ / $\neg A$ | $\neg B$

$(A \vee B)$ / A | B

$\neg(A \vee B)$ / $\neg A$, $\neg B$

$(A \to B)$ / $\neg A$ | B

$\neg(A \to B)$ / A, $\neg B$

$(A \leftrightarrow B)$ / A, B | $\neg A$, $\neg B$

$\neg(A \leftrightarrow B)$ / A, $\neg B$ | $\neg A$, B

Example

- Is the set

$$\{\neg(p \wedge \neg q), (q \to r), (p \wedge \neg r)\}$$

consistent or inconsistent?

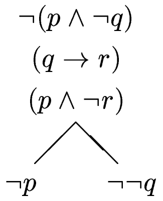
1.

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

$(q \to r)$

$(p \wedge \neg r)$

2.



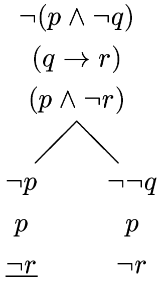
$\neg(p \wedge \neg q)$

$(q \to r)$

$(p \wedge \neg r)$

$\neg p$ $\neg\neg q$

3.



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

$(q \to r)$

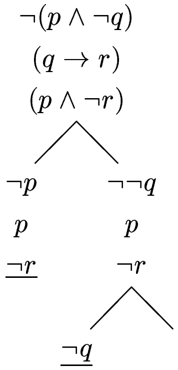
$(p \wedge \neg r)$

$\neg p$ $\neg\neg q$

p p

$\underline{\neg r}$ $\neg r$

4.



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

$(q \to r)$

$(p \wedge \neg r)$

$\neg p$ $\neg\neg q$

p p

$\underline{\neg r}$ $\neg r$

$\underline{\neg q}$ $\underline{r}$

- Each branch or the tableau is closed.

- Because each branch of the tableau closed, we say that *the tableau is closed*.
- This means that every branch of the tableau contains contradictory information.
  - we cannot find a valuation that will make every sentence on a given branch true.
  - there is no valuation that makes every sentence in the original set true.
- It follows that the set

  $$\{\neg(p \wedge \neg q), (q \rightarrow r), (p \wedge \neg r)\}$$

  is inconsistent.

Summary

- Semantic tableaux provide a technique for testing consistency/inconsistency of sets of sentences
- Tableaux are more expressive, and easier to use than truth tables
- The method is based on the idea of simplifying descriptions/sentences and looking for contradictions.
- The tableaux derivation rules allow us to grow a tree diagram representing possible situations.
- In contrast to the axiomatic system of propositional logic, the tableaux proof method is simple and straightforward to use.

Introduction to Logic 9

Last time:

- Consistency and Inconsistency
- Semantic Tableaux
- The Tableaux Technique
- Tableaux Derivation Rules

This time:

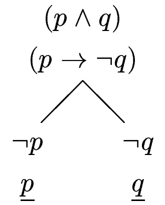
- Tableaux Examples
- Satisfying Valuations
- Justification for the Tableaux Rules
- Inconsistency and Entailment
- Bacon and Hamlet (Again)

Semantic Tableaux Examples

- Semantic Tableaux enable us to check consistency/inconsistency of sets of sentences.

  e.g.

  $G = \{(p \wedge q), (p \rightarrow \neg q)\}$
- Construct a tableau as follows:



$(p \wedge q)$

$(p \rightarrow \neg q)$

$\neg p$ $\quad$ $\neg q$

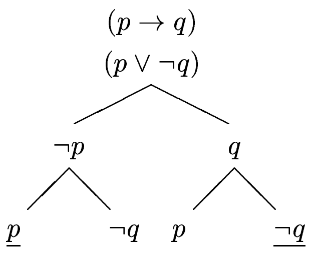
$\underline{p}$ $\quad$ $\underline{q}$

- Both branches are closed, so G is inconsistent!
- The method typically requires less effort than the method of truthtables (see start of last lecture for comparison).

- Is the following set of sentences inconsistent?

$$G = \{(p \to q), (p \vee \neg q)\}$$

- Construct a tableau as follows:



$(p \to q)$

$(p \vee \neg q)$

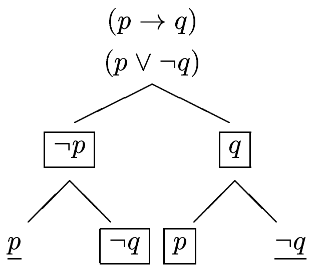
$\neg p$ q

$\underline{p}$ $\neg q$ p $\underline{\neg q}$

- The tableau is 'finished', but it is not closed.
- Two branches remain open: the set G is *consistent*.

Definition: Let G be a set of sentences and V a valuation. We say that V *satisfies* G if and only if V makes every sentence in the set G true.

- We may want to know what valuations satisfy a consistent set G.
- This information can be found from a tableau for G. For example:



$(p \to q)$

$(p \vee \neg q)$

$\boxed{\neg p}$ $\boxed{q}$

$\underline{p}$ $\boxed{\neg q}$ $\boxed{p}$ $\underline{\neg q}$

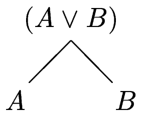
Question: What can we say about valuations that satisfy this set?

Justifying Tableaux Rules

- We can view the tableaux rules *syntactically*.
- We can also view them *semantically*.

  i.e. we can *interpret* the rules and show that they are sensible.
- Tableaux rules can be justified/motivated straightforwardly by considering truth tables.

  e.g.



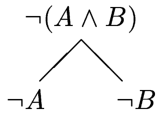
$(A \vee B)$

A B

A	B	$(A \vee B)$
t	t	t
t	f	t
f	t	t
f	f	f

- Note that there are just two sorts of 'situations' in which $(A \vee B)$ is true:
  1. situations where A is true
  2. situations where B is true

- Consider now the tableau rule for $\neg(A \wedge B)$:



$\neg(A \wedge B)$

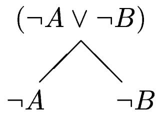
$\neg A$ $\neg B$

- Recall the following equivalence:

$$\neg(A \wedge B) \equiv (\neg A \vee \neg B)$$

(this is one of De Morgan's equivalences – see lecture 4).

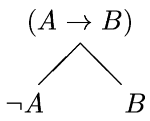
- So, using the tableau rule for disjunction, we can justify the rule by noting that:



$(\neg A \vee \neg B)$

$\neg A$ $\neg B$

- Similarly, we can justify the rule for $(A \to B)$:



- In this case we can make use of the following logical equivalence:

$$(A \to B) \equiv (\neg A \lor B)$$

(easy to check with truthtables; also given in lecture 4.)

- We can provide a justification for each of the derivation rules of the semantic tableaux method.
- This effectively shows that the method is **sound**

Inconsistency and Entailment

- The tableaux method allows us to test consistency/inconsistency of sets of sentences
- This may seem rather limiting, but it was claimed in the previous lecture that the method can also be used for testing entailment.

Question: *how do we use semantic tableaux to test for entailment?*

The answer to this can be found in the definition of entailment.

- Recall the definition:

  $G \models A$ *if and only if every valuation that makes each sentence in* G **true** *also makes* A **true** .

- or to put it another (and equivalent) way.....

$G \models A$ *if and only if every valuation that makes each sentence in* G **true** *also makes* $\neg A$ **false** .

- and what this comes down to is...

  $G \models A$ *if and only if the set of sentences* $G \cup \{\neg A\}$ *is inconsistent.*

- But we can use semantic tableaux to test consistency/inconsistency.
- So we can use semantic tableaux to test entailment.
- To test whether $G \models A$, we:
  1. form the set $G \cup \{\neg A\}$; and
  2. use tableaux to determine if the set is inconsistent (entailment holds) or consistent (entailment does not hold).

Example (Bacon and Hamlet (Again))

- Consider the following argument:

  If Bacon wrote Hamlet, then Bacon was a great writer. But Bacon did not write Hamlet. So Bacon was not a great writer.

- We can formalize the premisses and the conclusion of the argument as follows:

Premise 1	$(p \to q)$
Premise 2	$\neg p$
Conclusion	$\neg q$

- Moreover, this argument will be correct (valid, sound) just in case the following entailment holds:

$$\{(p \to q), \neg p\} \models \neg q$$

- We will test this entailment using the semantic tableaux method.

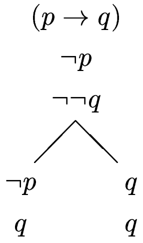
- To test whether

$$\{(p \to q), \neg p\} \models \neg q$$

we test consistency of the set:

$$\{(p \to q), \neg p, \neg\neg q\}$$

- Applying the tableau method yields:



- The tableau is ‘finished’, but not closed.
- It follows that the set is *consistent* ... so entailment does *not* hold ... and the argument is *not* valid.

Summary

- Semantic tableaux provide a convenient and systematic technique for testing consistency/inconsistency of sets of sentences
- Tableaux can be used to find the valuations that *satisfy* a set of statements.
- Tableax derivation rules can be given a semantic justification
- There is a close connection between the notions of *inconsistency* and *entailment*.
- This provides the basis for testing entailment using the method of semantic tableaux.

Introduction to Logic 10

Last time:

- Tableaux and Valuations
- Justifying the Tableaux Rules
- Inconsistency and Entailment
- Testing Validity of Arguments

This time:

- Un-natural Deduction
- Natural Deduction
- Introduction Rules
- Examples

Un-natural Deduction

- We have seen how logic can be viewed as a formal system of deduction consisting of:
  1. a *language* for expressing propositions;
  2. a set of *axioms*;
  3. a set of *rules of inference*
- We can furnish a precise definition of the notion of a proof (in a formal system):

Defintion: (Proof) A proof in a formal system is a sequence of sentences

$$A_1, A_2, \ldots, A_n$$

where each A_i $(1 \leq i \leq n)$ is either:

1. an *axiom*; or
2. a *direct consequence* of two earlier sentences A_j and A_k $(j, k < i)$

For example, in Lectures 6 and 7 we saw how the Propositional Calculus could be formalized as an axiomatic system.

- This systems had three axiom schemas and a single rule of inference (Modus Ponens);
- We have noted that proofs constructed within this system are not particularly 'natural':
  - They are hard to construct;
  - The use of axioms is not intuitive
  - The individual proof steps do not appear to correspond to steps in 'informal' proofs or argumentation;

Is it possible to formulate some system of deduction that is more 'natural' than this?

- The method of Semantic Tableaux has some merit:
  - it is easier to use than the axiomatic systems (i.e. constructing tableaux is a relatively straightforward, rule-governed process);
  - the tableaux derivation rules have a straightforward semantic interpretation;
- In other ways however, the method is not as 'natural' as we might like:
  - the use of tableaux to establish *inconsistency* is not particularly intuitive;
  - the individual derivation rules do not correspond well to steps in informal proofs or reasoning.

Natural Deduction

- People seem to use a variety of methods for constructing informal arguments or proofs in natural language.
- Even mathematicians do not generally proceed from axiom systems of the kind we have seen for the Propositional Calculus.
- Informal proofs exhibit 'patterns of reasoning' like the following:

$$\frac{\textit{if Logic is fun, then Bill is happy} \qquad \textit{Logic is fun}}{\textit{Bill is happy}}$$

- This instance of Modus Ponens seem quite natural.

Could we formulate a system of deduction based *entirely* on 'natural laws' such as the above?

- The new formal system of Natural Deduction will consist of the following components:
  1. The language of Propositional Logic
  2. Various rules of inference:
     - Introduction rules;
     - Elimination rules;
- Note that in constrast to the axiomatic system that we saw earlier, this formal system has *no* axioms.
- Also, rather than a single rule of inference (Modus Ponens) it has *many* such rules.
- The natural deduction rules are intended to express frequently used patterns of reasoning.
- The rules come in two varieties:
  - rules that produce complex statements from smaller statements by *introducing* connectives; and
  - rules that produce simpler statements from complex statements by *eliminating* connectives.

Introduction Rules

- The introduction rules are so-called because they are used to *introduce* connectives.

Conjunction Introduction ($\wedge$I)

- This rule captures the following informal 'pattern of reasoning':

> *If you know that A is* **true** *and that B is* **true** *, then it is valid to conclude that $(A \wedge B)$ is* **true** *.*

- In the system of Natural Deduction, this rule is represented diagrammatically as follows:

$$\frac{A \quad B}{(A \wedge B)}\ \wedge I$$

- The rule has two premisses A and B, and produces a conclusion $(A \wedge B)$, that has $\wedge$ as its principal connective.

Disjunction Introduction ($\vee$I)

- This rule corresponds to the following informal pattern of reasoning:

> *If you know that A is* **true** *, then you can conclue that $(A \vee B)$ is* **true** *(for* any *sentence B).*

- This rule of disjunction introduction actually corresponds to two rules of inference in the system of Natural Deduction:

$$\frac{A}{(A \vee B)}\ \vee I \qquad\qquad \frac{B}{(A \vee B)}\ \vee I$$

- The introduction rules for the connectives $\wedge$ and $\vee$ may seem rather trivial.

- a more interesting rule is **Implication Introduction**: the so-called **Method of Conditional Proof**.

Implication Introduction ($\rightarrow$ I):

- This is a method for introducing the conditional or implication connective $\rightarrow$.

- the method of conditional proof corresponds to the following line of argumentation:

> *Under the assumption that statement A is* **true** *, it is possible to reason to the conclusion that statement B is* **true** *.*
>
> *As the conclusion B rests on the assumption A, it is valid to conclude that $(A \rightarrow B)$ is* **true** *.*

- It is a little harder to represent this rule diagrammatically

- The rule of **implication introduction** does not correspond neatly to a single proof step.

- The reasoning in conditional proof concerns the overall structure of (part of) a proof.

- Implication introduction ($\rightarrow$ I) is represented as follows:

$$\begin{array}{c} \cancel{A} \\ \vdots \\ B \\ \hline (A \rightarrow B) \end{array}\ \rightarrow I$$

Note:

- The intermediate conclusion B rests on the assumption A. However, the final conclusion $(A \rightarrow B)$ does *not* rest on A!

- The assumption A must be *cancelled* or *discharged* once we draw the final conclusion $(A \rightarrow B)$.

- We cross out the assumption ($\cancel{A}$) to remind ourselves that $(A \rightarrow B)$ does not depend on A.

- We now have rules for introducing the connectives $\land$, $\lor$ and $\to$.
- Note that we have not provided introduction rules for $\neg$ and $\leftrightarrow$:
  - treatment of $\neg$ will be deferred until next lecture;
  - we will not consider $\leftrightarrow$ since, e.g.,

$$(A \leftrightarrow B) \equiv (A \to B) \land (B \to A)$$

• The final rule included here is simply called $\bot$ (*falsum*).

Falsum ($\bot$):

• In essense this rule states:

> *Anything follows from falsum (i.e. from an absurdity or contradiction).*

• The rule is notated as:

$$\dfrac{\bot}{A}\ \bot$$

Example

• Using just the introduction rules of the system of natural deduction, we will show that:

$$\vdash (p \to (p \lor q))$$

• The proof proceeds as follows:

$$\dfrac{\dfrac{\cancel{p}}{(p \lor q)}\ \lor I}{(p \to (p \lor q))}\ \to I$$

• Note that the assumption p has been cancelled.

• Thus the conclusion $(p \to (p \lor q))$ does not rest on any assumptions.

• This means that $(p \to (p \lor q))$ is a theorem.

Example

• We will show that

$$\{p\} \vdash (r \to ((p \lor q) \land r))$$

• The proof proceeds as follows:

$$\dfrac{\dfrac{\dfrac{p}{(p \lor q)}\ \lor I \qquad \cancel{r}}{((p \lor q) \land r)}\ \land I}{(r \to ((p \lor q) \land r))}\ \to I$$

• Note that in this case, only the assumption r has been cancelled.

• The proof still contains an undischarged assumption p.

• This means that the final statement rests on the assumption p (though not on r).

Summary

- The axiomatic system of Propositional Logic is not particularly intuitive or ‘natural’
- The method of Semantic Tableaux is more easy to apply, but does not correspond well with informal methods of proof or argumentation
- The System of Natural Deduction is an attempt to formalize reasoning in a way that captures commonly used ‘patterms of reasoning’.
  - There are no axioms...
  - ...but many rules of inference
- The inference rules fall into two groups: Introduction Rules and Elimination Rules.

Introduction to Logic 11

Last time:

- Un-natural Deduction
- Natural Deduction
- Introduction Rules
- Examples

This time:

- Natural Deduction Proof Rules
- Introduction Rules
- Elimination Rules
- Proof by Contradiction

Natural Deduction Proof Rules

- The system of Natural Deduction is a proof method that has some advantages over the axiomatic system and the tableaux method for propositional logic.
  - proofs are relatively easy to construct;
  - the proofs that result consist of a fairly natural sequence of steps
- The Natural Deduction inference rules attempt to capture frequently used patterns of reasoning or ‘logical laws’.
- Broadly, the rules fall into two groups:
  1. **Introduction Rules:** i.e. rules that *introduce* connectives;
  2. **Elimination Rules:** i.e. rules that *eliminate* connectives.

Introduction Rules

$$\frac{A \quad B}{(A \wedge B)}\ \wedge I$$

$$\frac{A}{(A \vee B)}\ \vee I \qquad\qquad \frac{B}{(A \vee B)}\ \vee I$$

$$\begin{array}{c} \cancel{A} \\ \vdots \\ \dfrac{B}{(A \rightarrow B)}\ \rightarrow I \end{array}$$

$$\frac{\bot}{A}\ \bot$$

Note:

$$\begin{aligned} (A \leftrightarrow B) &\equiv ((A \rightarrow B) \wedge (B \rightarrow A)) \\ \neg A &\equiv (A \rightarrow \bot) \end{aligned}$$

Elimination Rules

• Let us consider now the rules for eliminating connectives.

Conjunction Elimination (∧ E):

• Consider the following pattern of reasoning:

> *Suppose that you know that $(A \wedge B)$ is* **true** *, then it is safe to infer that A (or B) must be true.*

• Expressing this is the notation of the system of Natural Deduction, gives the following *two* rules of inference:

$$\frac{(A \wedge B)}{A}\ \wedge E \qquad\qquad \frac{(A \wedge B)}{B}\ \wedge E$$

Implication Elimination (→ E):

- Consider the following pattern of reasoning

> *Suppose you know that $(A \to B)$ is* **true** *and also that A is* **true** *. In this case, it is safe to infer that B is* **true** *.*

- In the system of natural deduction, this may be notated as:

$$\frac{(A \to B) \quad A}{B} \to E$$

Question: *Where have we seen this rule before?*

Example: $\{(p \to q), (q \to r)\} \vdash (p \to r)$

$$\cfrac{\cfrac{\cfrac{(p \to q) \quad \cancel{p}}{q} \to E \qquad (q \to r)}{r} \to E}{(p \to r)} \to I$$

Disjunction Elimination(∨ E)

- The rule for eliminating a disjunction (∨) is a little trickier to understand.

> *Suppose you know that $(A \lor B)$ is* **true** *. Suppose also that from the assumption that A is* **true** *you can reach a conclusion that C is* **true** *; and from the assumption that B is* **true** *, you can reach that <u>s</u>ame conclusion, that C is* **true**
>
> *In this case, it is safe to infer that C is* **true** *.*

- The rule is essentially that of analysis by cases:
  - whichever case we consider (A or B) we can show that C must be **true** ;
  - so we can conclude that C follows from $(A \lor B)$
- Like implication introduction, this rule is not straightforward to represent.

- Diagrammatically, the rule of Disjunction Elimination appears as follows:

$$\frac{(A \lor B) \qquad \begin{matrix}\cancel{A} \\ \vdots \\ C\end{matrix} \quad \begin{matrix}\cancel{B} \\ \vdots \\ C\end{matrix}}{C} \lor E$$

- Here is an example of its use:

$$\cfrac{((p \land q) \lor q) \quad \cfrac{\cfrac{\cancel{(p \land q)}}{p} \land E}{(p \lor q)} \lor I \quad \cfrac{\cancel{q}}{(p \lor q)} \lor I}{(p \lor q)} \lor E$$

- So:

$$\{((p \land q) \lor q)\} \vdash (p \lor q)$$

Proof By Contradiction (reductio ad absurdum)

- We now have introduction and elimination rules for each of the binary connectives: ∧, ∨ and →.
- We have not yet considered negation: ¬.
- Consider the following method of reasoning:

> *Suppose that we wish to prove that some statement A holds. Assume rather that $\neg A$ holds. If we can now show that this assumption leads to a contradiction, then it is safe to conclude that $\neg A$ cannot hold. In other words, A must hold.*

- This proof method is know as **Proof by Contradiction**, or **reductio ad absurdum (RAA)**.

- As a diagram, this proof rule **RAA** may be represented as follows:

$$\begin{array}{c} \neg\cancel{A} \\ \vdots \\ \dfrac{\bot}{A}\ RAA \end{array}$$

- The following example illustrates the use of RAA. We show:

$$\{\neg(\neg p \vee q)\} \vdash p$$

$$\dfrac{\dfrac{\dfrac{\neg\cancel{p}}{(\neg p \vee q)}\vee I \qquad \neg(\neg p \vee q)}{\bot}\to E}{p}\ RAA$$

- **NB:** *This proof also makes use of the fact that in this system,* $\neg A$ *is simply an abbreviation for* $(A \to \bot)$.

Elimination Rules

$$\dfrac{(A \wedge B)}{A}\wedge E \qquad\qquad \dfrac{(A \wedge B)}{B}\wedge E$$

$$\dfrac{(A \to B) \quad A}{B}\to E$$

$$\dfrac{(A \vee B) \quad \begin{array}{c}\cancel{A}\\ \vdots \\ C\end{array} \quad \begin{array}{c}\cancel{B}\\ \vdots \\ C\end{array}}{C}\vee E$$

$$\begin{array}{c} \neg\cancel{A} \\ \vdots \\ \dfrac{\bot}{A}\ RAA \end{array}$$

Example

- We show that

$$\vdash (p \vee \neg p)$$

- The proof proceeds as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\neg\cancel{p}^{(1)}}{(p \vee \neg p)}\vee I \qquad \neg(p \not\vee \neg p)^{(2)}}{\bot}\to E}{p}\ RAA}{(p \vee \neg p)}\vee I \qquad \neg(p \not\vee \neg p)^{(2)}}{\bot}\to E}{(p \vee \neg p)}\ RAA$$

Remarks

- There are just *two* assumption introduced in this proof
- In the end, the proof is perhaps not quite as ‘natural’ as we would like!

Summary

- The system of Natural Deduction has introduction and elimination rules for connectives.
- Elimination rules for conjunction and implication are straightforward. Implication elimination, in particular, is familiar as the rule Modus Ponens.
- The elimination rule for disjunction corresponds to a method of ‘reasoning by cases’
- The system also has a rule formalizing the famous *proof by contradiction* or *reductio ad absurdum*.

Introduction to Logic 12

Last time:

- Natural Deduction Proof Rules
- Introduction Rules
- Elimination Rules
- Proof by Contradiction

This time:

- Propositional Logic
- Limitations of Propositional Logic
- The Structure of Propositions
- A New Logical Notation

Propositional Logic (The story so far)

- We have introduced the language of propositional logic as a means of representing propositions and arguments.

  e.g.

  If $x > 3$, then $y < 4$. But $y \not< 4$, so $x \not> 3$.

- This can be represented as:

$$((p \to q) \land \neg q) \to \neg p$$

where:

p stands for '$x > 3$'; and

q stands for '$y < 4$'

- We have provided a precise notion of meaning for statements of propositional logic:

p	q	$\neg p$	$\neg q$	$p \to q$	$(p \to q) \land \neg q$	$((p \to q) \land \neg q) \to \neg p$
t	t	f	f	t	f	t
t	f	f	t	f	f	t
f	t	t	f	t	f	t
f	f	t	t	t	t	t

- This allows us to distinguish between statements that are **tautologies**, **contingencies** and **inconsistencies**.
- We can also use truth-tables to determine whether arguments are valid/invalid.

  e.g.

$$((p \to q) \land \neg q) \to \neg p$$

This is a tautology, so the argument is *valid*.

- The relation of **semantic entailment** ($\models$) captures a notion of logical consequence between propositions.

  e.g.

$$\{(p \to q), \neg q\} \models \neg p$$

The statement $\neg p$ is a consequence of the set of statements $\{(p \to q), \neg q\}$

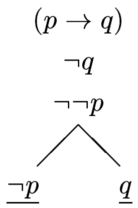
i.e. Any valuation that makes both $p \to q$ and $\neg q$ **true** , also makes $\neg p$ **true** .

p	q	$\neg p$	$\neg q$	$p \to q$
t	t	f	f	t
t	f	f	t	f
f	t	t	f	t
f	f	**t**	**t**	**t**

- We have looked at purely *formal* techniques for ‘calculating’ with (sets of) statements.
  - The classical axiomatic presentation – logic as a formal, deductive system;
  - The method of Semantic Tableaux;
  - The system of Natural Deduction.

Example (Tableaux Method)

$$\{(p \to q), \neg q\} \models \neg p$$



- Tableaux is closed, so entailment holds.

Example (Natural Deduction)

$$\{(p \to q), \neg q\} \vdash \neg p$$

$$\cfrac{\cfrac{\cfrac{p \to q \quad \cancel{p}}{q} \to E \qquad \neg q}{\cfrac{\bot}{\neg p}\ RAA} \to E}{}$$

- The relation $\vdash$ captures a notion of *deduction* or *proof*.
- It is the syntactic (formal) counterpart of the semantic relation $\models$.
- Consequently, we should expect that:

$$G \models A \textit{ if and only if } G \vdash A$$

Limitations of the Propositional Calculus

- Consider the following argument:

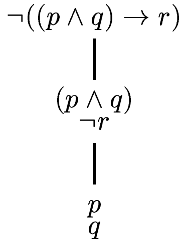
  All lecturers are happy.
  Bill is a lecturer
  So, Bill is happy.

Question: *Is the reasoning here sound (i.e. does the conclusion follow from the premises)?*

Question: *How might the argument be represented in propositional logic?*

- testing validity using a semantic tableau

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



- The tableaux for $\neg((p \wedge q) \to r)$ does not close.
- That means that $(p \wedge q) \to r$ is *not* a tautology.
- That in turn means that the argument is *not* valid!

The Structure of Propositions

- Consider the argument again:

  All lecturers are happy.
  Bill is a lecturer
  So, Bill is happy.

- **Insight:** We need some way of representing the *structure* of the elementary propositions.
- Propositions involve:
  - **named individuals** that the propositions are ‘about’:

    e.g. *Bill, Brighton, Logic*,...
  - **properties** of these individuals:

    e.g. *is_happy*, *is_a_city*, *is_a_lecturer*, ...
  - **relations** between individuals:

    e.g. *teaches*, *lives_in*,....

Question: *Is there anything else involved?*

- Consider the premise

  Bill is a lecturer

- This statement:
  1. expresses a proposition ‘about’ the individual *Bill*’; and
  2. asserts that the individual has the property *is_a_lecturer*
- Rather than use a simple propositional variable (p say), we might represent this by:

  b has the property L

  In fact, we are going to write:

  $$L(b)$$

  where:
  - b stands for the individual called “*Bill*”; and
  - L stands for the property expressed by “*is_a_lecturer* ”’.

- Likewise, we might represent the conclusion of the argument

  Bill is happy

  as follows:

  $$H(b)$$

- But what about the first premise?

  All lecturers are happy

- Note that:
  - this is a *generalization*;
  - it is not about a particular individual, but a whole group.

How can we represent general statements of this kind?

- Paraphrasing a little:

  For all individuals, if he/she is a lecturer, then he/she is happy

- Or perhaps:

  For all x, if x is a lecturer, then x is happy.

- This might be written more succinctly as:

  For all x, $(L(x) \to H(x))$

  In fact, we are going to write:

  $$\forall x.(L(x) \to H(x))$$

- Here, the symbol $\forall$ is know as the **universal quantifier** and can be read as ”for all”.

- So now the whole argument can be notated:

$$\begin{array}{l} \forall x.(L(x) \rightarrow H(x)) \\ L(b) \\ \hline H(b) \end{array}$$

- The notation introduced informally here is the **First Order Predicate Calculus (FOPC)**.
- Predicate Logic is more expressive than simple Propositional Logic.
- We will explore this new logic in the remainder of this course.

Summary

- Propositional logic allows us to represent simple propositions/arguments.
- We have explored the language from the point of view of its meaning and form.
- Propositional logic has limitations – there are some valid arguments that we cannot represent.
- There is more to the structure of propositions than simple boolean combinations of 'atomic' propositions.
- Propositions are about individuals (or sets thereof) and their properties.
- We need a new language for representing this structure: the language of predicate logic.

Introduction to Logic 13

Last time:

- Propositional Logic
- Limitations of Propositional Logic
- The Structure of Propositions
- Individuals, Properties and Quantifiers

This time:

- The Language of the Predicate Calculus
  - Basic Expressions
  - Terms and Formulas
- Expressing Propositions
- Semantic Preliminaries

The Language of FOPC

• The basic expressions of Predicate Logic fall into four separate categories:

1. **Individual names:** a, b, c, ...
   These represent specific objects, persons or events
2. **Individual variables:** x, y, z, ...
   These are variables that range over individuals.
3. **Predicate Symbols:** P, Q, R, ...
   Predicate symbols are used to represent properties or relations over individuals
4. **Function Symbols:** f, g, h, ...
   Function symbols denote functions that map individuals to individuals.

A Note about Predicates and Functions

- **Note:** The predicate and function symbols represent relations over individuals.

  e.g.

Bill **talks**	1 individual
Bill **likes** *Logic*	2 individuals
Bill **teaches** *Moira Logic*	3 individuals

- More generally, we can have relations or functions over an arbitrary number n of individuals

  Terminology: An n-place predicate or function symbol is said to have **arity** n.

- We assume that each predicate or function symbol is associated with a known, fixed arity

- In addition to the four classes of basic expression, the predicate calculus includes:
  - A truth-functionally complete set of connectives: e.g. $\neg, \wedge, \vee, \rightarrow$, and $\leftrightarrow$.
  - Two quantifier symbols:

    The Universal Quantifier: $\forall$

    The Existential Quantifier: $\exists$
  - Brackets '(' and ')', and punctuation symbols ',' and '.'.
- These symbols, taken together with the basic expressions, form the **alphabet** of the language of First Order Predicate Logic.
- The language is defined in two stages: **terms**, and **formulas**

Terms

- Terms are used to pick out individuals:

Definition *A term t is either:*

1. *an individual name; or*
2. *an individual variable; or*
3. *a* functional term $f(t_1, \ldots, t_n)$ *where f is a function symbol of arity n, and $t_1, \ldots, t_n$ are terms*

Question: *Which of the following are terms?*

a

y

P

$f(a, x)$

$Q(x, y)$

$(f(x, x))$

$f(g(x, a), h(b))$

Formulas

Definition *A* **well-formed formula** *of Predicate Logic is either:*

1. *an* atomic formula $P(t_1, \ldots, t_n)$ *where P is a predicate symbol of arity n and $t_1, \ldots, t_n$ are terms; or*
2. *a* compound formula *of one of the following forms:*

   (a) $(\neg A)$

   (b) $(A \wedge B)$

   (c) $(A \vee B)$

   (d) $(A \rightarrow B)$

   (e) $(A \leftrightarrow B)$

   (f) $\forall v.A$

   (g) $\exists v.A$

   where A and B are wffs, and v is an individual variable.

Question: *Which of the below are well-formed formulas of Predicate Logic?*

$P(a)$

$(P(a) \rightarrow Q(a))$

$P(Q(a))$

$(\neg P(f(a, x)))$

$\forall x.(P(x) \rightarrow (\neg Q(x)))$

$(P(\forall x.) \wedge Q(a))$

$(P(a) \vee \exists x.Q(x))$

$\exists x.(P(x) \wedge \forall y.(Q(y) \rightarrow R(x, y)))$

Notes:

- We can drop brackets (as for Propositional Logic) by adopting conventions for operator precedence, etc.
- We may relax conventions for naming predicate symbols, individual names etc.

Representing Propositions

- The FOPC permits finer-grained representation of propositions.

e.g.

Logic is fun

$$F(l)$$

If Logic is fun, then Bill is happy

$$F(l) \rightarrow H(b)$$

Either Logic is fun or Bill is not happy

$$F(l) \vee \neg H(b)$$

All lecturers are happy

$$\forall x.(L(x) \rightarrow H(x))$$

Some lectures are happy

$$\exists x.(L(x) \wedge H(x))$$

Question: *How might you represent the following?*

Some lecturer is not happy

No lecturer is happy

All lecturers teach Logic

All lecturers teach some course

Every course tutor is happy

Semantic Preliminaries

- The semantics of Propositional Logic was particularly simple:
  - **Valuations:** $V : Prop \rightarrow \{t, f\}$
  - Truth tables for connectives:
- For Predicate Logic, the picture is more complicated:
  - four kinds of basic expression;
  - quantification;
  - terms and formulas.
- Meaning is no longer just a matter of **true** and **false** .

- We need a richer semantic domain:
  - individuals – for names
  - functions over individuals – for function symbols
  - relations over individuals - for predicates
- Our semantics should:
  1. map basic expressions onto elements of the semantic domain;
  2. associate individuals with terms; and
  3. give us a way of determining the truth-values of
     - atomic formulas
     - compound formulas (including quantified formulas)

Summary

- The language of First Order Predicate Logic has an alphabet consisting of four basic kind of expression.
- The language is defined in two stages: first order terms, and first order formulas.
- Predicate Logic provides a richer language for representing propositions.
- We can represent propositions concerned with particular individuals, or capture generalisations about groups of classes of individual.
- Our definition of meaning must be correspondingly rich.

Introduction to Logic 14

Last time:

- The Language of the Predicate Calculus
- Expressing Propositions
- Semantic Preliminaries

This time:

- Interpretations
  - Examples
- Formalizing Interpretations
  - Examples

Interpretations

- The Predicate Calculus allows us to:

1. make statements about particular individuals: e.g.

$$T(b,l) \rightarrow H(b)$$

2. make statements about relationships between individuals:
   e.g.

$$T(b,l)$$

3. make general statements about individuals: e.g.

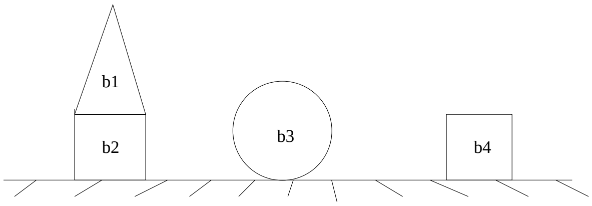
$$\forall x.(T(x,l) \rightarrow \neg H(x))$$

- Just as for Propositional Logic, we can study this new language in two different ways:
  1. in terms of its meaning.
  2. in terms of its form;
- To do the former, we must *interpret* the language.

  i.e. we must:
  1. fix a **domain of interpretation** D
     - the set of things we are interested in talking about
  2. provide an **interpretation function** I
     - relates expressions of the language to our domain D

The ‘Blocks World’

Domain:



Interpretation Function:

Notation	*interpreted as*	Denotation
a	$\Longrightarrow$	b_1
b	$\Longrightarrow$	b_2
c	$\Longrightarrow$	b_3
d	$\Longrightarrow$	b_4
C	$\Longrightarrow$	*is a cube*
P	$\Longrightarrow$	*is a pyramid*
O	$\Longrightarrow$	*is on top of*
R	$\Longrightarrow$	*is red*

Question: *Given the Blocks World domain and associated interpretation function, what do the following statements mean?*

$$C(d)$$

$\Longrightarrow$

$$O(a, b)$$

$\Longrightarrow$

$$\exists x.(C(x) \wedge R(x))$$

$\Longrightarrow$

$$\exists x.(R(x) \wedge P(x) \wedge \exists y.(C(y)) \wedge O(x, y)))$$

$\Longrightarrow$

The Integers

Domain: $\{\ldots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$

Interpretation Function:

Notation	interpreted as	Denotation
z	$\Longrightarrow$	*integer zero*
p	$\Longrightarrow$	*predecessor function*
s	$\Longrightarrow$	*successor function*
L	$\Longrightarrow$	*is less than*

Question: *What do the following statements mean?*

$$L(z, s(z))$$

$\Longrightarrow$

$$\forall x.\exists y.L(x, y)$$

$\Longrightarrow$

$$\exists x.\forall y.L(x, y)$$

$\Longrightarrow$

Formalizing Intepretations

Definition *An interpretion for a first order language is a pair:*

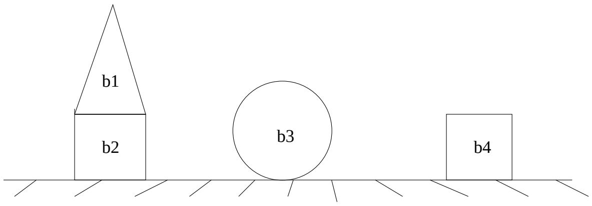
$$\mathcal{I} = (D, I)$$

where:

- *D is a set of individuals (the "domain interpretation"); and*
- *I is an interpretation function*

- The interpretation function I has to
  - assign a fixed element of D to each individual constant a;
  - assign an n-ary function on D to each function symbol f of arity n;
  - assigns an n-ary relation on D to each predicate symbol P of arity n.

The Blocks World (again)



$$D = \{b_1, b_2, b_3, b_4\}$$

$$\begin{array}{lcc}
I(a) & = & b_1 \\
I(b) & = & b_2 \\
I(c) & = & b_3 \\
I(d) & = & b_4 \\
I(C) & = & \{b_2, b_4\} \\
I(P) & = & \{b_1\} \\
I(O) & = & \{(b_1, b_2\} \\
I(R) & = & \{b_1, b_3\}
\end{array}$$

The Integers (more formally)

$$D = \{\ldots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$$

$$\begin{array}{lcll}
I(z) & = & 0 & \\
I(p) & = & \{\ldots & (-3, -4) \\
 & & & (-2, -3) \\
 & & & (-1, -2) \\
 & & & (0, -1) \\
 & & & (1, 0) \\
 & & & (2, 1) \ldots\}
\end{array}$$

$$\begin{array}{lcll}
I(L) & = & \{\ldots & (-5, 1) \\
 & & & (-4, 1) \\
 & & & (-3, 1) \\
 & & & (-2, 1) \\
 & & & (-1, 1) \\
 & & & (0, 1) \ldots\}
\end{array}$$

Notes:

1. So far, we have not mentioned the interpretation of individual variables.
   - We will assume the existence of a separate **variable assignment function** g
   - The assignment function g will map variables onto elements of the domain D
2. Note that a particular interpretation just tells us about the meaning of basic expressions.
   - it does not tell us (directly) about the meaning of compound expressions.
   - .... the meaning of connectives and quantifiers is 'fixed'
   - .... there are general rules for calculating the meaning of compound expressions

Summary

- The language of predicate logic permits us to express propositions about particular individuals, and to make generalizations.
- Like propositional logic, we can study the language from the perspective of its meaning, or its form.
- To study the language in terms of its meaning we must provide an interpretation.
- An interpretation for a first-order language consists of a domain and an intepretation function.
- A particular interpretation fixes the meaning of the basic expressions of a first-order language.
- There are general rules for evaluating the meaning of compound expressions (terms and formulas).