140 Logic

Ian Hodkinson and Alessandra Russo

Department of Computing, Imperial College London

Thanks to Krysia Broda, Robin Hirsch and Dirk Pattinson for additional material

Logic home page:
`http://www.doc.ic.ac.uk/~imh/teaching/140_logic/logic.html`
Slides and exercises: `https://cate.doc.ic.ac.uk/`
Pandora: `http://www.doc.ic.ac.uk/pandora`
`http://www.doc.ic.ac.uk/pandora/newpandora/`
LOST06, LOST07: `programs/lab/Lost` and `programs/lab/LostCD`

Course layout

In about 18 lectures, we will cover:

- Propositional logic
- Predicate logic (also known as 'first-order' logic, or sometimes just 'classical' logic)

There will be weekly tutorials, and you are advised to try the Pandora and LOST programs in the labs later on.

There will be marked (but unassessed) pmt exercises most weeks, and unmarked exercises in the rest. You will get solutions later.

There's a 45-minute logic question in the Xmas test near the end of term, and 2 logic questions (40 minutes each) in the exams in May.

The material is needed for 141 'Reasoning about programs' next term, and several later courses.
You may need it to answer exam questions in these courses.

Books

- *Reasoned Programming,* K. Broda, S. Eisenbach, H. Khoshnevisan, S. Vickers, Prentice-Hall, 1994. Best buy. Out of print I think. See `http://www.doc.ic.ac.uk/pandora`
- *Logic,* Wilfrid Hodges, Penguin, 1977. Excellent introduction, highly recommended for beginners, but doesn't cover the more technical parts of the course.
- *Software Engineering Mathematics,* J. Woodcock, M. Loomes, Pitman, 1988. Early chapters quite close to course. Natural deduction done Gentzen-style, not with boxes, but you should be able to work it out.
- *Beginning Logic,* E. Lemmon, Van Nostrand Reinhold, 1982. Classic. More for mathematicians, natural deduction in different style from the course. But you might make a go of it with effort.

- *How to prove it,* D. Velleman, Cambridge University Press, 1995. Examples mostly from maths. Early chapters OK but no natural deduction.
- Just go to the library and find one you like.

More advanced:

- *Logic in computer science: modelling and reasoning about systems,* M. Huth and M. Ryan, Cambridge University Press, 2000.
- *Formal Logic: its scopes and limits,* R. Jeffrey, McGraw Hill, 1981.
- *Logic: techniques of formal reasoning,* Kalish, Montague, Mar. Harcourt Brace Janovich, 1980 (2nd edn).
- *Methods of Logic,* W. Quine, Harvard UP, 1982.

1. Introduction: what is logic?

Shorter Oxford Dictionary:

> 'Logic, from Greek *logos* (reasoning, discourse)
> The branch of philosophy that deals with forms of reasoning and thinking, especially inference and scientific method.
> Also, the systematic use of symbolic techniques and mathematical methods to determine the forms of valid deductive argument.'

We are not concerned here with low-level 'logic gates' etc in computing — that is for the Hardware course.

We are concerned with using logic to describe, specify, verify, and reason about software.

At higher levels, logic can shed light on 'forms of thinking' — relevant to AI.

Why should you learn logic?

- Logic is the 'calculus of computing': a good mathematical foundation for dealing with information, and reasoning about program behaviour.
- It also provides a good training in correct reasoning and accurate, unambiguous description.
- It is needed in later courses/areas: Reasoning about programs, Prolog, Databases(?), SE – Design, SE – Systems Verification, Software Reliability, many AI courses, ...
- Some of the more sophisticated real-world application areas need and use logic. Example: Prolog is a logic programming language that you can buy commercially. The SQL database language, and model-checking, are based on logical ideas.
- Propositional and first-order logic are the most fundamental of all logical systems. If you ever do any logic, you really have to do these first.

So what makes up logic?

A logical system usually consists of three things:

1. Syntax — a *formal* language (like a programming language) that will be used to express concepts.
2. Semantics — provides *meaning* for the formal language.
3. Proof theory — a purely syntactic way of obtaining or identifying the valid statements of the language. It provides a way of *arguing* in the formal language.

Far, far more can be done with logic than this. Logic has now attained the depth and sophistication of modern mathematics: eg Fermat's last theorem.

But this is a first course in logic, and we stick near the ground!

2. Propositional logic

This is the study of 'if-tests':

```
if count>0 and not found then
  decrement count; look for next entry;
end if
```

Features:

- basic (or 'atomic') statements — here: `count>0`, `found` — are true or false depending on circumstances
- can use boolean operations — `and`, `or`, `not`, etc. — to build more and more complex test statements from the atomic propositions
- the final complex statement evaluates to true or false.

Propositional logic is not very expressive. Predicate logic (later) can say much more. Eg. 'every student has a tutor'.

Why do propositional logic?

Anyone who's written imperative programs knows about if-tests. Why study them in university?

- All logics are based on propositional logic in some form. We have to start with it.
- We want to handle arbitrarily complicated if-tests and study their general features.
- We don't just want to *evaluate* if-tests.

  We want to know how to find out when two if-tests mean the same, when one implies another, whether an if-test (or loop test) can ever be made true or not.
- Propositional logic encapsulates an important class of computational problems, and so comes in to algorithms and complexity theory.

2.1 Syntax — the formal language

First, we need to fix a precise definition of the kinds of if-test we will consider. This will constitute the formal language of propositional logic. It has three ingredients.

1st ingredient: Propositional atoms

We are not concerned about which atomic facts (`count>0`, `found`) are involved. So long as they can get a *truth value* (true or false), that's enough.

So we fix a collection of algebraic symbols to stand for these atomic statements.

The symbols are called *propositional atoms.* Many better-educated people call them *propositional variables* or *propositional letters* instead.
For short, I'll usually call them *atoms.*

They are like variables x, y, z in maths.

But because they are Propositional, we usually use the letters $p, p', p_0, p_1, p_2, \ldots$, and also $q, r, s, \ldots$

2nd ingredient: Boolean connectives

We are interested in the following boolean operations, or *connectives*:

- and: written as $\wedge$ (or sometimes &, and in old books, '$\cdot$')
- not: written as $\neg$ (or sometimes $\sim$)
- or: written as $\vee$ (in old books, $+$)
- 'if-then', or 'implies'. This is written as $\rightarrow$ (or sometimes $\supset$, but not as $\Rightarrow$, which is used differently).
- if-and-only-if: written $\leftrightarrow$ (but not as $\iff$ or $\equiv$, which are used differently)
- truth and falsity: written as $\top, \bot$

Warning: 'implies' is just the way to *say* $\rightarrow$. We'll discuss the *meaning* of $\rightarrow$ later. The same applies to the other connectives.

Examples

- Our test `count>0 and not found` would come out as $(\texttt{count} > 0) \wedge \neg\, \texttt{found}$, or maybe $p \wedge \neg q$.
- Symbols for and, or, implies, if-and-only-if take 2 arguments and are written in infix form: $p \wedge q,\ \ p \vee q,\ \ p \rightarrow q,\ \ p \leftrightarrow q$.
- Negation $\neg$ takes 1 argument, which is written to the right of it: eg, $\neg p,\ \ \neg q$.
- $\top, \bot$ take no arguments.
  They are logical *constants* (like π, e).

You'll have to learn these new symbols, so you can read logic books.
They are also quicker to write than `and, or, not,` etc.

3rd ingredient: Punctuation

We need brackets to disambiguate our if-tests.

This is like terms in arithmetic. $1 - 2 + 3$ is ambiguous. We can read it as $(1 - 2) + 3$ or $1 - (2 + 3)$, and the difference matters.

For example, $p \wedge q \vee r$ might be read as

- $(p \wedge q) \vee r$,
- $p \wedge (q \vee r)$,

and the difference matters.

So we start by putting all brackets in, and then deleting those that aren't needed.

Formulas

What I've called if-tests are called *formulas* by logicians. (Some call them *well-formed formulas,* or *wffs,* but I think this is daft.)

A propositional formula is a string of symbols made from propositional atoms using the boolean connectives above, and brackets, in the appropriate way. More accurately:

Definition 2.1 (propositional formula)

1. *Any propositional atom (p, q, r, etc) is a (propositional) formula.*
2. *$\top$ and $\bot$ are formulas.*
3. *If A is a formula then so is $(\neg A)$.* Note the brackets!
4. *If A, B are formulas then so are $(A \wedge B)$, $(A \vee B)$, $(A \to B)$, and $(A \leftrightarrow B)$.*
5. *That's it: nothing is a formula unless built by these rules.*

Examples of formulas

The following are formulas:

- p
- $(\neg p)$
- $((\neg p) \wedge \top)$
- $(\neg((\neg p) \wedge \top))$
- $((\neg p) \rightarrow (\neg((\neg p) \wedge \top)))$

You can see the brackets are a pain already. We need some conventions to get rid of (some of) them.
What we'll get are (strictly speaking) *abbreviations* of genuine formulas. But most people seem to think of them as real formulas.

We can always omit the final, outermost brackets:
$\neg p$, and $(\neg p) \rightarrow (\neg((\neg p) \wedge \top))$ are unambiguous.

Binding conventions

To get rid of more brackets, we *order* the boolean connectives according to decreasing binding strength:

$$\text{(strongest) } \neg,\ \wedge,\ \vee,\ \rightarrow,\ \leftrightarrow \text{ (weakest)}$$

This is like in arithmetic, where $\times$ is stronger than $+$. This means that $2 + 3 \times 4$ is usually read as $2 + (3 \times 4)$, not as $(2 + 3) \times 4$. So:

- $p \vee q \wedge r$ is read as $p \vee (q \wedge r)$, not as $(p \vee q) \wedge r$.
- $\neg p \wedge q$ is read as $(\neg p) \wedge q$, not as $\neg(p \wedge q)$.
- $p \wedge \neg q \rightarrow r$ is read as $(p \wedge (\neg q)) \rightarrow r$, rather than $p \wedge (\neg(q \rightarrow r))$ or $p \wedge ((\neg q) \rightarrow r)$.

But don't take the conventions too far. Eg, $p \rightarrow \neg q \vee \neg\neg r \wedge s \leftrightarrow t$ is a mess! USE BRACKETS if it helps readability, even if they are not strictly needed.

Repeated connectives

What about $p \to q \to r$? The binding conventions are no help now. Should we read it as $p \to (q \to r)$ or as $(p \to q) \to r$?

Haskell would suggest $p \to (q \to r)$, common sense might suggest $(p \to q) \to r$. (See what you think later on.)

There probably is a convention here. But after many years doing logic, I still find it too hard to remember to be worth the space saved. I (and most logicians) would always put brackets in such a formula.

However, $p \wedge q \wedge r$ and $p \vee q \vee r$ are usually OK as they are.

As we'll see, their logical meaning is the same however we bracket them. $(p \wedge q) \wedge r$ and $p \wedge (q \wedge r)$ are different formulas. But we will see that they always have the same truth value in any situation: they are 'logically equivalent'.
So we don't really care how we disambiguate $p \wedge q \wedge r$. (Usually, it is $(p \wedge q) \wedge r$ — from left.)

Special cases

Sometimes the binding conventions don't work as we would like.

For example, if I saw

$$p \to r \wedge q \to r,$$

I'd probably read it as $(p \to r) \wedge (q \to r)$.

How can I justify this? $\wedge$ is stronger than $\to$. So shouldn't it be $p \to (r \wedge q) \to r$?
Read according to a plausible convention about repeated $\to$, this is $(p \to (r \wedge q)) \to r$.

But I say few sane people would want to write $(p \to (r \wedge q)) \to r$.
So $p \to r \wedge q \to r$ seems more likely to mean $(p \to r) \wedge (q \to r)$.

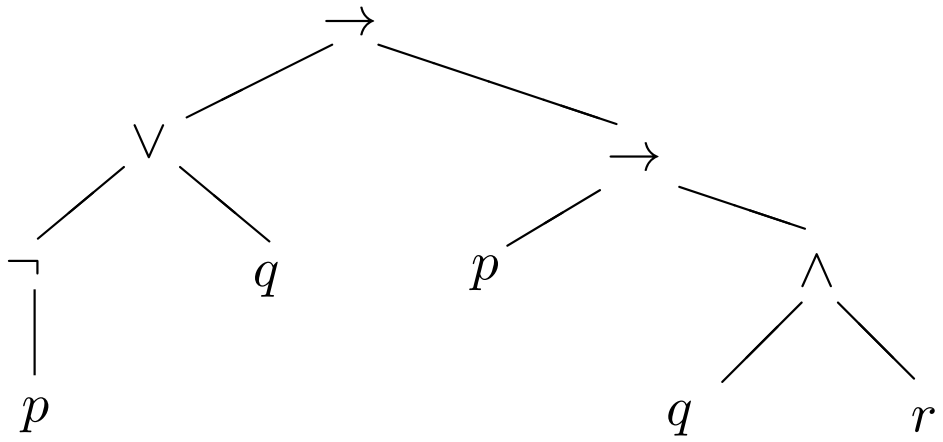
The lesson: formulas like $p \to r \wedge q \to r$ can be misinterpreted without brackets, even if they are not strictly needed. Don't write such formulas without brackets.
USE BRACKETS if it helps readability, even if they are not strictly needed.

Parsing: formation tree, logical form

We have shown how to read a formula unambiguously — to *parse* it.

The information we gain from this can be represented as a tree: the *formation tree* of the formula.

For example, $\neg p \vee q \to (p \to q \wedge r)$ has the formation tree



This is a nicer (but too expensive) way to write formulas.

Principal connective

Note that the connective at the root (top!) of the tree is $\rightarrow$. This is the *top connective* or *principal connective* of $\neg p \vee q \rightarrow (p \rightarrow q \wedge r)$. This formula has the overall *logical form* $A \rightarrow B$.

Every non-atomic formula has a principal connective, which determines its overall logical form. You have to learn to recognise it.

- $p \wedge q \rightarrow r$ has principal connective $\rightarrow$.
  Its overall logical form is $A \rightarrow B$.
- $\neg(p \rightarrow \neg q)$ has principal connective $\neg$. Its logical form is $\neg A$.
- $p \wedge q \wedge r$ has principal connective $\wedge$ (probably the 2nd one!). Its logical form is $A \wedge B$.
- $p \vee q \wedge r$ has principal connective $\vee$. Its logical form is $A \vee B$.

What are the formation trees of these formulas?

Subformulas

The tree view makes it easy to explain what a subformula is.

The *subformulas* of a formula A are the formulas built in the stages on the way to building A as in definition 2.1.

They correspond to the nodes, or to the subtrees, of the formation tree of A.

The subformulas of $\neg p \vee q \to (p \to q \wedge r)$ are:

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

$$\neg p \vee q \qquad\qquad p \to q \wedge r$$

$$\neg p \qquad q \qquad p \qquad q \wedge r$$

$$p \qquad\qquad q \qquad r$$

Notes: there are *two different* subformulas p.
And $p \vee q$ and $p \to q$ are substrings but NOT subformulas.

Abbreviations

You may see some funny-looking formulas in books:

- $\bigwedge_{1\leq i\leq n} A_i$, $\bigwedge_{i=1}^{n} A_i$, $\bigwedge_{i=1}^{n} A_i$, $\bigwedge_{1\leq i\leq n} A_i$, $\bigwedge\!\!\bigwedge_{1\leq i\leq n} A_i$.

  These all abbreviate $A_1 \wedge A_2 \wedge \ldots \wedge A_n$.

  Example: $\bigwedge_{1\leq i\leq n} p_i \to q$ abbreviates $p_1 \wedge \ldots \wedge p_n \to q$.

- $\bigvee_{1\leq i\leq n} A_i$, $\bigvee_{i=1}^{n} A_i$, $\bigvee_{i=1}^{n} A_i$, $\bigvee_{1\leq i\leq n} A_i$, $\bigvee\!\!\bigvee_{1\leq i\leq n} A_i$.

  These abbreviate $A_1 \vee A_2 \vee \ldots \vee A_n$.

This is like in algebra:

$\sum_{i=1}^{n} a_i$ abbreviates $a_1 + a_2 + \cdots + a_n$.

Technical terms for logical forms

To understand logic books, you'll need some jargon. Learning it is tedious but necessary.

Definition 2.2

- *A formula of the form* $\top$, $\bot$, *or* p *for an atom* p, *is (as we know) called* atomic.
- *A formula whose logical form is* $\neg A$ *is called a* negated formula. *A formula of the form* $\neg p$, $\neg\top$, *or* $\neg\bot$ *is sometimes called* negated-atomic.
- *A formula of the form* $A \wedge B$ *is called a* conjunction, *and* A, B *are its* conjuncts.
- *A formula of the form* $A \vee B$ *is called a* disjunction, *and* A, B *are its* disjuncts.
- *A formula of the form* $A \to B$ *is called an* implication. A *is called the* antecedent, B *is called the* consequent.

Technical terms ctd. — literals and clauses

Definition 2.3

- *A formula that is either atomic or negated-atomic is called a* literal. *(In maths it's sometimes called a* basic formula.*)*
- *A* clause *is a disjunction* $(\vee)$ *of one or more literals.*

Eg, the formulas p, $\neg r$, $\neg\bot$, $\top$ are all literals.

Four examples of clauses

$$p$$

$$\neg p$$

$$p \vee \neg q \vee r$$

$$p \vee p \vee \neg p \vee \neg\bot \vee \top \vee \neg q$$

Some people allow the empty clause (the disjunction of zero literals!), which by convention is treated the same as $\bot$.

2.2 Semantics — meaning

We know how to read and write formulas. But what do they mean? Posh way: 'what is their semantics?'

The Boolean connectives (and $\wedge$, not $\neg$, or $\vee$, if-then $\rightarrow$, if and only if $\leftrightarrow$) have *roughly* their English meanings.

But English is a natural language, full of ambiguities, subtleties, and special cases. Translating between English and logic is fraught with difficulties. See (eg) Hodges' book for many nasty examples.

We are engineers. We need a precise idea of the meaning of formulas — especially as there are infinitely many of them. We may want to implement it. In Haskell.

So we prefer to give a formal mathematical-style definition of meaning.

Giving meaning to formulas

Definition 2.4 (situation) *A* situation *is simply something that determines whether each propositional atom is true or false.*

For if-tests, a situation is some point in a program execution. The current values of the program variables will determine whether or not each atomic statement of an if-test (such as $x > 0$, $x = y$, etc) is true.

For *'if (it rains) then (it is cloudy)',* a situation is the weather.

For atoms $p, q, \ldots$, a situation just specifies which of them are true and which are false.

Never forget: there is more than one situation. In a different situation, the truth values may be different. Obviously.

Knowing the situation, we can work out the *truth value* of any given propositional formula — whether it is true or false *in this situation.* We work from simple formulas to more complex ones, as follows.

Working out truth values

Definition 2.5 (evaluation) *The truth value of a propositional formula in a given situation is defined as follows.*

- *The situation tells us directly the truth values of atoms.*
- $\top$ *is true, and* $\bot$ *is false.*

Assume A, B *are formulas, and we've already worked out whether they are true or false in the given situation. Then in this situation:*

- $\neg A$ *is true if* A *is false, and false if* A *is true.*
- $A \wedge B$ *is true if* A *and* B *are both true; otherwise it is false.*
- $A \vee B$ *is true if one or both of* A *and* B *are true, false otherwise. (Inclusive or)*
- $A \to B$ *is true if* A *is false or* B *is true (or both). Otherwise — if* A *is true and* B *is false — it is false.* 'If A is true, so is B.'
- $A \leftrightarrow B$ *is true if* A *and* B *have the same truth value (both true, or both false). If they don't, it's false.*

Truth tables for connectives

We can express these rules of meaning using *truth tables* for the connectives.

We write 1 for true and 0 for false. (Some write *tt, ff* instead.)
Do not confuse them with $\top, \bot$, which are formulas, not truth values.

A	$\neg A$	$\top$	$\bot$
1	0	1	0
0	1	1	0

A	B	$A \wedge B$	$A \vee B$	$A \to B$	$A \leftrightarrow B$
1	1	1	1	1	1
1	0	0	1	0	0
0	1	0	1	1	0
0	0	0	0	1	1

Aside: general connectives

Actually, *any* truth table gives us a (possibly new) connective.

E.g., the table

A	B	$A \uparrow B$
1	1	0
1	0	1
0	1	1
0	0	1

defines what is called the NAND connective, also known as the Sheffer stroke, and often written $\uparrow$.

Any connective can be expressed with $\wedge, \neg$ (and even with just $\uparrow$). E.g., $A \uparrow B$ can be expressed as $\neg(A \wedge B)$.

See exercises and exam questions for more examples and information.

Examples

Suppose that p is true and q is false in a certain situation. Then in this situation:

- $\neg p$ is false
- $\neg q$ is true
- $p \wedge q$ is false
- $p \vee q$ is true
- $p \rightarrow q$ is false
- $q \rightarrow p$ is true
- $p \leftrightarrow q$ is false, and $\neg p \leftrightarrow q$ is true.

A different situation

Suppose now that p is false, and q is true. In this new situation:

- $\neg p$ is now true
- $\neg q$ is now false
- $p \wedge q$ is still false
- $p \vee q$ is still true
- $p \rightarrow q$ is now true
- $q \rightarrow p$ is false now
- $p \leftrightarrow q$ is still false, and $\neg p \leftrightarrow q$ is still true.

So do not ask whether your formula is true.
(It's like asking 'is $3x = 7$ true?')
Ask if it is true in such-and-such a situation.

Now it's your turn

Suppose again that p is true and q is false in a certain situation. Are the following true or false?

1. $p \to \neg q$
2. $q \to q$
3. $\neg p \vee \neg q$
4. $p \wedge q \to q$
5. $\neg(p \leftrightarrow \top)$
6. $\neg(q \to \neg(p \vee q))$

2.3 English correspondence

As I said, translating from English to logic is difficult. But we have to do it for applications.

We can only translate (some) *statements;* not questions, commands/invitations, exclamations.

Basic idea: extract suitable atomic propositions from the English sentence, and join them up appropriately with boolean connectives. But you will need massive amounts of common sense.

Eg: 'I bought beans and peas' could be translated as
'I bought beans $\wedge$ I bought peas'.

'I am not good at logic' could be translated as '$\neg$(I am good at logic)'.

'If it's raining then it's not sunny' could be translated as
'raining $\rightarrow \neg$sunny'.

'The answer is 3 or 6' could be translated as
'The answer is 3 $\vee$ the answer is 6'.

English variants

‘But’ means ‘and’.
‘I will go out, but it is raining’ translates to ‘I will go out $\wedge$ it is raining’.

‘Unless’ generally means ‘or’.
‘I will go out unless it rains’ translates to ‘I will go out $\vee$ it will rain’.
(Note the extra ‘will’.)
You could also use ‘$\neg$(it will rain) $\rightarrow$ I will go out’.

But you may think that ‘I will go out unless it rains’ implies that if it does rain then I won’t go out. This is a stronger reading of ‘unless’. If you take that view, you’d translate it as ‘I will go out $\leftrightarrow$ $\neg$(it will rain)’. (This is ‘exclusive or’.)

Deciding whether to take the strong or weak reading is done by individual context, can be very subtle, and is sometimes impossible — English is not always precise.
But in computing, people always prefer the weak form.

Or

In everyday English, ‘or’ sometimes means ‘exclusive or’ (the ‘strong reading’).
You’d probably say ‘I have keys or coins in my pocket’ is false if I have both. So this ‘or’ is strong.

But sometimes it means ‘inclusive or’ (the ‘weak reading’).
‘If I have keys or coins in my pocket, I’d better be careful at night’ implies I’d still better be careful if I have both. So this ‘or’ is weak!

(The way English handles nested connectives is very subtle!)

But in computing we always take the weak reading: inclusive or.

We use English as a technical language. As an initiate/neophyte, you have to learn to accept this and use the weak forms.

But when reading English written by non-computing people, use your common sense to determine what they meant (or ask them!)

Modalities (nasty)

A modality shifts the context of utterance.

Eg time: 'I will be rich and famous' can't be accurately translated as 'I will be rich $\wedge$ I will be famous'.

The formula is true if I get famous only when I lose all my money.

The English probably isn't: it suggests I'll be rich and famous at the same time.

Eg permission: 'You can have chicken or fish' usually means 'You can have chicken $\wedge$ you can have fish'.

Eg obligation: 'I must do topics or a foreign language' (was true) can't be translated as 'I must do topics $\vee$ I must do a language' (was false — you could choose which to take).

Modalities are beyond the scope of this course.
See 4th year course 499H modal logic.

From logic to English

You might think this is straightforward:

- '$p \wedge q$' translates to 'p and q'
- '$\neg p$' to 'not p'
- $p \to q$ to 'if p then q' or perhaps 'q if p',

etc. Eg, 'It is raining $\to$ $\neg$ I go out' translates to 'If it is raining then I do not go out'.

But there are problems.

Complex formulas are hard to render naturally in English.

'$\to$' is a nightmare to translate. The semantics of $\to$ works superbly for logic. But in English we use if-then in many different ways, and not always carefully. Don't read $\to$ as 'causes' — too strong.

Eg 'I am the pope $\to$ I am an atheist' is true here and now (why?)

But would you say that the English sentence 'If I am the pope, then I am an atheist' is true?

A really vicious example

This one deserves careful thought. Consider the formulas

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

(2) $(p \to r) \vee (q \to r)$.

Let p be 'I get into the lift', q be 'you get into the lift', and r be 'the lift breaks, crashes to the ground, and burns'.

Then (1) says that *if we both get in the lift, then the lift breaks.*

(2) says that *if I get in then the lift breaks, or if you get in then the lift breaks.* Right?

Would you say these mean the same? I guess not.

But they do! The formulas (1) and (2) are true in exactly the same situations. They are 'logically equivalent'! We'll see this later.

I think the trouble is that in the English version of (2), you read the 'or' as 'and'. You shouldn't!

3. Arguments, validity

You now know how to read, write, and evaluate propositional formulas. You know a little about translating between English and logic.

But logic is not just about saying things. It allows arguing too. Some say it is the study of valid arguments.

Here is an old argument:

- Socrates is a man.
- Men are mortal.
- Therefore, Socrates is mortal.

Is this a valid argument? What does it mean to say that it is or is not a valid argument?

3.1 Valid arguments

Our answer is, it is a valid argument if for each situation in which Socrates is a man and also men are mortal, Socrates is mortal in this situation.

In our simple propositional logic, a situation simply defines the truth value of each propositional atom (see definition 2.4).

So we propose:

Definition 3.1 (valid argument) *Given formulas $A_1, A_2, \ldots, A_n, B$, an argument*

$$\text{`}A_1, \ldots, A_n\text{, therefore } B\text{'}$$

is valid *if: B is true in every situation in which $A_1, \ldots, A_n$ are all true.*

If this is so, we write '$A_1, \ldots, A_n \models B$'.

The '$\models$' is called 'double turnstile'. You can read it as 'logically entails/implies', or 'semantically entails'.

Examples of arguments

Let A, B be arbitrary propositional formulas.

- 'A, therefore A' is valid, since in any situation in which A is true, A is true. $A \models A$.
- '$A \wedge B$, therefore A' is valid. $A \wedge B \models A$.
- 'A, therefore $A \wedge B$' is not in general valid: depending on A and B, there might be situations in which A is true but $A \wedge B$ is not. Then, $A \not\models A \wedge B$.
- '$A,\ A \to B$, therefore B' is valid. $A,\ A \to B \models B$. This argument style is so common and old, it has a name: *modus ponens.*
- $A \to B,\ \neg B$, therefore $\neg A$ is also valid: $A \to B, \neg B \models \neg A$. This argument is called *modus tollens.*
- '$A \to B$, B, therefore A' is not valid in general, in spite of what lawyers and politicians may say. $A \to B, B \not\models A$.

3.2 Valid, satisfiable, equivalent formulas

Three important ideas are related to valid arguments.

Definition 3.2 (valid formula) *A propositional formula is* (logically) valid *if it is true in every situation.*

We write '$\models A$' if A is valid.

Valid *propositional* formulas are often called *tautologies.* A tautology is just another name for a valid propositional formula.

Definition 3.3 (satisfiable formula) *A propositional formula is* satisfiable *if it is true in at least one situation.*

Definition 3.4 (equivalent formulas) *Two propositional formulas A, B are* logically equivalent *if they are true in exactly the same situations. Roughly speaking: they mean the same.*

Some people write $A \equiv B$ for this. But $\equiv$ is also used in other ways, so watch out.

Examples

Formula	valid	satisfiable
$\top$	yes	yes
$\bot$	no	no
p	no	yes
$p \wedge \neg p$	no	no
$p \rightarrow p$	yes	yes

Equivalent?	p	$\top$	$p \rightarrow q$
$p \wedge p$	yes	no	no
$p \vee \neg p$	no	yes	no
$\neg p \vee q$	no	no	yes

Some useful validities and equivalences are given later as exercises for you to check.

3.3 Links between the four concepts

Valid arguments and valid, satisfiable, and equivalent formulas are all definable from each other:

argument	validity	satisfiability	equivalence
$A \models B$	$A \to B$ valid	$A \wedge \neg B$ unsatisfiable	$(A \to B) \equiv \top$
$\top \models A$	A valid	$\neg A$ unsatisfiable	$A \equiv \top$
$A \not\models \bot$	$\neg A$ not valid	A satisfiable	?
$A \models B$ and $B \models A$	?	$A \leftrightarrow \neg B$ unsatisfiable	$A \equiv B$

All four statements in each line amount to the same thing.

4. Checking validity

So we need only deal with (e.g.) valid formulas — we get the other three notions for free.

How do we tell whether a formula is valid?

In principle, we know how: check that the formula is true in every situation.
Finding good ways of doing it in practice will occupy us off and on till Xmas.

For propositional formulas, it can be done, though computationally it's a hard problem.

For predicate logic, coming later, it's much harder, because the situations are more complex and numerous (infinitely many). But we can often show that particular predicate formulas are valid by tricks, inspiration, or good luck.

Ways to check propositional validity

There are several ways to find out if a given propositional formula is valid or not:

- Truth tables — the obvious way. You just go through all possible relevant situations and see if the formula is true in each.
- Direct 'mathematical' argument. This is the fastest of all, once you get used to it. I myself often use this method.
- Equivalences. You make a stock of useful pairs of equivalent formulas. You use them to reduce your formula step by step to $\top$, which is valid. This takes some art but is quite useful. I use it quite often (especially combined with direct argument).
- Various proof systems, including tableaux, Hilbert systems, natural deduction. We will look at natural deduction: it is similar to direct argument. It deserves a section (§5) to itself.

4.1 Truth tables

Let's show that $(p \to q) \leftrightarrow (\neg p \vee q)$ is valid.
We write 1 for true and 0 for false.

We have two atoms, each taking one of two truth values. The truth values of other atoms are irrelevant.
So there are four possible relevant situations. We evaluate all subformulas of the formula in each one:

p	q	$p \to q$	$\neg p$	$\neg p \vee q$	$(p \to q) \leftrightarrow (\neg p \vee q)$
1	1	1	0	1	1
1	0	0	0	0	1
0	1	1	1	1	1
0	0	1	1	1	1

We see that $(p \to q) \leftrightarrow (\neg p \vee q)$ is true (it has value 1) in all four situations. So it's valid.

Equivalence and satisfiability

The same table shows that $p \to q$ and $\neg p \vee q$ are equivalent: in each of the four situations, they have the *same truth value.*

What do we look for to show satisfiability?

Pros and cons of truth tables

They are boring and tedious, dumb, and error-prone ('write-only').

But they always work — at least for propositional logic, where only finitely many situations are relevant to a given formula. (This is not true for predicate logic: we have to find other ways there.)

They can also be used for showing satisfiability and equivalence.

They are easy to implement (not recommended).

They illustrate how hard deciding validity, satisfiability, etc can be — even for 'trivial' propositional logic. A formula with n atoms needs 2^n lines in its truth table. Adding an atom doubles the length of the table.

No satisfactory way to determine propositional satisfiability is known. You may see in later years that this problem is NP-complete. Breaking RSA public-key encryption is no harder, and is believed to be easier!

4.2 Direct argument

Let's show that $p \to p \vee q$ is valid.

Take an arbitrary situation. We show $p \to p \vee q$ is true in this situation.

So we have to show that IF p is true in this situation THEN so is $p \vee q$.

Well, if p is true, then one of p, q is true, so $p \vee q$ is true. And there we are.

Exercise 4.1 Show that $A \wedge (A \to B) \to B$ is valid.

Direct argument: another example

Let's show $(A \wedge B) \wedge C$ is logically equivalent to $A \wedge (B \wedge C)$. (Equally (page 44), we could check that $(A \wedge B) \wedge C \leftrightarrow A \wedge (B \wedge C)$ is valid.)

Take any situation.

$(A \wedge B) \wedge C$ is true in this situation if and only if $A \wedge B$ and C are both true (by definition of semantics of $\wedge$).

This is so if and only if A and B are true, and also C is true — that is, they're all true.

This is so if and only if A is true, and also B and C are true.

This is so if and only if A and $B \wedge C$ are true.

This is so if and only if $A \wedge (B \wedge C)$ is true.

So $(A \wedge B) \wedge C$ and $A \wedge (B \wedge C)$ have the same truth value in this situation. The situation was arbitrary, so they are logically equivalent.

Direct argument: another example

Here's a more complex example. Let's try to show that $((p \to q) \to p) \to p$ (known as 'Peirce's law') is valid.

Take an arbitrary situation. If p is true in this situation, then $((p \to q) \to p) \to p$ is true, since any formula $A \to B$ is true when B is true. We are done.

If not, then p must be false in this situation.

So $p \to q$ is true, because $A \to B$ is true when A is false.

So $(p \to q) \to p$ is false, because the LHS is true and the RHS false: $A \to B$ is false when A is true and B false.

So $((p \to q) \to p) \to p$ is true, because $A \to B$ is true when A is false. We are done again, and finished.

This was an *argument by cases:* p true, or p false. They are exhaustive: this is known as the *'law of excluded middle'*.

4.3 Equivalences

These form a toolbox for simplifying a formula or converting a formula into another, always preserving logical equivalence.

So (eg) if we can simplify a formula to $\top$, we know it's valid.

(We do this kind of thing in arithmetic:

$$\begin{aligned} 3(x+y^2)-2y(y+4) &= 3x+3y^2-2y^2-8y \\ &= 3x+y^2-8y. \end{aligned}$$

The value is preserved in each step.)

We are going to list some examples of equivalences that have been found useful for this.

There are quite a lot, but they all embody basic logical principles that you should know. You should check that the formulas really are logically equivalent, using truth tables or direct argument.

Equivalences involving $\wedge$

In the equivalences, A, B, C will denote arbitrary formulas. For short, I will often say 'equivalent' rather than 'logically equivalent'.

1. $A \wedge B$ is logically equivalent to $B \wedge A$
   (commutativity of $\wedge$)
2. $A \wedge A$ is logically equivalent to A
   (idempotence of $\wedge$)
3. $A \wedge \top$ and $\top \wedge A$ are logically equivalent to A
4. $\bot \wedge A$, $A \wedge \bot$, $A \wedge \neg A$, and $\neg A \wedge A$ are all equivalent to $\bot$
5. $(A \wedge B) \wedge C$ is equivalent to $A \wedge (B \wedge C)$ (associativity of $\wedge$)

Equivalences involving $\vee$

6. $A \vee B$ is equivalent to $B \vee A$
   (commutativity of $\vee$)
7. $A \vee A$ is equivalent to A
   (idempotence of $\vee$)
8. $\top \vee A$, $A \vee \top$, $A \vee \neg A$, and $\neg A \vee A$ are equivalent to $\top$
9. $A \vee \bot$ and $\bot \vee A$ are equivalent to A
10. $(A \vee B) \vee C$ is equivalent to $A \vee (B \vee C)$ (associativity of $\vee$)

Equivalences involving $\neg$

11. $\neg\top$ is equivalent to $\bot$
12. $\neg\bot$ is equivalent to $\top$
13. $\neg\neg A$ is equivalent to A

Equivalences involving $\rightarrow$

14. $A \rightarrow A$ is equivalent to $\top$
15. $\top \rightarrow A$ is equivalent to A
16. $A \rightarrow \top$ is equivalent to $\top$
17. $\bot \rightarrow A$ is equivalent to $\top$
18. $A \rightarrow \bot$ is equivalent to $\neg A$
19. $A \rightarrow B$ is equivalent to $\neg A \vee B$, and also to $\neg(A \wedge \neg B)$
20. $\neg(A \rightarrow B)$ is equivalent to $A \wedge \neg B$.

Equivalences involving $\leftrightarrow$

21. $A \leftrightarrow B$ is equivalent to
    - $(A \to B) \wedge (B \to A)$,
    - $(A \wedge B) \vee (\neg A \wedge \neg B)$,
    - $\neg A \leftrightarrow \neg B$.
22. $\neg(A \leftrightarrow B)$ is equivalent to
    - $A \leftrightarrow \neg B$,
    - $\neg A \leftrightarrow B$,
    - $(A \wedge \neg B) \vee (\neg A \wedge B)$.

    (This is one way to express the *exclusive or* of A, B.)

De Morgan laws

Augustus de Morgan (19th-century logician) did not discover these (they are much older), and he could not even express them in his own notation!

23. $\neg(A \wedge B)$ is equivalent to $\neg A \vee \neg B$

24. $\neg(A \vee B)$ is equivalent to $\neg A \wedge \neg B$

Distributivity of $\wedge, \vee$

25. $A \wedge (B \vee C)$ is equivalent to $(A \wedge B) \vee (A \wedge C)$.
    $(B \vee C) \wedge A$ is equivalent to $(B \wedge A) \vee (C \wedge A)$.

26. $A \vee (B \wedge C)$ is equivalent to $(A \vee B) \wedge (A \vee C)$
    $(B \wedge C) \vee A$ is equivalent to $(B \vee A) \wedge (C \vee A)$

Absorption

27. $A \wedge (A \vee B)$ and $A \vee (A \wedge B)$ are equivalent to A.
    So are $A \wedge (B \vee A)$, $(A \wedge B) \vee A$, etc.

Proving validity by equivalences

Take any subformula of the given formula, and replace it by any equivalent formula.
Repeat. Any formula we get to is logically equivalent to the original.

If we end up with $\top$, the original formula is valid.

Eg: let's show $p \to p \vee q$ is valid.

- It is equivalent to $\neg p \vee (p \vee q)$
  — by the equivalence '$A \to B \equiv \neg A \vee B$'.
- This is equivalent to $(\neg p \vee p) \vee q$ — by $A \vee (B \vee C) \equiv (A \vee B) \vee C$.
- This is equivalent to $\top \vee q$ — by $\neg A \vee A \equiv \top$.
- This is equivalent to $\top$ (by $\top \vee A \equiv \top$), which is valid.

We got to $\top$, a valid formula. So $p \to p \vee q$ is valid.

Another example

Let's show $A \to B$ is equivalent to $\neg B \to \neg A$.

$\neg B \to \neg A$ is equivalent to $\neg(\neg B) \vee \neg A \quad (X \to Y \equiv \neg X \vee Y)$

This is equivalent to $B \vee \neg A \quad (\neg\neg X \equiv X)$.

This is equivalent to $\neg A \vee B \quad$ (commut. of $\vee$: $X \vee Y \equiv Y \vee X$).

And this is equivalent to $A \to B \quad (\neg X \vee Y \equiv X \to Y$ again).

And another

Let's show that $(p \vee \neg q) \wedge (p \vee q)$ is logically equivalent to p.

First instinct: 'multiply out'. You'll get
$(p \wedge p) \vee (p \wedge q) \vee (\neg q \wedge p) \vee (\neg q \wedge q)$,
and you can probably (eventually) simplify this to p.

But it's faster to go:

$(p \vee \neg q) \wedge (p \vee q)$ is equivalent to $p \vee (\neg q \wedge q)$
(by $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ used 'backwards').

This is equivalent to $p \vee \bot$ ($\neg A \wedge A \equiv \bot$).

And this is equivalent to p ($A \vee \bot \equiv A$).

So look out for 'backwards-uses' of equivalences.

4.4 Normal forms

Equivalences allow us to rewrite a formula in equivalent *normal form.* There are two common normal forms:

Definition 4.2 (normal forms DNF, CNF)

- *A formula is in* disjunctive normal form *if it is a disjunction of conjunctions of literals, and is not further simplifiable by equivalences without leaving this form.*

  Examples: $p \vee q \vee \neg r$.
  $(p \wedge \neg q) \vee r \vee (\neg p \wedge q \wedge \neg r)$.
  Non-example: $(p \wedge p) \vee (q \wedge \top \wedge \neg q)$.

- *A formula is in* conjunctive normal form *if it is a conjunction of disjunctions of literals (that is, a conjunction of clauses), and is not further simplifiable by equivalences without leaving this form.*

  Example: $(p \vee \neg q) \wedge (q \vee r) \wedge (\neg p \vee q)$.

See definition 2.3 for literals, clauses.

Rewriting a formula in normal form

1. Get rid of $\rightarrow, \leftrightarrow$:

   Replace all subformulas $A \rightarrow B$ by $\neg A \vee B$.
   Replace all subformulas $A \leftrightarrow B$ by
   $(A \wedge B) \vee (\neg A \wedge \neg B)$.

   It's faster to replace $\neg(A \rightarrow B)$ by $A \wedge \neg B$, and $\neg(A \leftrightarrow B)$ by
   $(A \wedge \neg B) \vee (\neg A \wedge B)$.

2. Then use the De Morgan laws to push negations down next to atoms. Delete all double negations (replace $\neg\neg A$ by A).

3. Now rearrange using distributivity to get the desired normal form.

4. Simplify: replace subformulas $p \wedge \neg p$ by $\bot$, and $p \vee \neg p$ by $\top$.
   Replace $\top \vee p$ by $\top$, $\top \wedge p$ by p, $\bot \vee p$ by p, and $\bot \wedge p$ by $\bot$.
   Equivalence 27 is often useful too. Repeat till no further progress.

Eg: write $p \wedge q \rightarrow \neg(p \leftrightarrow \neg r)$ in DNF

- $p \wedge q \rightarrow \neg(p \leftrightarrow \neg r)$ [the original formula]
- $\neg(p \wedge q) \vee ((p \wedge \neg\neg r) \vee (\neg p \wedge \neg r))$ [got by eliminating $\rightarrow, \neg\leftrightarrow$]
- $\neg p \vee \neg q \vee (p \wedge r) \vee (\neg p \wedge \neg r)$ [by De Morgan law & $\neg\neg A \equiv A$.]
- $\neg p \vee (\neg p \wedge \neg r) \vee (p \wedge r) \vee \neg q$ [by commutativity of $\vee$]
- $\neg p \vee (p \wedge r) \vee \neg q$ [$A \vee (A \wedge B) \equiv A$].
  This is in DNF.

We can simplify further if we are willing to leave DNF temporarily:

- $[(\neg p \vee p) \wedge (\neg p \vee r)] \vee \neg q$ [by distributivity]
- $[\top \wedge (\neg p \vee r)] \vee \neg q$ [$\neg A \vee A \equiv \top$]
- $\neg p \vee r \vee \neg q$ [$\top \wedge A \equiv A$]

The drastic simplification from the original shows the benefit of rewriting in normal form.

5. Proof systems: natural deduction

A *proof system* is a way of showing formulas to be valid by using purely syntactic rules — not using meaning at all. A computer should be able to apply the rules.

Modern automated reasoning is a growing and successful area.

There are many proof systems. We will do *natural deduction,* a system invented by F. B. Fitch (and related to work of Gentzen); it is a major topic and will take some time.

Pandora: www.doc.ic.ac.uk/pandora

Pandora is a lab program that allows you to do natural deduction proofs as a kind of computer game.

It was originally done by Dan Ehrlich as a 4th-year individual project, c. 1998. It has evolved since then, always developed by students, supervised by *Krysia Broda* and (in the past) Gabrielle Sinnadurai. It's becoming known worldwide.

I asked the 2005, 2006 and 2009 students to rate how useful they found Pandora in learning natural deduction. Responses:

	not tried	$1/5$	$2/5$	$3/5$	$4/5$	$5/5$	total
2005	6	4	7	15	31	12	75
2006	3	3	4	12	23	15	60
2009	2	1	5	14	37	29	86

So please try it, at least until you get the hang of it.

What is natural deduction?

- A formalisation of ‘direct argument’.
- Starting perhaps from some given facts — some formulas A_1, $\ldots$, A_n — we use the rules of the system to reason towards a formula B. (In practice, we reason forwards, backwards, and especially inwards.) If we succeed, we can write $A_1, \ldots, A_n \vdash B$.
- The reasoning works by writing down intermediate formulas using the rules. These formulas form the *proof* of B from the givens $A_1, \ldots, A_n$. Once established, they may be usable later. Each step of the proof is a valid argument.
- Mostly, there are two rules for each connective: one for *introducing* it (for bringing in a formula of which it is the principal connective), and one for *using* it (using a formula of which it is the principal connective). The rules are based on the semantics for the connectives given earlier.

5.1 Natural deduction rules for $\wedge, \rightarrow, \vee$

There are two rules for each of these connectives. The rules reflect the meanings of the connectives.
The easiest is $\wedge$ ('and').

Rules for $\wedge$

- ($\wedge$-introduction, or $\wedge I$) To introduce a formula of the form $A \wedge B$, you have to have already introduced A and B.

1	A	we proved this...
	$\vdots$	(other junk)
2	B	and this...
3	$A \wedge B$	$\wedge I(1,2)$

The line numbers are essential for clarity.

Rules for $\wedge$ (continued)

- ($\wedge$-elimination, or $\wedge E$) If you have managed to write down $A \wedge B$, you can go on to write down A and/or B.

1	$A \wedge B$	we proved this somehow
2	A	$\wedge E(1)$
3	B	$\wedge E(1)$

Assumptions

In ND proofs, we often need to temporarily '*assume* for the sake of argument that…'.

An *assumption* here is just a formula, but it's used in a special way. We use our imagination to conjure up situation(s) in which the formula holds. That is, we *assume* that (we are in a situation in which) the formula is true. Then we may make progress, because we know more about the situation.

Example ('Silver Blaze', A. Conan Doyle, 1893): Sherlock Holmes *assumes* for a moment that whoever took the valuable racehorse in the night was a stranger. Then, he is sure, the guard dog would have barked, which would have woken the stable-boys.

But the boys slept all night. Holmes deduces that the dog didn't bark in the night, so that whoever took the horse was no stranger.

VITAL NOTE: his assumption was (therefore) *wrong* — but he could still assume or imagine it mentally, and it was useful to do so.

Rules for $\to$

- ($\to$-introduction, $\to I$: 'arrow-introduction') To introduce a formula of the form $A \to B$, you *assume* A and then prove B.

  During the proof, you can use A as well as anything already established. But *you can't use A or anything from the proof of B from A later on* (because it's based on an extra assumption).
  So we isolate the proof of B from A, in a *box:*

1	A	ass
	⟨the proof⟩	hard struggle
2	B	we made it!
3	$A \to B$	$\to I(1,2)$

  Nothing inside the box can be used later.

In natural deduction, boxes are used when we make additional assumptions. The first line inside a box should always be labelled 'ass' (assumption) — with one exception, coming later ($\forall I$).

Rules for $\rightarrow$ (continued)

- ($\rightarrow$-elimination, or $\rightarrow E$) If you have managed to write down A and $A \rightarrow B$, in either order, you can go on to write down B. (This is modus ponens.)

1	$A \rightarrow B$	we got this somehow...
	$\vdots$	other junk
2	A	and this too...
3	B	$\rightarrow E(1, 2)$

Rules for $\vee$

- ($\vee$-introduction, or $\vee I$)
  To prove $A \vee B$, prove A, or (if you prefer) prove B.

1	A	proved this somehow
2	$A \vee B$	$\vee I(1)$

  B can be any formula at all!

1	B	proved this somehow
2	$A \vee B$	$\vee I(1)$

  A can be any formula at all.

Rules for $\vee$ (continued)

- ($\vee$-elimination, or $\vee E$) To prove something from $A \vee B$, you have to prove it by assuming A, AND prove it by assuming B.

  This is arguing by cases.

1	$A \vee B$				we got this somehow	
2	A	ass		5	B	ass
3	$\vdots$	the 1st proof		6	$\vdots$	the 2nd proof
4	C	we got it		7	C	we got it again
8	C					$\vee E(1,2,4,5,7)$

  The assumptions A, B are not usable later, so are put in (side-by-side) boxes. *Nothing inside the boxes can be used later.*

  C can be any formula at all; but both boxes must end with the same C.

5.2 Examples

With rules for $\wedge, \to, \vee$ we can already do quite a lot. (We still need rules for $\neg, \top, \bot$.)

Example 5.1 Let's prove $A \to A \vee B$ by natural deduction (where A, B are arbitrary formulas).

We did this earlier by direct argument (page 50). Compare the two proofs: they're quite similar!

1	A	ass
2	$A \vee B$	$\vee I(1)$
3	$A \to A \vee B$	$\to I(1,2)$

A similar English argument: 'In some situation, assume A is true. Then $A \vee B$ is true. So, $A \to A \vee B$ is true in *any* situation.'

The rules are natural. . . but they are still syntactic

Note how natural the rules are — they come from the meaning of the connectives.

HOWEVER: yes, the ND rules are *motivated* by the meaning of $\wedge, \vee, \ldots$, but nonetheless they are just *syntactic rules.*

A natural deduction proof is syntactic. A computer could do it, without knowing about 'meaning'.

More examples

Example 5.2 Given $A \to (B \to C)$, prove $A \wedge B \to C$.

1	$A \to (B \to C)$	given
2	$A \wedge B$	ass
3	A	$\wedge E(2)$
4	$B \to C$	$\to E(1,3)$
5	B	$\wedge E(2)$
6	C	$\to E(4,5)$
7	$A \wedge B \to C$	$\to I(2,6)$

English paraphrase: 'Suppose $A \to (B \to C)$ is true (in some situation). To show $A \wedge B \to C$ is true, assume $A \wedge B$ is also true (in this situation), and show C. Well, as $A \wedge B$, we have A, so using $A \to (B \to C)$, we see that $B \to C$ (is true). But $A \wedge B$ tells us B too, so we get C. Done.'

The notation '$\vdash$'

We proved $A \wedge B \to C$ from the 'given' $A \to (B \to C)$.
We can now write

$$A \to (B \to C) \ \vdash \ A \wedge B \to C.$$

Definition 5.3 *Let $A_1, \ldots, A_n, B$ be arbitrary formulas.*

$$A_1, \ldots, A_n \vdash B$$

means that there is a (natural deduction) proof of B, starting with the formulas $A_1, \ldots, A_n$ as 'givens'.

- $\vdash B$ (the case $n = 0$) means we can prove B with no givens at all. We then say that B is a 'theorem' (of natural deduction).
- Can read $A_1, \ldots, A_n \vdash B$ as 'B is provable from $A_1, \ldots, A_n$', and $\vdash B$ as 'B is a "theorem"'. '$\vdash$' is called 'single turnstile'.
- Do not confuse $\vdash$ with $\models$.
  $\vdash$ is syntactic and involves proofs.
  $\models$ is semantic and involves situations.

Another example

Example 5.4 $A \to C,\ B \to C \ \vdash \ A \vee B \to C$.

1	$A \to C$				given
2	$B \to C$				given
3	$A \vee B$				ass
4	A	ass	6	B	ass
5	C	$\to E(4,1)$	7	C	$\to E(6,2)$
8	C				$\vee E(3,4,5,6,7)$
9	$A \vee B \to C$				$\to I(3,8)$

In English: 'Suppose we're given $A \to C$ and $B \to C$. To show $A \vee B \to C$, assume we have $A \vee B$, and show C. Well, $A \vee B$ tells us we have A or B. If we have A, then remember we have $A \to C$ as well — so we have C. If we have B, then because we have $B \to C$, we have C. This covers all cases. So we have C anyway. Done.'

5.3 Rules for $\neg$

This is the trickiest case. Also, $\neg$ has three rules! The first two treat $\neg A$ like $A \to \bot$.

- ($\neg$-introduction, $\neg I$) To prove $\neg A$, you assume A and prove $\bot$.

  As usual, you can't then use A later on, so enclose the proof of $\bot$ from assumption A in a box:

1	A	ass
2	$\vdots$	more hard work, oh no
3	$\bot$	we got it!
4	$\neg A$	$\neg I(1,3)$

Rules for $\neg$ (continued)

- ($\neg$-elimination, $\neg E$): From A and $\neg A$, deduce $\bot$.

1	$\neg A$	proved this somehow...
2	$\vdots$	junk
3	A	...and this
4	$\bot$	$\neg E(1,3)$

- ($\neg\neg$-elimination, $\neg\neg$): From $\neg\neg A$, deduce A.

1	$\neg\neg A$	proved this somehow...
2	A	$\neg\neg(1)$

See example 5.8 for a real example of $\neg\neg$ in use.

5.4 Examples of ¬-rules in action

Example 5.5 (Holmes and the racehorse)

- stranger = ‘whoever took the valuable racehorse in the night was a stranger’
- barked = ‘the dog barked in the night’
- woke = ‘the stable-boys woke up’

1	stranger $\rightarrow$ barked	given
2	barked $\rightarrow$ woke	given
3	¬woke	given
4	stranger	ass
5	barked	$\rightarrow E(4,1)$
6	woke	$\rightarrow E(5,2)$
7	$\bot$	$\neg E(3,6)$
8	¬stranger	$\neg I(4,7)$

Another example

Example 5.6 Let's prove $A \vdash \neg\neg A$.

1	A	given
2	$\neg A$	ass
3	$\bot$	$\neg E(1,2)$
4	$\neg\neg A$	$\neg I(2,3)$

And another...

Example 5.7 Now prove $\neg(A \vee B) \vdash \neg A$.

1	$\neg(A \vee B)$	given
2	A	ass
3	$A \vee B$	$\vee I(2)$
4	$\bot$	$\neg E(1,3)$
5	$\neg A$	$\neg I(2,4)$

In English: 'Given $\neg(A \vee B)$, if you had A, then you'd have $A \vee B$, which is a contradiction. So you've got $\neg A$.'

This is another illustration of the 'Law of Excluded Middle': if you haven't got A, you've got $\neg A$.

This law is true of classical logic, but not of some other logics.

And finally. . . you can prove anything from $\bot$!

Example 5.8 Let's show $\bot \vdash A$ *for any* A.

1	$\bot$	given
2	$\neg A$	ass
3	$\bot$	$\checkmark(1)$
4	$\neg\neg A$	$\neg I(2,3)$
5	A	$\neg\neg(4)$

In English: 'Suppose we are in a situation where $\bot$ is true. If $\neg A$ is true then $\bot$ is true (as we knew already), and this is impossible. So $\neg\neg A$ and hence A are true in this situation'.
(Note the use of $\checkmark$ in line 3 to justify a formula that's already available.)

Anything follows from falsity!
Anything follows from a contradiction!

Tricky, eh?

You are going to find the argument above hard to accept. You will not be alone: it has led some to reject or reformulate classical logic.

The effect of dividing by zero in arithmetic is the same. It leads to paradox — that is, $\perp$.

Discussion

I could just say that we are just following proof rules. If the rules allow us to get from $\bot$ to A, then fine.

Better: the English paraphrase says SUPPOSE we are in a situation in which $\bot$ is true... then A is true. (And $\neg A$ is true too, so A is false,...)
But there is no such situation. So you can't dispute the crazy things that follow from this supposition.
Compare: $A \to B$ is true if A is false.

The practical point is that there is no situation in which $\bot$ is true, so you can only prove $\bot$ under (contradictory) assumptions in a box in the middle of a bigger proof. You often get such assumptions — e.g., in an argument by cases $(\vee E)$. You can then deduce anything you want, so the box is no obstacle to the overall goal. So $\bot$ is a useful formula to aim to prove. See example 5.11 below.

Proving $A \vee \neg A$

Example 5.9 Let's prove $\vdash A \vee \neg A$. This is very useful, but the proof involves more $\neg$s and is moderately complicated.

1	$\neg(A \vee \neg A)$	ass
2	A	ass
3	$A \vee \neg A$	$\vee I(2)$
4	$\bot$	$\neg E(1,3)$
5	$\neg A$	$\neg I(2,4)$
6	$A \vee \neg A$	$\vee I(5)$
7	$\bot$	$\neg E(1,6)$
8	$\neg\neg(A \vee \neg A)$	$\neg I(1,7)$
9	$A \vee \neg A$	$\neg\neg(8)$

In English: 'Assume $\neg(A \vee \neg A)$. If A, then $A \vee \neg A$, which is impossible. So $\neg A$ (line 5). But then, $A \vee \neg A$, again impossible.

So the assumption was wrong, and so $\neg\neg(A \vee \neg A)$. Therefore, $A \vee \neg A$. Done.'

5.5 Rules for $\top$

- ($\top$-introduction, $\top I$) You can introduce $\top$ anywhere (for all the good it does you).
- ($\top$-elimination, $\top E$) You can prove nothing new from $\top$, sorry!

5.6 Rules for $\bot$

- ($\bot$-introduction, $\bot I$) To prove $\bot$, you must prove A and $\neg A$ (for any A you like).

1	A	got this somehow
$\vdots$		
2	$\neg A$	and this
3	$\bot$	$\bot I(1,2)$

  This is the same rule as $\neg E$. (There are two names for this rule!)

- ($\bot$-elimination, $\bot E$) You can prove any formula at all from $\bot$! (See example 5.8.)

1	$\bot$	got this somehow...
2	A	$\bot E(1)$

The rules for $\bot$ can actually be obtained from the other rules already explained.

5.7 Lemmas

A *lemma* is something you prove that helps in proving what you really wanted.

In example 5.9 we proved $\vdash A \vee \neg A$. This is useful in all sorts of proof. It divides the argument into two cases (A, and $\neg A$). This makes it easier, since we know more in each case.

So in ND proofs, I'll generally allow you to quote $A \vee \neg A$ as a lemma, without proving it. Justify by *'Lemma'.*

It's the only lemma I'm going to give you. . . so remember it well.

But always *you* have to choose which A to use.

Warning: Pandora calls this particular lemma 'EM' (standing for Excluded Middle). Click on the 'EM' button to use it.

The Pandora 'lemma' button just chops a proof into two halves (try it!). It lets you make your own lemmas. Don't use it if you want $A \vee \neg A$ — use the 'EM' button!

Example 5.10 Let's prove $\vdash ((A \to B) \to A) \to A$ (Peirce's law). (We did this earlier by direct argument.)

1	$(A \to B) \to A$					ass
2	$A \vee \neg A$					lemma
3	A	ass	5	$\neg A$		ass
			6	A		ass
			7	$\bot$		$\neg E(5,6)$
			8	B		$\bot E(7)$
			9	$A \to B$		$\to I(6,8)$
4	A	$\checkmark(3)$	10	A		$\to E(9,1)$
11	A					$\vee E(2,3,4,5,10)$
12	$((A \to B) \to A) \to A$					$\to I(1,11)$

5.8 Rules for $\leftrightarrow$

Roughly, we treat $A \leftrightarrow B$ as $(A \to B) \wedge (B \to A)$.

- ($\leftrightarrow$-introduction, $\leftrightarrow I$)
  To prove $A \leftrightarrow B$, prove both $A \to B$ and $B \to A$.

1	$B \to A$	proved this somehow...
	$\vdots$	
2	$A \to B$	...and this
3	$A \leftrightarrow B$	$\leftrightarrow I(1, 2)$

Rules for $\leftrightarrow$ (continued)

- ($\leftrightarrow$-elimination, $\leftrightarrow E$)
  From $A \leftrightarrow B$ and A, you can prove B.
  From $A \leftrightarrow B$ and B, you can prove A.

1	$A \leftrightarrow B$	proved this somehow...
2	A	...and this
3	B	$\leftrightarrow E(1,2)$

1	$A \leftrightarrow B$	proved this somehow...
2	B	...and this
3	A	$\leftrightarrow E(1,2)$

5.9 Derived rules

These are NOT necessary, but they do help to speed proofs up.

A good one is PC (Proof by Contradiction):

To prove A, assume $\neg A$ and prove $\bot$.

The effect of PC is to combine applications of $\neg I$ and $\neg\neg$:

1	$\neg A$	ass
2	$\vdots$	hard slog
3	$\bot$	got it!
4	$\neg\neg A$	$\neg I(1,3)$
5	A	$\neg\neg(4)$

Can replace this by:

1	$\neg A$	ass
2	$\vdots$	as before
3	$\bot$	got it!
4	A	$PC(1,3)$

Using PC cuts out a line. This is surprisingly helpful.

5.10 Advice for doing natural deductions

1. Learn the rules. Practise! Use Pandora.
2. Think of a direct argument to prove what you want. Then translate it into natural deduction.
3. If really stuck in proving A, it can help to:
   - Assume $\neg A$ and prove $\bot$. ('Reductio ad absurdum', or 'proof by contradiction', PC.)
     We did this when proving $\vdash A \vee \neg A$ in example 5.9.
   - Use the lemma $B \vee \neg B$, for suitable B. It amounts to arguing by cases. See example 5.10 for an application of this lemma.

The 'Reasoned Programming' book and `www.doc.ic.ac.uk/~imh/teaching/140_logic/advice.pdf` gives more advice, but the above is the most important.

A typical nasty-ish example

Example 5.11 Let's show

$$A \vee B, \quad \neg C \to \neg A, \quad \neg(B \wedge \neg C) \quad \vdash \quad C.$$

Baffled, eh?

Well, assume for the sake of argument that you had $\neg C$.
Then you'd have $\neg A$ — but you're given A or B, so you get B.
Now you've got both B and $\neg C$, which you're told you don't:
contradiction.

SO, you must have C.

This is quite easy to translate into ND. There's a scuffle with the $\vee$, but you get used to ND's idiosyncrasies like this.

Example 5.11: $A \vee B,\ \neg C \to \neg A,\ \neg(B \wedge \neg C) \ \vdash\ C$

1	$A \vee B$		given
2	$\neg C \to \neg A$		given
3	$\neg(B \wedge \neg C)$		given
4	$\neg C$		ass
5	$\neg A$		$\to E(2,4)$
6	A	ass	
7	$\bot$	$\neg E(5,6)$	
8	B	$\bot E(7)$	
9	B	ass	
10	B	$\checkmark(9)$	
11	B		$\vee E(1,6,8,9,10)$
12	$B \wedge \neg C$		$\wedge I(11,4)$
13	$\bot$		$\neg E(3,12)$
14	C		$PC(4,13)$

Boxes: things to remember

A box is 'its own little world' with its own assumptions. Care is needed.

1. A box *always* starts with an assumption (the *only* exception is in $\forall I$ in predicate logic).
2. An assumption can *only* occur on the first line of a box.
3. Inside a box, you can use any earlier formulas (except formulas in completed earlier boxes — see (5)).
4. The *only* ways of exporting information from a box are by the rules $\rightarrow I$, $\vee E$, $\neg I$, and PC (and also $\exists E$ and $\forall I$ in predicate logic). The first line after a box *must* be justified by one of these.
5. No formula inside a box can be used outside, except via (4).

You can check these conditions for yourself. If your box doesn't meet them, your proof is wrong.

Example of how not to use boxes: $\neg A \vdash \neg A$ (!)

1	$\neg A$	given	
2	$\neg A$	$\checkmark(1)$	the best proof!

1	$\neg A$	given	
2	A	ass	
3	$\bot$	$\neg E(2,1)$	
4	$\neg A$	$\neg I(2,3)$	correct but silly

1	$\neg A$	given	
2	A	ass	
3	$\bot$	$\neg E(2,1)$	
4	$\neg A$	$\bot E(3)$	←WRONG→

1	$\neg A$	given	
2	A	ass	
3	$\bot$	$\neg E(2,1)$	
4	$\neg A$	$\bot E(3)$	
5	$\neg A$	$\checkmark(4)$	

Variants (see exam questions!!)

The ND system we've seen can be varied by

- changing the rules (carefully!)
- introducing new connectives and giving rules for them.

E.g. (exam 2007): The IF connective can be defined as

$$IF(p, q, r) = (p \to q) \wedge (\neg p \to r).$$

Here's an introduction rule for IF, based on the rules $\to I$ and $\wedge I$:

1	A	ass	3	$\neg A$	ass
	$\vdots$			$\vdots$	
2	B	got this somehow...	4	C	...or other
5	$IF(A, B, C)$			$IFI(1, 2, 3, 4)$	

Exercise: what would a good elimination rule (or rules) be?

5.11 $\vdash$ versus $\models$

Our main concern is $\models$:
recall $A_1, \ldots, A_n \models B$ if B is true in all situations in which $A_1, \ldots, A_n$ are true.

$\vdash$ is useless unless it helps to establish $\models$.

Definition 5.12 *A theorem is a formula that can be established ('proved') by a given proof system.*

We are concerned with natural deduction. So a theorem is any formula A such that $\vdash A$.

Definition 5.13 *A proof system is sound if every theorem is valid, and complete if every valid formula is a theorem.*

Soundness and completeness

In fact, it can be shown that natural deduction is sound and complete:

Theorem 5.14 (soundness) *Let $A_1, \ldots, A_n, B$ be any propositional formulas. If $A_1, \ldots, A_n \vdash B$, then $A_1, \ldots, A_n \models B$.*

Slogan (for the case $n = 0$):
'Any provable propositional formula is valid.'
'Natural deduction never makes mistakes.'

Theorem 5.15 (completeness)
Let $A_1, \ldots, A_n, B$ be any propositional formulas. If $A_1, \ldots, A_n \models B$, then $A_1, \ldots, A_n \vdash B$.

Slogan (for $n = 0$):
'Any propositional validity can be proved.'
'Natural deduction is powerful enough to prove all valid formulas.'

So we can use natural deduction to check validity.

Consistency

This is an important (fundamental) logical idea.

Definition 5.16 (consistency) *A formula A is said to be* consistent *if* $\nvdash \neg A$.

A collection $A_1, \ldots, A_n$ of formulas is said to be consistent *if* $\nvdash \neg \bigwedge_{1 \leq i \leq n} A_i$.

One can extend this definition to infinite collections of formulas too.

By soundness and completeness (theorems 5.14 and 5.15), we get:

Theorem 5.17 *A formula is consistent if and only if it is satisfiable.*

See definition 3.3 for 'satisfiable'.

Interlude: very sketchy history of logic

Logic has been studied since ancient Greek times (especially Aristotle). It began as analysis of human reasoning, to put notion of 'valid and invalid argument' (the 'laws of thought') on a solid foundation. Some say it should return to these roots...

Aristotle laid down logical principles called *syllogisms,* (eg 'All women are human, no human is immortal, *so* no woman is immortal') which were taught and used to analyse/verify arguments through the middle ages. (See Lemmon's book.) Famous medieval logicians include Peter Abelard, William of Ockham, and Ibn Sīnā.

In the 19th century, algebraic versions of logic were developed by, e.g., George Boole. People realised the limitations of 'the syllogism', and algebraic-style logics to handle binary relations (eg 'x is a daughter of y') were developed. The algebraic tradition survives today but is no longer so popular.

Around the end of the 19th century, Gottlob Frege developed *quantifier logic* and much of modern logic depends on his work, even though Bertrand Russell wrote him a letter showing his system to be inconsistent. Charles S. Peirce (pronounced 'Purse') independently followed a similar path about the same time.

The 'Russell paradox' led to a crisis in the foundations of mathematics, which was resolved slowly over the early 20th century. First-order logic (and set theory) now serves as the chief foundation for modern mathematics.

The limitations of first-order logic (as a result of its being too powerful) were revealed by work of Kurt Gödel, Alan Turing, and Alonzo Church in the 1930s.

Gödel also proved soundness and completeness of a deductive system for first-order logic (1929). Alfred Tarski gave semantics (formal meaning) for it (1933–1950s). Gentzen and others extended its proof theory. At this point, first-order logic takes its modern form.